Edition 3/2020 | May & June | www.elektormagazine.com

elektor

elektor magazine

DESIGN ⊳ SHARE ⊳ SELL ELECTRONICS

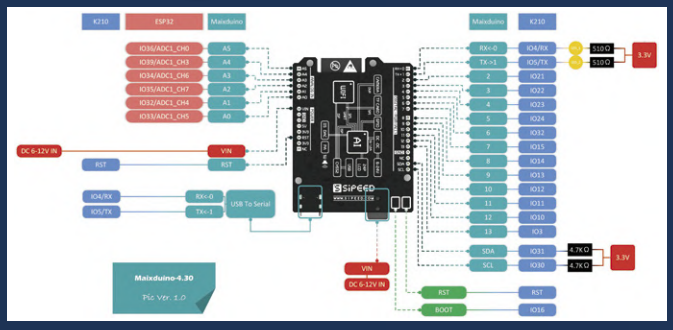
Open-Network Weather Station Mk.2

Measure, log, stream, multi-sensor environmental data to ThingSpeak, openSenseMap



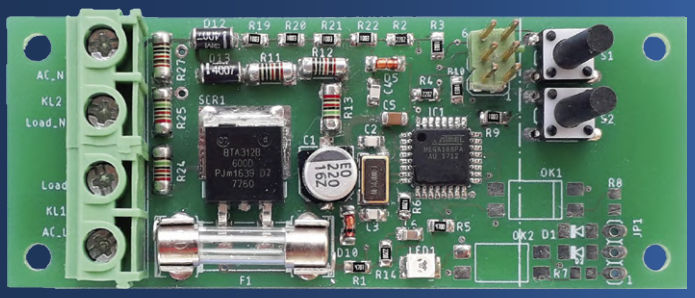
New Series

Artificial Intelligence for Beginners



Object recognition using the Maixduino board

Universal Triac Control Module with ATmega



Switch and dim a variety of loads

Analogue Electronics Design ⊳ BalBot: a Self-Balancing Robot ⊳ BASIC for the ESP32/ESP8266 ⊳ CAN Bus + Arduino for Solar PV Cell Monitoring ⊳ Elektor 1.2 GHz Multifunction Frequency Meter (1992/93) ⊳ Elektor New Precise Nixie Clock Revisited ⊳ How-To: GitHub For Dummies ⊳ Inexpensive E-Scooter ⊳ Joy-IT HD35 Electronic USB Load ⊳ My IoT Button: A Button for the Web ⊳ Open-Network Weather Station Mk.2 ⊳ PeakTech 6080 A Lab Power Supply ⊳ Practical ESP32 Multitasking ⊳ Safe products in the IoT era ⊳ SigFox and the IoT ⊳ Small Circuits Revival ⊳ Starting Out in Electronics ⊳ Tamper-Evident Box ⊳ TMS0280 Speech Synthesizer ⊳ ...and more

IoT Home Hacks with ESP8266



NOW € 31,46 for members

Description:

Ready to start building Internet of Things (IoT) solutions? With a little know-how and the right tools, such as the ESP8266 Wi-Fi module, you can design and build handy IoT solutions for your home or office. In IoT Home Hacks with ESP8266, Hans Henrik Skovgaard introduces the ESP8266 – in the form of the WeMos D1 Mini Pro – and then covers a variety of IoT hacks that you'll find immediately useful.

You'll learn to build the following and much more!

- Colorful smart home accessory
- Refrigerator controller
- 230-V power monitor
- Door lock monitor

The custom software for the IoT devices and PCB layouts are available for free at Elektor.com. Order the book today!

elektor

Read more on:

www.elektor.com/iot-home-hacks-with-esp8266

Elektor Magazine,
English edition

Edition 3/2020
Volume 46, No. 501
May & June 2020

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

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@elektor.com

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

Copyright Notice
The circuits described in this magazine are for domestic and educational use only. All drawings, photographs, printed circuit board layouts, programmed integrated circuits, disks, CD-ROMs, DVDs, software carriers, and article texts published in our books and magazines (other than third-party advertisements) are copyright Elektor International Media b.v. and may not be reproduced or transmitted in any form or by any means, including photocopying, scanning and recording, in whole or in part without prior written permission from the Publisher. Such written permission must also be obtained before any part of this publication is stored in a retrieval system of any nature. Patent protection may exist in respect of circuits, devices, components etc. described in this magazine. The Publisher does not accept responsibility for failing to identify such patent(s) or other protection. The Publisher disclaims any responsibility for the safe and proper function of reader-assembled projects based upon or from schematics, descriptions or information published in or in relation with Elektor magazine.

© Elektor International Media b.v. 2020
Printed in the Netherlands



A well-developed network — more important than ever!

Just a few weeks ago I had planned to report about the Embedded World 2020 trade show in this editorial space. As always, I discovered interesting products there. But the show, held at the end of February, now seems to be a thing of the distant past. In the meantime, we got used to queueing in front of supermarkets and DIY stores, and seeing our colleagues on video only.
On the other hand, videos from last year on the Internet now seem almost surreal to me — lectures on electronics to a closely packed audience, or music festivals with cheering crowds.
However, life goes on. With a little creativity you can network even in these times. One of my colleagues recently celebrated his birthday using the 'Zoom' video platform. Among his 'guests' were many electronics engineers, some of whom had joined in from their lab. Personally, together with a few friends I am about to broadcast a regular DJ livestream, from living rooms but properly equipped technically. You may also have ideas on how to continue exchanging ideas with like-minded people in order to advance a common project. Elektor offers a platform for electronics projects at www.elektor-labs.com, which was recently given a new look, a faster engine and extended functionality.
Elektor's English edition quietly reached number 500 in March 2020 and Elektor Germany turns 50 this month. Both are unique successes in the industry, precisely because many readers have remained loyal to us for decades. But especially in these times, this should not be a reason for Elektor to rest on its laurels. On the contrary, wires and brain cells are glowing. For example, I can divulge that new offers and media formats are in active development to inform all electronics technicians in an even more targeted way.
Stay tuned and above all, stay healthy!

Jens Nickel
International Editor-in-Chief, Elektor Magazine

The Circuit

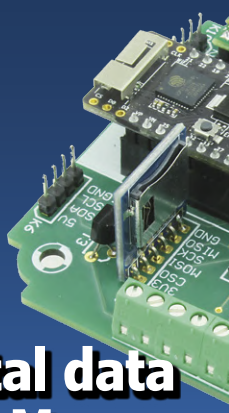
International Editor-in-Chief:	Jens Nickel
Editor-in-Chief, English edition:	Jan Buiting
Membership Manager:	Denise Bodrone
International Editorial Staff:	Eric Bogers, Rolf Gerstendorf, Denis Meyer, Dr Thomas Scherer, Clemens Valens
Laboratory Staff:	Mathias Claussen, Ton Giesberts, Luc Lemmens, Clemens Valens, Jan Visser
Graphic Design & Prepress:	Giel Dols
Publisher:	Don Akkermans

This Edition

Volume 47 – Edition 3/2020
No. 501 May & June 2020

Open-Network Weather Station Mk.2

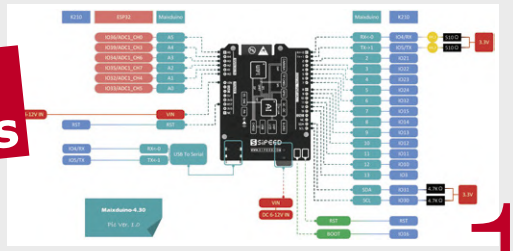
6



Measure, log, stream, multi-sensor environmental data to ThingSpeak, openSenseMap

Artificial Intelligence for Beginners

New Series



Object recognition using the Maixduino board

12

Regulars

26 How-To: GitHub For Dummies
How to download something from GitHub

35 Peculiar Parts, the series
TMS0280 Speech Synthesizer

36 Interactive
Corrections & Updates || Questions & Answers

42 Developer's Zone
Tips & Tricks, Best Practices and Other Useful Information

50 Starting Out in Electronics (3)
Easier than imagined!

52 Small Circuits Revival
Capita Selecta from the Elektor Project Suggestions Box

90 Analogue Electronics Design (3)
Case Study #1 — Section 3: Preamplifier response optimization and trade-offs

98 Steeped in Electronics
Lead-free soldering and EU regulatory zeal



100 Lab, Sweet Lab
A glimpse of The Holy Place — no unauthorized entry.

104 Elektor Store
What's 4 Sale @ www.elektor.com

106 Beyond Electronics
A Big Challenge: safe products in the IoT era

108 Retronics: Elektor 1.2 GHz Multifunction Frequency Meter (1992/93)
Crossing the magic 1 GHz border

114 Hexadoku
The original Elektorized Sudoku

Features

12 Artificial Intelligence for Beginners (1)
Object recognition using the Maixduino board

81 Start-Up Update
The Elektor Investment Program

82 Review:
Joy-IT HD35 Electronic USB Load
Test & verify USB power performance

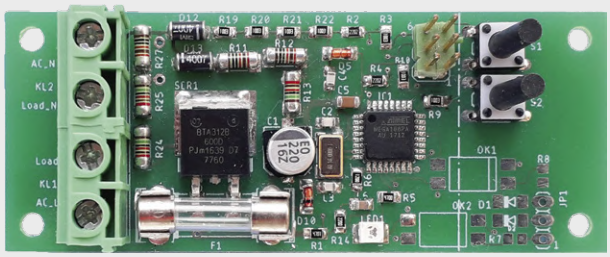
Retronics

Elektor 1.2 GHz Multifunction Frequency Meter

(1992/93)

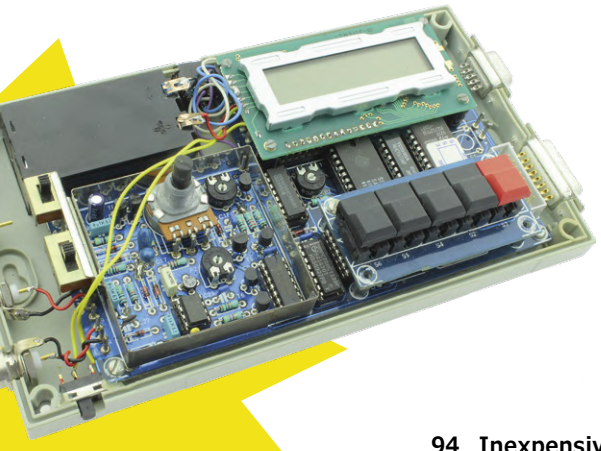


Universal Triac Control Module with ATmega



18

Switch and dim a variety of loads



108

Projects

6 Open-Network Weather Station Mk.2
Part 1: Introduction and hardware elements

18 Universal Triac Control Module with ATmega
Switch and dim a variety of loads

28 My IoT Button: A Button for the Web (2)
Part 2: Prototyping with board and cloud

38 BASIC for the ESP32/ESP8266 (2)
An hourglass with the ESP8266 and Annex WiFi RDS

45 BalBot: a Self-Balancing Robot
Modern acceleration sensors ease the construction process

54 Tamper-Evident Box
Send data securely through postal services

62 Elektor New Precise Nixie Clock Revisited
An engineering bulletin for enthusiastic users new and old

68 Practical ESP32 Multitasking (3)
Software timers

72 SigFox and the IoT (4)
Setting up a dashboard

86 CAN Bus + Arduino for Solar PV Cell Monitoring
Detect and locate serviceable panels in large arrays

94 Inexpensive E-Scooter
How does Lidl's „road approved" 300-euro E-scooter perform?

101 Review: The PeakTech 6080 A Lab Power Supply
A neat lab PSU for not much money

Next Edition

Elektor Magazine Edition 4/2020 (July & August)

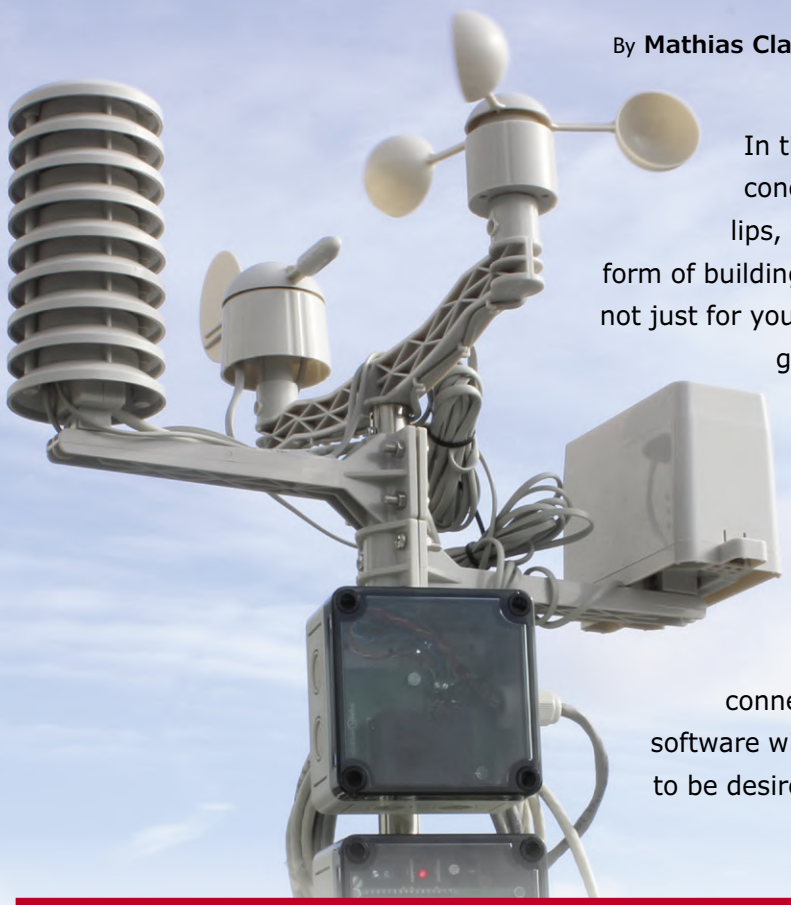
A Wireless Temperature Sensor for the Nixie Bargraph Thermometer • Home Information System • Advanced LCR Meter Sneak Preview • Open-Network Weather Station Mk.2 Software • S/PDIF Converter • Analogue Electronics Design (4) • HV Power supply with Curve Tracer • LED Booster for Microcontrollers • Home Automation Made Easy • Node Red Book • Small Circuits & Tricks • Retronics • and more.

Elektor Magazine edition 4/2020 covering July & August 2020 is published around 3 July 2020. Delivery of printed copies to Elektor Gold Members is subject to transport. Contents and article titles subject to change.

eLABS PROJECT

Open-Network Weather Station Mk.2

Part 1: Introduction and hardware elements



By **Mathias Claußen** and **Luc Lemmens** (Elektor Labs)

In these times of climate change and global concerns about the environment on everyone's lips, you too can make a positive contribution in the form of building a powerful and extendable weather station, not just for yourself and your 'local log' but also as part of global, open networks like ThingSpeak and openSenseMap. The station described in this 2-, possibly 3-installment article was designed with flexibility in mind as far as measuring and logging weather data is concerned, linked to acceptable power consumption and high reliability out there exposed to the elements. The project revolves around an ESP32 module connected to off the shelf sensors and running free software with a lot of hooks for you to link to. Leaving little to be desired except some sunshine and a light breeze!

Features

- 3-PCB system: Main; Connector; Dust/Particle
- TH components and commercial modules
- Sensors (basic system): temperature, relative humidity, air pressure, dust/particulate matter
- ESP32 based (Pico Kit)
- I²C, I²C bus
- 12-V (solar) battery powered with automatic low voltage shutdown
- Screw terminal connectors for most wiring
- IP66-class cases for Main Board and Dust/Particle Board
- Open-source software
- Compatible with ThingSpeak, openSensemap
- Extendable with custom sensors
- RFM95 LoRa extension (optional, in development)
- ESPHome and Home Assistant compatible
- WiFi accessible
- SD card for local control

Although its title includes "Mk.2" (mark-2), this project is a fundamental redesign of the ESP32 Weather Station [1], which is no longer recommended for new construction. The PCBs for the present project were designed to match the mechanical parts of the weather station kit available in the Elektor Store, which includes an installation rack and sensors for wind speed, wind direction, rain, temperature, and relative humidity. In terms of hardware, the present project is split up into three PCBs:

- Main board no. 191148-1 for installing outdoors in a waterproof case mounted relatively close to your home.

- Connector board no. 191148-2 designed to fit into the case of a thermo/hygro sensor, i.e. replacing the original internal hardware. It also provides RJ11 sockets for connecting the wind and rain sensor modules. The -1 and -2 boards are linked by 7 wires in a conventional Ethernet cable (sporting 8 wires).
- Dust sensor board no. 191148-3, which is no more than an adapter board to alleviate connecting the Honeywell type HPMA115 dust sensor, enabling it to be mounted remotely in the same case type as used for the weather station's main PCB.

These three boards will be discussed in this article, deferring the much larger part on the software to Part 2, possibly with a "stack overflow" to Part 3 once the programmers start talking.

Main board (no. 191148-1)

The main board contains a number of elements common to the sensors employed in the weather station. The elements are easily identified, even functionally, in the schematic in **Figure 1**.

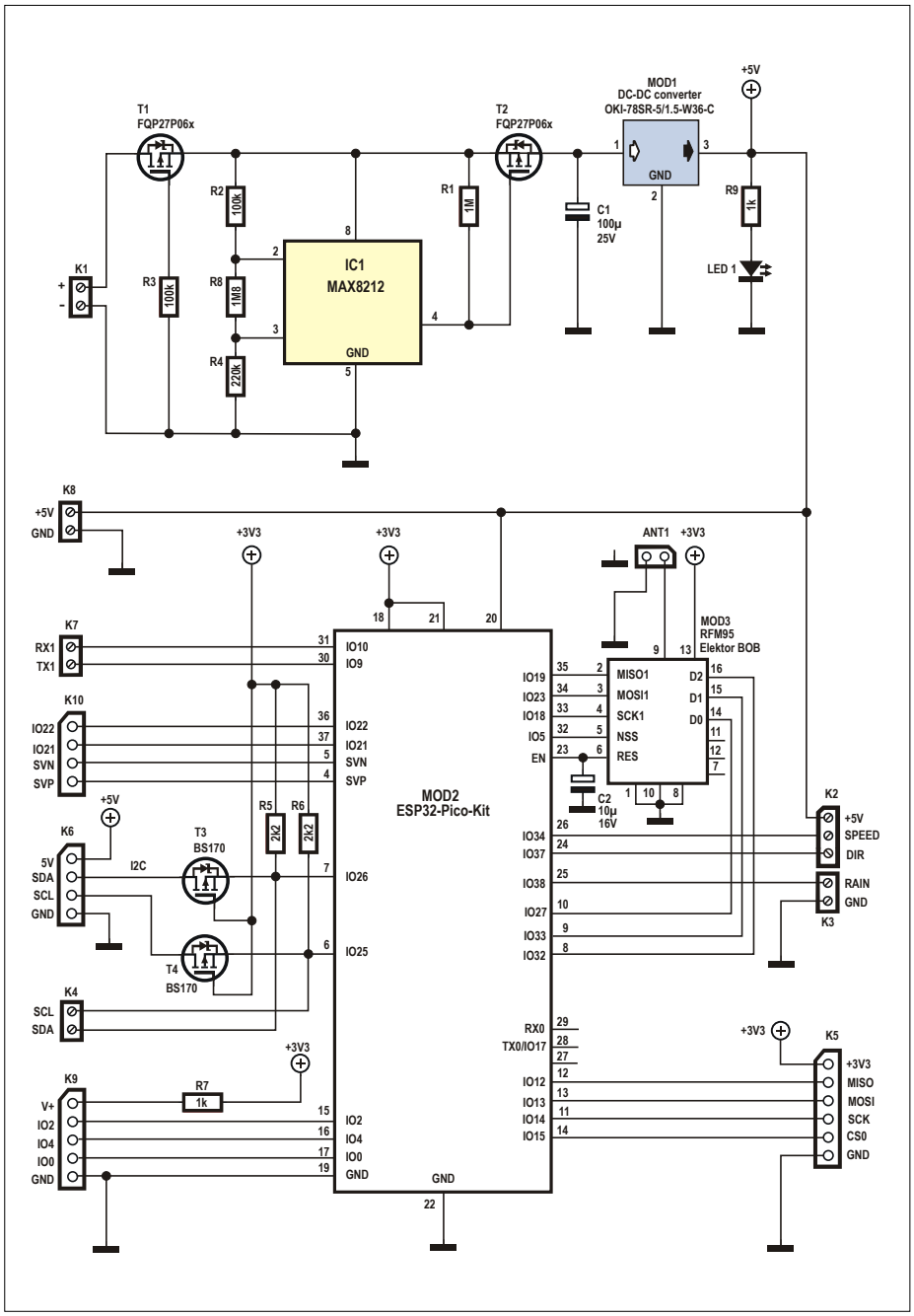


Figure 1: Schematic of the Main Board in the Open-Network Weather Station Mk.2 that's compatible with global sensorics platforms like ThingSpeak and openSenseMap. Note the bare-bones but perfectly working bidirectional 3.3-V/5-V, FET-based level shifters on the I²C bus.

PROJECT DECODER

sensors · weather · ESP32 · ThingSpeak · openSenseMap

entry level
intermediate level
expert level

3 hours approx.
(PCB assembly only)

lab tools, small drill

€60 (main board incl. enclosure)
€40 (cone including BME280),
€50 (dust sensor incl. case)
(prices approx.)

Power supply with polarity protection and undervoltage lockout

The weather station is normally powered by a 12 V rechargeable battery (-system), which gets its charge current from a solar panel combined with an external solar charger. A maximum current draw of 1.5 A is specified for the weather station, heavily depending on type and number of sensors used. The usable input voltage range is approximately 11–14 VDC.

FET T1 and resistor R3 afford battery polarity protection, the MOSFET conducting only if the battery terminals are connected the right way around.

If — for whatever reason — the charging process or the powering device fails to produce enough energy to compensate for the power inevitably dissipated by the weather station, the energy source must be disconnected to prevent deep discharging, which is a common source of battery failure. For this task we selected the good old programmable voltage detector type ICL8212, which today equals its CMOS successor type MAX8212 from Maxim Integrated. With only three external resistors for adjusting thresholds/hysteresis and one external p-channel MOSFET as a switching device, the '8212 does exactly what we want: avoid any risk of deep discharge of the battery.

With the given values for R2, R8 and

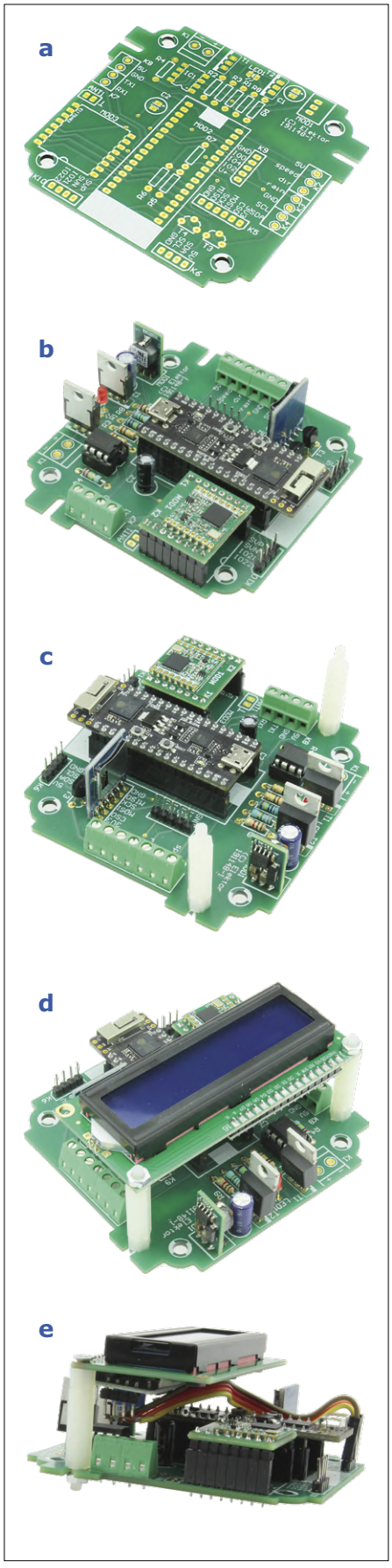


Figure 2: The main board at various stages of completion: (a) unpopulated; (b) with components populated and modules inserted; (c) with standoffs mounted; (d) with display mounted; (e) with internal cables connected.

R4 we have an 'undervoltage lockout' which disconnects the battery when its terminal voltage drops below 10.5 V and connects it back in circuit again at a level exceeding 11.1 V.

A DC/DC converter (MOD1) steps the 12 V battery voltage down to the 5 V regulated supply voltage for the 'real' electronics, with LED1 acting as the Power On indicator. Note that the ESP32 Picokit module sports an internal 3.3 V regulator with its output bonded out to pins 18 and 21 and furnishing a 3.3 V supply rail for use by other elements on the main board.

Brains & brawn from an ESP32

The ESP32 is a cheap yet powerful microcontroller with an enormous amount of ready-to-run software snippets and other

COMPONENT LIST

Main Board no. 191148-1 v.1

Resistors

R1 = 1MΩ, carbon film, 5%, 0.25W, 250V
R2,R3 = 100kΩ, carbon film, 5%, 0.25W, 250V
R4 = 220kΩ, carbon film, 5%, 0.25W, 250V
R5,R6 = 2.2kΩ, carbon film, 5%, 0.25W, 250V
R7,R9 = 1kΩ, carbon film, 5%, 0.25W, 250V
R8 = 1.8MΩ, carbon film, 5%, 0.25W, 250V

Capacitors

C1 = 100µF, 25V, radial 2.5mm
C2 = 10µF, 16V, radial 5mm

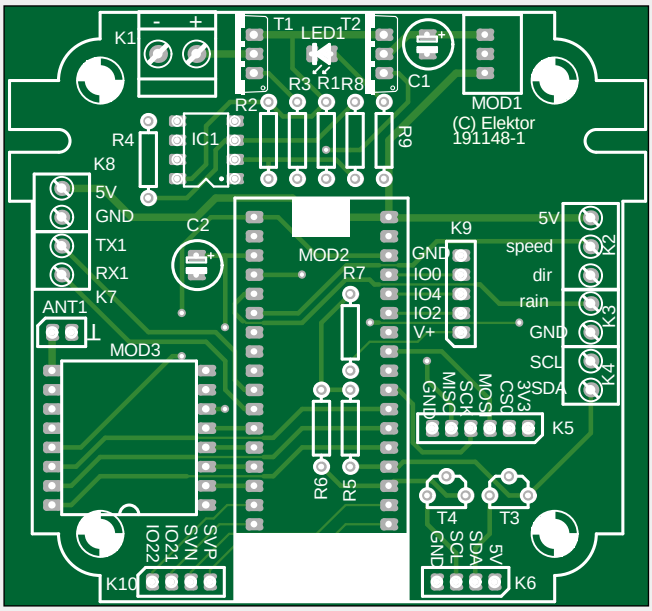
Semiconductors

IC1 = MAX8212EPA+ voltage supervisor
LED1 = red, 3 mm
T1,T2 = FQP27P06X, -60V, -27A, 120W, V_{gs}=-4V, R_{dson}=70mΩ
T3,T4 = BS170, 60V, 500mA, 830mW, V_{gs}=2.1V, R_{dson}=1.2Ω

Miscellaneous

K1 = 2-way PCB screw terminal block, 0.2 in. (5.08mm) pitch, 630V
K2 = 3-way PCB screw terminal block, 0.15 in. (3.81mm) pitch, 300V
K3,K4,K7,K8 = 2-way PCB screw terminal block, 0.15 in. (3.81mm) pitch, 300V
IC1 = DIL-8 IC socket
MOD2 = 17-way SIL pinheader socket, 0.1" pitch
K5 = Micro SD card BoB, from Elektor Store*
K6,K10 = 4-pin SIL pinheader, snapoff, vertical
K9 = 5-pin SIL pinheader, snapoff, vertical
LCD1 = 2x16 character backlit LCD with I²C Backpack, from Elektor Store *
MOD1 = OKI-78SR-5/1.5-W36-C, 5V 1.5A DC/DC converter, (Murata)
MOD2 = ESP32 Pico KIT V4, from Elektor Store
MOD3 = Elektor RFM95 BoB (optional for future extension)
Case: TK PS 94×94×57 mm, IP66 class (Spelsberg)
S1 = pushbutton switch, IP68, Alcoswitch PB6B2FM3M1CAL07
2 pcs hex spacer M3×25mm
2 pcs M3 nut
2 pcs M3×10mm machine screw
2 pcs PG7 Cable Gland Nylon, IP68
6 pcs short jumper wire, female-female
PCB 191148-1 V1.2 from Elektor Store

* Stock/shipping issues expected due to production and export issues.



'soft' building blocks available on the web. It also has good properties as far as sensor interfacing is concerned. Here, a drop-in ESP32 Picokit module (MOD2) is used to control and read out the sensors, display information on the LCD, write logged data on the SD card, and communicate with an (indoor) computer via WiFi (or LoRa, once we have it implemented). The main board also contains a Reset and a User button, plus a USB connection for flashing the weather station's firmware.

SD card

For the micro-SD card connector (K5) we opted for a small breakout board available in the Elektor Store. It has a 100-mil pitch pinheader that's easily soldered and connected to the main PCB. This BoB has to be mounted perpendicular to the main board in order to access the card. You can either replace the straight pinheader on the BoB with an angled type or use pliers and judiciously applied force to bend the pins over 90 degrees. Be careful not to damage this small board though.

I²C bus & LCD

The on-board I²C bus is used for communication with the LCD (on K6) and the air pressure/temperature/humidity sensors (on K4), but it can also accommodate other I²C compatible sensors or devices you may want to add to the weather station. Pull-up resistors R5 and R6 are needed for this bus. However, if you connect an I²C device on K4 that already contains pull-ups on the bus connections, be sure to omit R5 and R6 or desolder/cut these parts from the PCB. The LCD module suggested for use here [2] does have on-board pull-up resistors tied to its 5 V supply voltage. FETs T3 and T4 act as bidirectional 3.3 V / 5 V level shifters to match the ESP32 I/O pins, which are not 5 V tolerant.

Not forgetting...

MOD3 accommodates an optional RFM95 breakout board (BoB) to cater for LoRa connectivity. This is a future extension and not supported yet in the current software release.

Connectors K7 and K8 interface to a Honeywell particle (dust) sensor. The sensor is subject to damage and malfunction caused by frost, so we didn't mount it in the thermo/hygro sensor case which requires installing in more open space.

The PCB designed for the main board is a perfect fit for the Spelsberg IP66-class case specified in the Component List. The lab prototype of the board is pictured in **Figure 2**. Note that the 3.81-mm (0.15-inch) PCB terminal blocks can be turned 180 degrees (i.e. with the connections facing inwards) to create more space in the housing for connecting the cables. An oddball solution indeed but totally workable.

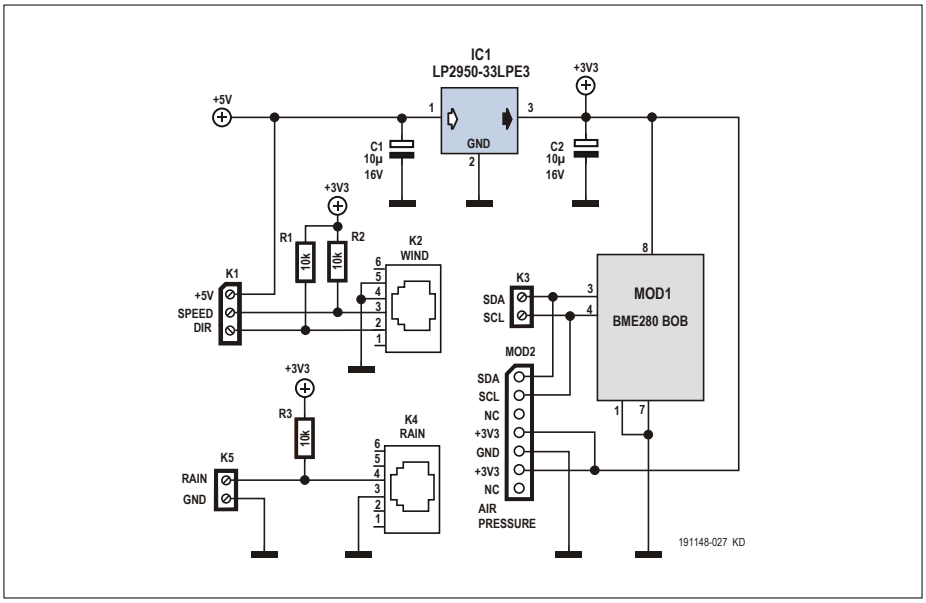


Figure 3: Schematic of the Connector Board for the rain and wind sensors. It also accommodates the pressure/humidity/temperature sensor module.

Connector board (no. 191148-2)

Although this board also accommodates the pressure/humidity/temperature sensor (module), its main function is to provide connectivity between the ESP32 microcontroller and the peripheral sensors. Refer to the circuit diagram in **Figure 3**. This 'circuitry' replaces the board inside the thermo/hygro unit that

COMPONENT LIST

Connector Board no. 191148-2

Resistors

R1,R2,R3 = 10kΩ, carbon film, 5%, 0.25W, 250V

Capacitors

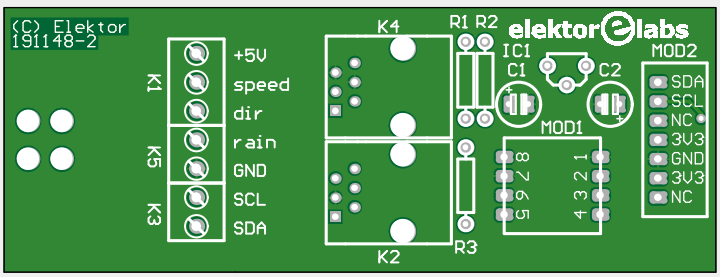
C1,C2 = 10µF, 16V, radial 5mm (6.5mm height)

Semiconductors

IC1 = LP2950-33LPE3, 3.3V LDO, TO-92

Miscellaneous

MOD1 = BME280 BoB from Elektor Store
MOD2 = WSEN-PADS BoB (Würth)
K1 = 3-way PCB screw terminal block, 0.15 in. pitch (3.81mm), 300V
K2,K4 = RJ-11 modular connector, 6P6C, vertical, Molex type 95503-2661
K3,K5 = 2-way PCB screw terminal block, 0.15 in. (3.81mm) pitch, 300V
PCB 191148-2 V1.1 from Elektor Store



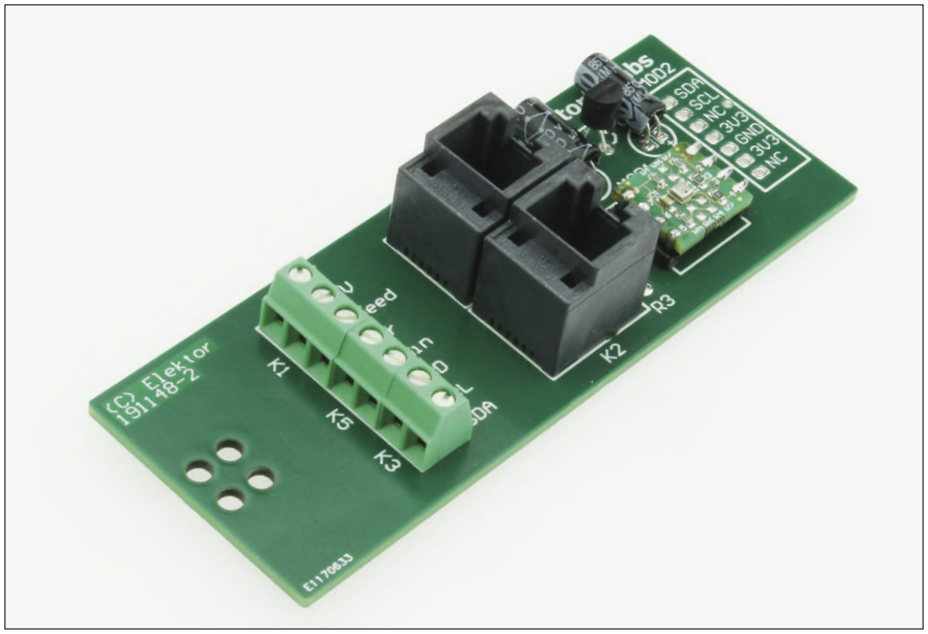


Figure 4: Finished Connector Board. Also on this PCB is the combined pressure/humidity/temperature sensor module.

comes with the WH-SP-WS02 outdoor kit, and slips perfectly inside the case.
The 3.3 V LDO (low dropout) regulator IC1 is the only active component on this PCB. Apart from the three pull-up resistors R1, R2, R3 for the switches in the rain and wind sensors, the other parts are limited to a bunch of connectors for the sensor modules used in the weather station.
The RJ-11 sockets connect to the wind sensor (K2) and the rain sensor (K4), their position corresponding to the holes in the 'evacuated' thermo/hygro unit and the text on the case.
Position MOD2 is for further sensors and user access to the I2C Bus in the cone.
Elektor's popular BME280 breakout board is again used as a 3-in-1 sensor module covering temperature, relative humidity and air pressure. The pin numbering on the PCB for position MOD1 matches that on the BME280 BoB. Should you prefer to apply a different brand of I²C compatible T/RH/P sensor module, you can connect it to position MOD2 and simply remove the BME280 BoB from the PCB.
Figure 4 shows the connector board as fabricated and tested by Elektor labs. Note that the four holes permit securing the connection cable to the board with two tie-wraps.

Dust sensor board (no. 191148-3)

The main purpose of this board is to provide easy connection and mounting of the Honeywell HPMA115S0-xxx dust/particulate matter sensor module. Refer to the schematic in **Figure 5**. The Honeywell sensor has an 8-way, 1.25-mm pitch (0.05 in.) connector that's troublesome to connect without professional tools. The sensor is better manageable if you use the Würth Elektronik connector described in the Component List. Just solder the four wires to K3 (two for 5 V power supply, and two for RxD and TxD) and carefully plug the 1.25-mm pitched connector into the sensor. Connectors K1 and K2 then provide an easily managed and user-friendly connection between the dust sensor and the main board.

4 SALE @ WWW.ELEKTOR.COM

→ **Main Board, bare PCB**
www.elektor.com/191148-1

→ **Connector Board, bare PCB**
www.elektor.com/191148-2

→ **Dust/Particle Sensor Board, bare PCB**
www.elektor.com/191148-3

→ **Weather Station Mk. II, kit of parts**
www.elektor.com/191148-71

→ **2×16 character LCD with I²C Backpack***
www.elektor.com/two-wire-lcd-071035-93

→ **Wireless Outdoor Unit for Professional Weather Station WH-SP-WS02**
www.elektor.com/professional-outdoor-weather-station-wh-sp-ws02

→ **Micro SD-Board BoB***
www.elektor.com/19251

→ **BME280 breakout board, I²C version (160109-91)**
www.elektor.com/bme280-mouser-intel-i2c-version-160109-91

→ **ESP32-PICO-Kit V4**
www.elektor.com/esp32-pico-kit-v4

* Stock/shipping delays due to production and export issues.

Web Links

[1] ESP32 Weather Station, Elektor Labs magazine edition 1/2019: www.elektormagazine.com/magazine/elektor-70/42351
[2] 2×16 character LCD with I²C: www.elektor.com/two-wire-lcd-071035-93
[3] Publication support page: www.elektormagazine.com/191148-01
[4] Project development page: www.elektormagazine.com/labs/remake-elektor-weather-station

Figure 6 shows the completed board with the sensor connected.
Use three M2 x 10 mm machine screws to attach the sensor to the PCB. Sure enough you'll need to drill a couple of ventilation holes in the case to yield sufficient airflow for correct measurements, yet prevent moisture to remain in the sensor once it is installed out in the open.

Products

The three unpopulated circuit boards for the Open-Network Weather Station Mk.2, as well as a kit of parts, are available from the Elektor Store — see the **4 Sale @ www.elektor.com textbox**. Elektor Members preferring to order their boards from a PCB manufacturer can do so using the Gerber files and BOM files available at [3].

Next time

In Elektor magazine edition 4/2020 covering the pleasant-weather months of July & August we continue with the software developed for the Open-Network Weather Station Mk.2. You can actively follow the progress on our Labs website [4]. ⏮

191148-01

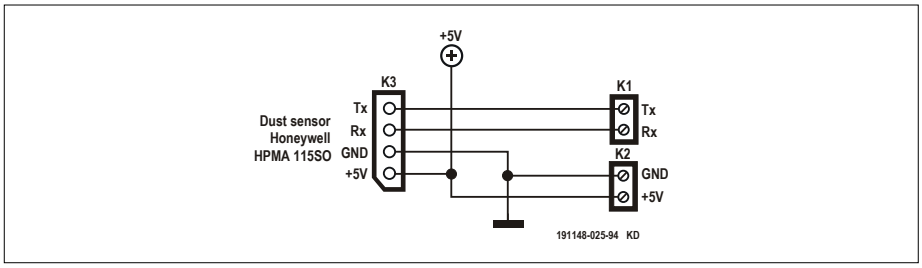


Figure 5: Schematic of the Dust/Particle Sensor board accommodating the Honeywell HPMA115 sensor.

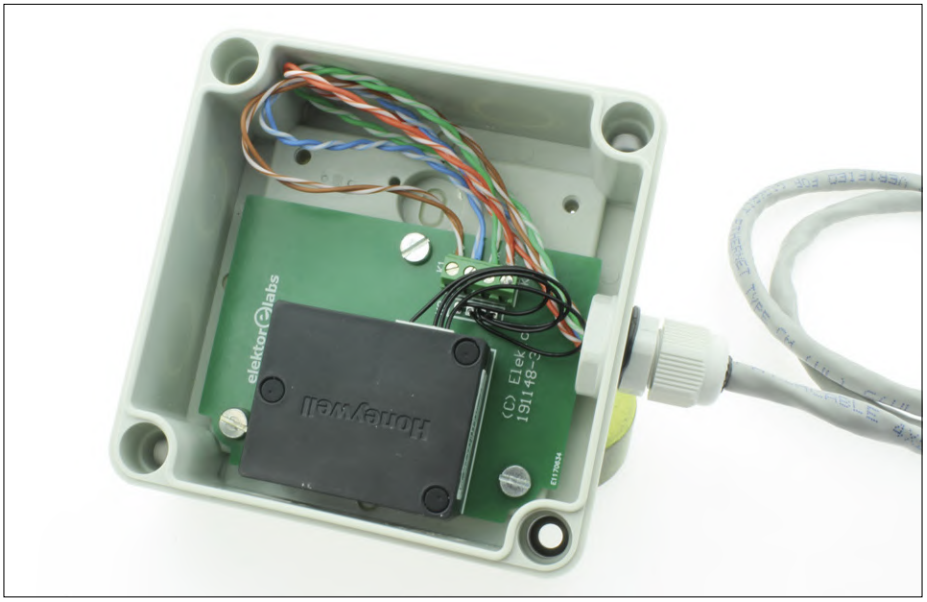


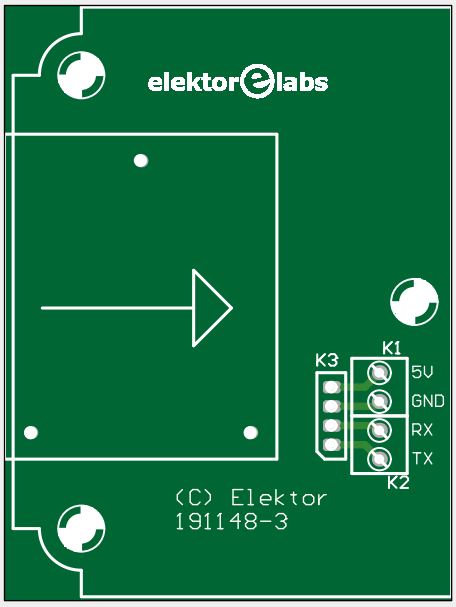
Figure 6: Ready assembled Dust/Particle Sensor Board mounted in the IP66 class case.



COMPONENT LIST

Dust/Particle Sensor Board no. 191148-3

MOD1 = Particulate Matter sensor type HPMA115S0-xxx (Honeywell), Farnell # 2770767
K1,K2 = 3-way PCB screw terminal block, 0.15 in. pitch (3.81mm), 300V
WR-PHD 8-way receptacle, 0.05 in (1.25mm) pitch, Würth Elektronik # 653008113322
2 pcs pre-crimped wires, Würth-Elektronik # 653153128130
3 pcs M2×10 machine screw
Case: TK PS 94×94×57 mm, IP66 class (Spelsberg)
PCB 191148-3 V1.0 from Elektor Store



Artificial Intelligence for Beginners (1)

Object recognition using the Maixduino board

By **Walter Trojan** (Germany)

There is no hotter topic in the world of IT than artificial intelligence (AI). A good low-cost introduction to the fields of speech and image recognition as well as other aspects of AI is the Maixduino board which, including the fitted camera and small LCD panel, comes in at only around €30/£25/$35; what's more, it can readily be programmed using the familiar Arduino IDE. In the first instalment of this series we will look at the comprehensive array of hardware offered by the Maixduino and at a couple of demonstration programs, including one that can recognize 1000 different objects.

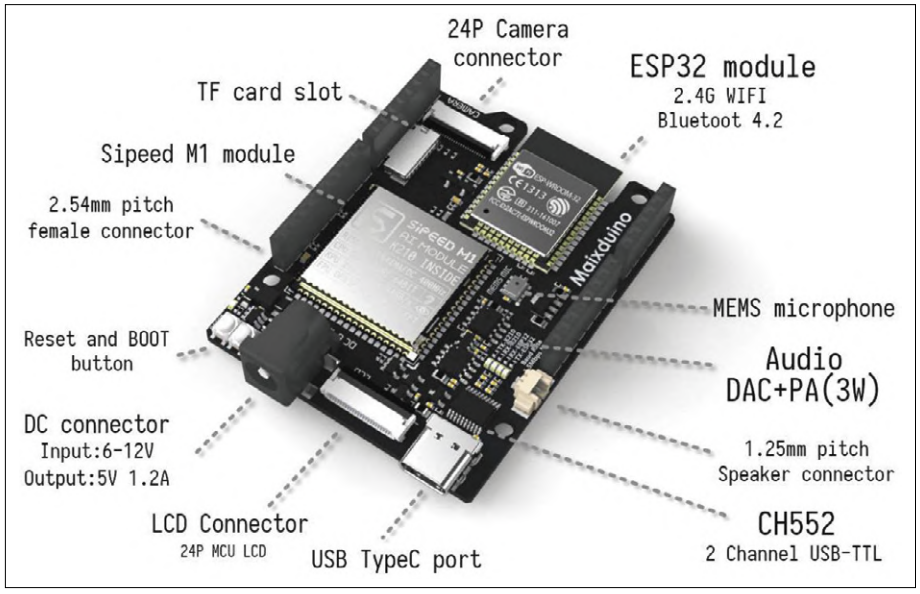


Figure 1: The hardware facilities offered by the Maixduino.

The use of artificial intelligence is on the increase in many areas, and it is finding its way into many types of device and application. This year it is estimated that worldwide business applications in the field of AI will generate revenues of some $4.8bn, and that figure is predicted to rise to over $30bn by 2025. Anyone with knowledge of the subject area can look forward to enhanced employment prospects as expertise in AI is urgently sought after. It is therefore worthwhile to take a little time to get to grips with the topic and the new worlds of possible applications that it opens up. And last but not least, it's a lot of fun!

The term 'artificial intelligence' is generally considered to cover the creation of systems exhibiting intelligent behaviour. One part of this is 'machine learning', whereby instead of having an application function specified in all its detail by the developer, a general-purpose programming framework in the form of a neural network (NN) is used, which learns the required function by itself using a large corpus of training data. One class of machine learning is called 'deep learning', where more complex and optimized program structures are used to improve results further. We will be looking at these ideas in more detail later in this short series of articles.

However, AI is not a new subject: its beginnings can be traced back to the 1950s. The breakthrough that occurred a few years ago was the result of considerable increases in the amount of computational power available. Graphics cards with hundreds of processor cores operating in parallel and highly-specialized AI chips enabled AI systems to be implemented and trained. That opened the door to automatic speech recognition, as used in many personal assistants, and image recognition of objects of all kinds. AI also plays an important role in autonomous vehicles. In some areas AI has already surpassed human abilities: AI systems can beat world champions in the games of chess and go, and can detect tumours more reliably than human medical specialists. But do not worry: the systems are only superior in the particular areas in which they have been trained, and the areas in which the systems are trained will continue to be chosen by humans.

A low-cost introduction

A good low-cost introduction to this world is the Maixduino, which in the form of the 'MAix BiT' kit including the board, camera and small LCD panel can be obtained from Elektor: see the text box. The board

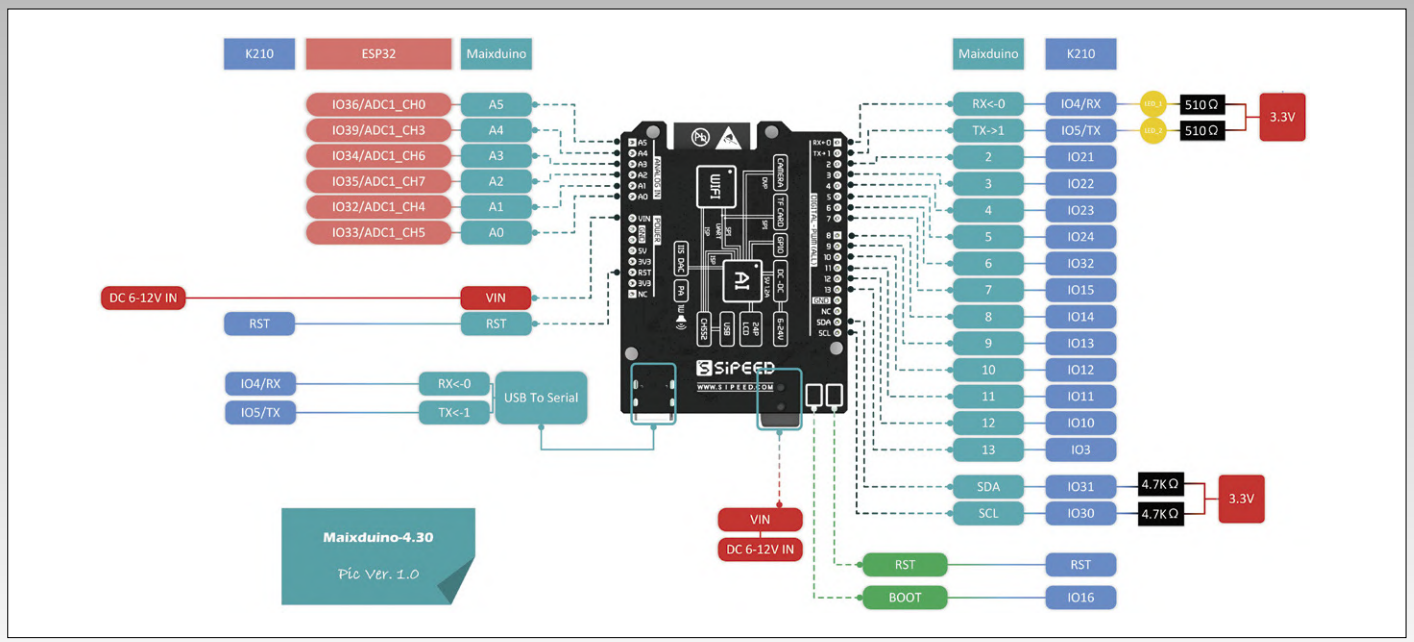


Figure 2: Maixduino pinout.

has the same format as an Arduino Uno (from which it gets its name), but the hardware it includes is considerably more comprehensive. The board is manufactured by the Chinese company Sipeed. Alternatives to the Maixduino include the Nvidia Jetson Nano, the ROCK PI N10 Model A, the Intel Neural Compute Stick 2 and others, but these all cost considerably more.

In this first instalment of this series we will look at the rich hardware environment offered by the Maixduino and how it can be programmed with the help of the Arduino IDE. As well as a couple of typical Arduino-esque applications we will look at how to use the camera and the screen. And the final highlight will be a demonstration of how to use the board to recognize objects.

In the second and third instalments we will delve deeper into the topic of artificial intelligence, describe the structure of neural networks, install MicroPython and its accompanying IDE and demonstrate how facial recognition works. We will also look at how you can program your own AI applications and how to communicate with the Internet.

A deep dive into the 'super Arduino'

The specifications of the Maixduino will have hardware enthusiasts licking their lips! The overall size and construction of the board is broadly aligned with that of the Arduino Uno, but you will immediately notice the higher component density and that many additional connections are available. At the heart of the board there are two large modules. The first is the Sipeed M1 AI processor module based on the Kendryte K210 device, whose interior workings we will look at later, and the second is an ESP32 module for communicating over WLAN and Bluetooth and for acquiring analogue signals. The ESP32 contains two processor cores clocked at 240 MHz and so by itself offers a considerable amount of processing power, and it can be used to offload communication functions from the main processors. The ESP32 has already featured in many articles in this magazine, and so we will not go into further detail about it here.

The header connectors correspond in number and arrangement to the original Arduino, and the pin assignments are mostly identical. But beware that the inputs and outputs are designed for 3.3 V or even 1.8 V operation, and **the input circuitry will be destroyed if 5 V is applied**.

Further details are shown in **Figure 1**. Two 24-pin sockets are provided for interfacing to the camera and to the LCD panel. A slot is provided for a microSD card which can be used to make a large amount of storage available to the Maixduino. The USB connector is the modern type-C kind and is used for programming and monitoring. And, to allow the processing of audio data, the board includes a digital microphone and an audio amplifier with a 3 W output driven by a digital-to-analogue converter. So there is plenty of hardware on the board to let you get started on a range of possible projects without having to add extra boards.

The pinout of the board (see **Figure 2**) closely resembles the original. The supply voltage can be provided over the DC barrel jack or using the VIN pin at 6 V to 12 V; alternatively, 5 V can be supplied over the USB connector. Six of the ESP32's GPIOs are brought out; these are the ones that can alternatively be used as analogue inputs A0 to A5. On the opposite edge of the board are the inputs and outputs of the K210 module. These can be controlled using the usual Arduino commands, with the 'Arduino pin number' being the number of the input or output bit of the K210.

```
#define LED 12
   // K210 IO12, Maixduino pin 10
pinMode(LED, OUTPUT);
   // configure port as output
digitalWrite(LED, HIGH);
   // port to high (3.3 V)
```

The RST pin is designed to operate at 1.8 V and external circuitry should not apply a higher voltage than that to it. Outputs IO36 to IO47 are also designed for low-voltage operation: these are not brought out to the headers, but are used internally, for example to drive the LCD panel. The serial RX and TX ports and the

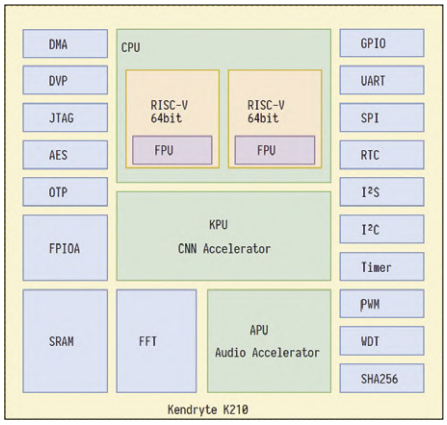


Figure 3: Block diagram of the Kendryte K210.

I²C interface are provided with suitable pull-up resistors fitted on the board. Unfortunately the K210 datasheet does not specify the maximum output current available at the outputs. However, they should be capable of driving an LED at up to say 10 mA or so; for higher currents a driver IC should be added.

The big enchilada: Kendryte K210

The K210 SoC (system on a chip) made by the Chinese company Kendryte is at the heart of the Maixduino. It is made using a low-power silicon technology on a 28 nm process and has been available on the market since September 2018. For carrying out 'normal' work it offers two 64-bit processor cores that are clocked at 400 MHz, with overclocking up to 800 MHz possible. These cores are based on the RISC-V specification: this saves the manufacturer from having to pay licensing fees to Arm and so helps to contribute to the low total cost of the device. Both processors come with an FPU (floating-point unit) that operates in both single and double precision. **Figure 3** shows in more detail what is inside this chip.

The special feature of the K210, however, is its KPU (knowledge processing unit) that can be used to construct and execute neural networks. The total compute power available is astonishing at this price point: 0.46 Tops, or 460 billion operations per second. With overclocking this figure can even be doubled, allowing, for example, up to 60 objects per second to be recognized. The high processing performance is achieved using 64 arithmetic units operating in parallel and a bus width of 576 bits. Also, the power dissipation of just 0.3 W is very low in comparison to other AI systems. For example, Nvidia recommends using a 5 V 4 A (hence 20 W) power supply for its Jetson Nano, which at 0.4 Tops offers comparable computing power.

The KPU can implement advanced neural network architectures, including convolutional networks. These have a particularly efficient filter structure that makes them especially well suited to image processing applications: more on this in the second instalment of this series. The main memory included in the SoC has a capacity of 8 MB, divided into 2 MB for the main processors and 6 MB for the KPU. That means that up to 5.9 MB is available for storing the neural network configuration, which is enough to implement a medium-sized network.

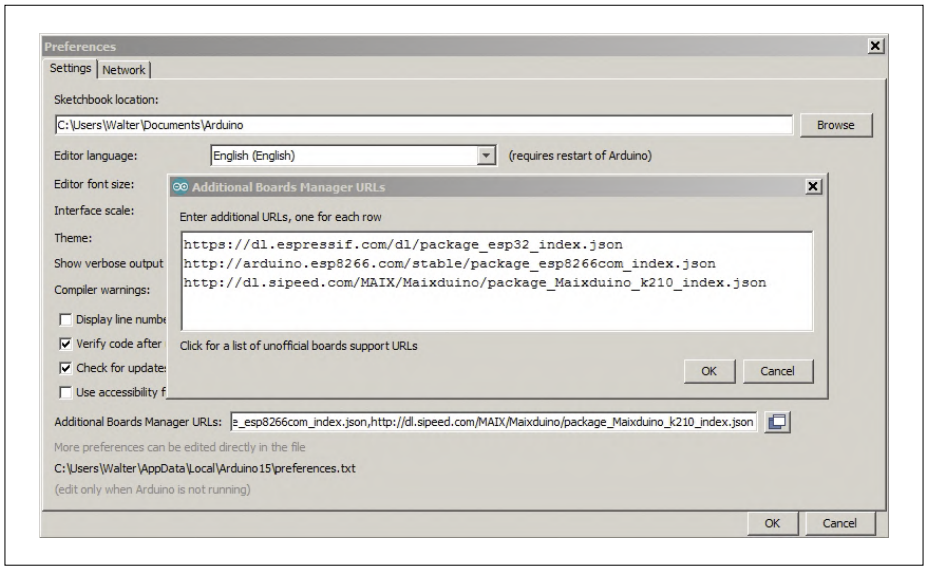


Figure 4: Adding the Maixduino core configuration.

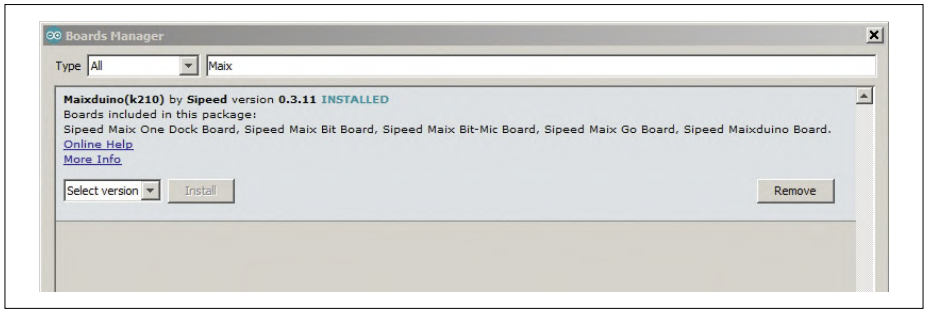


Figure 5: Installation of the Maixduino core tools.

However, the K210 has more hardware tricks up its sleeve. These include an audio processing unit (APU) which is particularly helpful for pre-processing in speech recognition applications. The unit can handle up to 8 channels (or 4 stereo channels) at input sample rates of up to 192 kHz. An FFT (fast Fourier transform) unit is available to carry out analysis of the frequency spectrum of a signal.

Also noteworthy are the AES and SHA-256 accelerators provided to speed up cryptographic functions.

And of course the device includes the usual complement of peripherals including UART, I²C, SPI, I²S, timer, RTC and PWM.

Sipeed has added 16 MB of flash memory on the board alongside power supply circuitry, the microphone and 3 W power amplifier, and the excellent ESP32 module.

What more could you wish for? More information on the operation of the board, the circuit diagram and other details can be found by following links [1], [2] and [3].

Software development

It is not just the shape of the board that is similar to the original Arduino:

the board uses the same Arduino IDE, with the Maixduino core being integrated into the environment in the same way as the ESP8266 or ESP32. Under File -> Preferences it is necessary to add a new board manager URL: if you right-click on the button to the right of the text entry box a small window will open to make entering the URL easier (see **Figure 4**). If you only want to program the Maixduino, then only the 'sipeed' line needs to be added; then the on-board ESP32 will also be programmed using it.
To install the Maixduino board tools select the menu item Tools -> Board -> Boards Manager. Enter the search term 'Maix' (as shown in **Figure 5**) and then proceed to install the tools.
Now we can start to do some programming. Rather than the conventional 'hello world' program, we will jump straight into to testing the camera and screen. Connect the Maixduino over USB and in the Tools menu set the following parameters.

Board: Sipeed Maixduino
CPU Clock Frequency: 400 MHz
Burn Tool Firmware: open-ec
Burn Baud Rate: 1.5 Mbps
Tool Install Location: Standard
Port: <COM port that you are using>
Programmer: k-flash

A demonstration program that captures an image and displays it on the LCD is already available within the IDE. Call it up using File Examples Sipeed_OV2640 selfie and once it is uploaded to the board (you may need to press the reset button at this point) you can run it. The code is shown in **Figure 6**.
The program starts by bringing in the functions from the Sipeed libraries for driving the camera and LCD, which are connected to the board using an SPI bus. The image format chosen is QVGA resolution (320 by 240 pixels) with RGB565 colour. The setup routine initializes both devices, and in the main infinite loop the captured images are transferred directly to the LCD: it could hardly be simpler.
The image is not particularly contrasty, but it is sharp and updates are smooth. **Figure 7** shows the results.
As you can see, the Maixduino is as easy to use as an Arduino Uno. However it offers many more possibilities and we can immediately start to build more sophisticated applications.

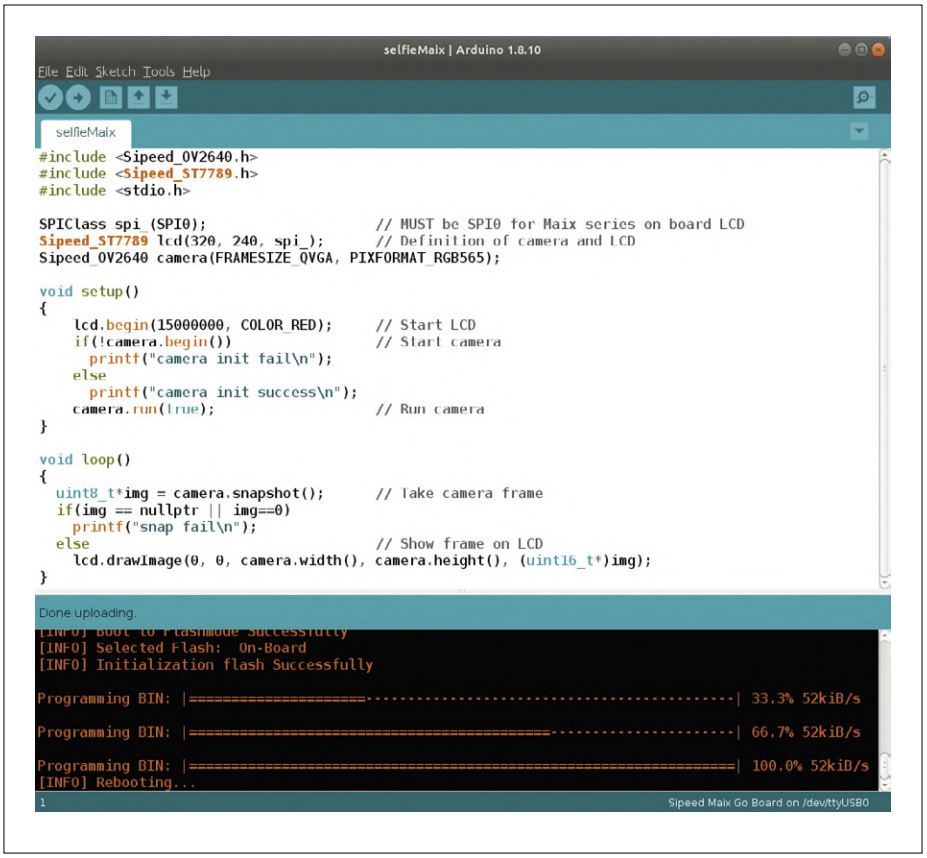


Figure 6: Arduino IDE with the 'selfie' program.

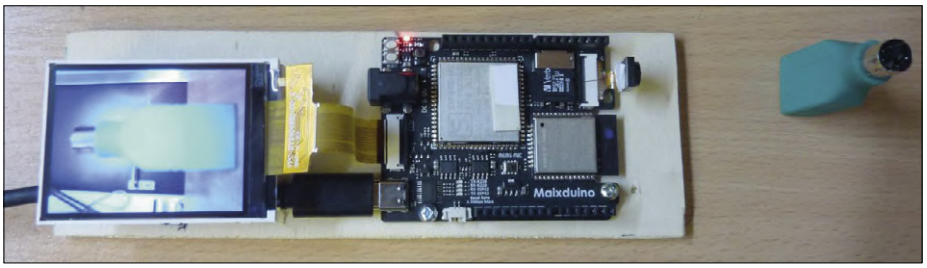


Figure 7: Captured image of an adaptor plug.

Our first AI model

We will be looking in more detail at deep learning aspects of artificial intelligence in the second article in this series, but here we can demonstrate a simple application. We will be using 'MobileNet': nothing to do with mobile phones, but rather an image classifier that can recognize and identify 1000 types of everyday objects. This uses a neural network, which is a software structure built from nodes organized in layers trained using a lengthy process in which it is presented with thousands of images.
Since the file containing the training image set runs to some 200 GB and the training process itself is very time-consuming, it is not really practical to carry it out on the Maixduino. However, it is possible to install a ready-trained model and immediately set it to work recognizing images. The steps to achieve this are as follows.

The required software can be found at link [4]. It consists of the following components.

- `mobilnet_v1.ino`: main C++ program for running the demonstration
- `MBNet_1000.h`: header file for the demonstration routines
- `MBNet_1000.cpp`: C++ routines for

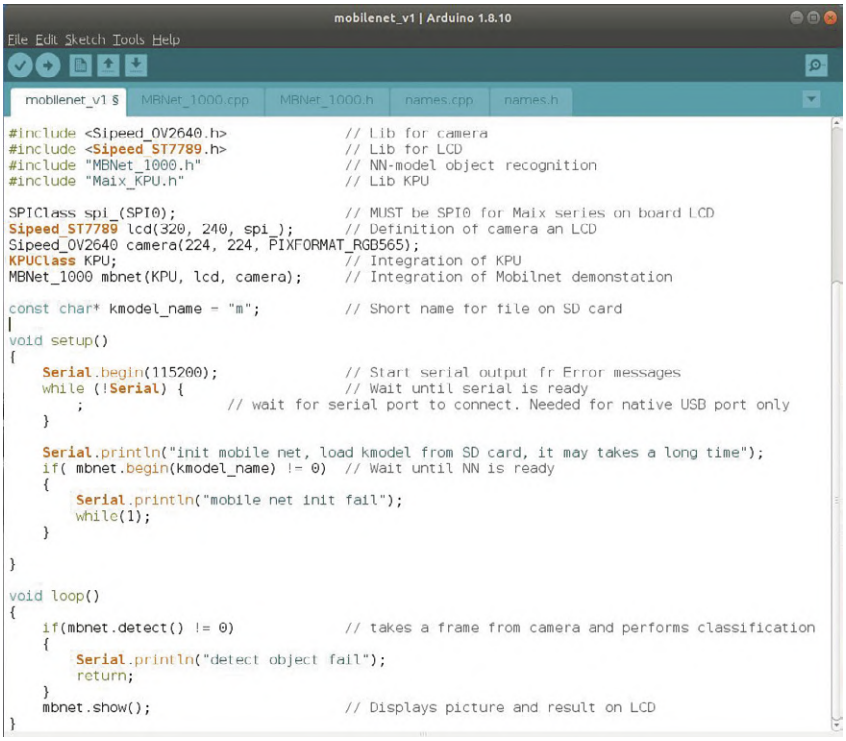


Figure 8: MobileNet demonstration program.

acquiring, recognizing and displaying objects

- `names.h`: header file with object descriptions
- `names.cpp`: C++ routine for describing the recognized objects

Download these files and place them all in the same directory. The pre-trained AI model mobilenet_0x300000.kfpkg can be downloaded from link [5]: compressed, it is just a few megabytes long. After decompressing it with 7zip you should find a folder mobilenet_0x300000 in which there are two files: the model is the one called 'm'. Copy this file onto a microSD card in the root (top-level) directory and insert the card into the slot on the Maixduino.

A glance at the program code in **Figure 8** shows that it is possible to program even very complex applications with little effort. That is of course all down to the powerful libraries: in the field of AI there are very many highly efficient libraries available. The first part of the program declares the camera, LCD and the KPU along with their parameters. The camera resolution is configured to the same format as the training images, 224 by 224 pixels. Finally, the object `mbnet` marshals together the KPU, camera and LCD resources.

The setup routine initializes the demonstration code, and then in the main loop we have the classification of the objects in the images acquired by the camera and their display on the LCD panel. Again, it is hard to imagine that an object recognition program could be any simpler.

As a test we can put a photograph of a cat in front of the camera (as shown in **Figure 9**). The system immediately recognizes it as a tabby or Egyptian cat (**Figure 10**). The application is capable of classifying about five images per second and is therefore capable of producing useful results even if the camera is moving. The most important things are to have good lighting and not too busy a background to the image.

As can be seen from Figure 10 the acquired image uses only a blue and white colour palette, in the interests of increasing the recognition rate. This is a commonly-used method: instead of using all three colour channels in an image we use only one to reduce the amount of pixel data and hence the amount of processing power required. The technique of using a ready-trained network, or in other words of carrying out training and classification on different platforms, is also widespread in this type of application. Many users take advantage of the large amounts of processing power available on AWS, Microsoft Azure

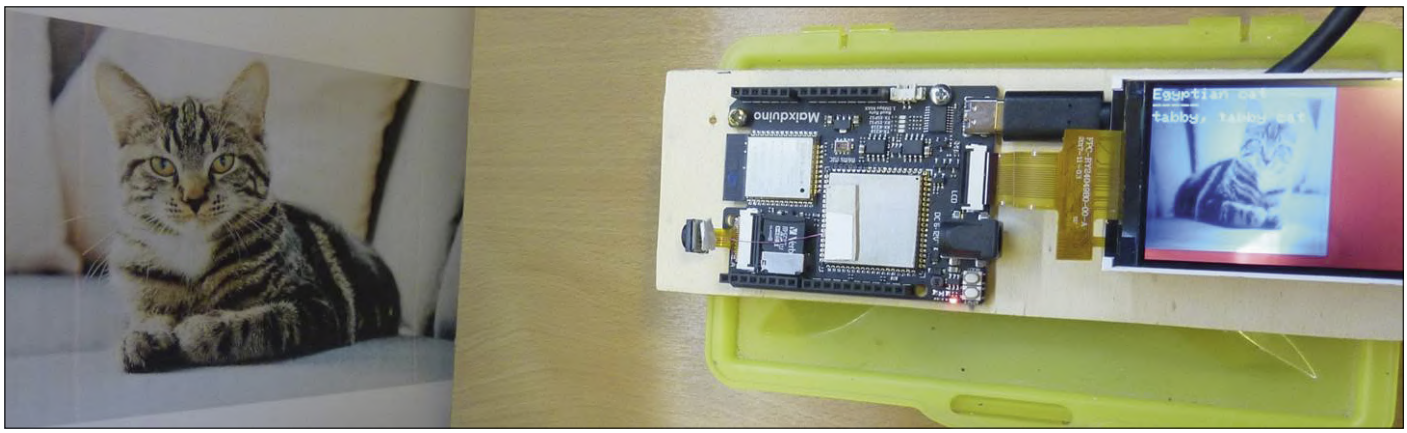


Figure 9: Acquisition of an image of an object.

or Google Cloud to train their neural networks, while running them (which requires considerably less in the way of resources) on smaller-scale platforms. The semiconductor industry is already responding to this demand: Intel recently announced their Nervana NNP-T and NNP-I neural network processors, where the 'T' version offers higher processing power for training, and the 'I' version is a lower-performance device aimed at inference and classification applications.

Coming up...

The demonstration application we have looked at in this article can only scratch the surface of the topic of artificial intelligence, but with luck your interest in the possibilities of machines that learn has been piqued.

In the next instalment of this series we will look at the structure and function of a neural network in more detail. That will come with a new development environment: we will be introduced to the temptations of Linux and the Python programming language, as that is where the most powerful libraries and frameworks are available. These frameworks make the creation of a neural network as easy as plugging together Lego bricks. We will also look at a face recognition application and at how to develop your own neural network structures. If you are too impatient or curious to wait for that, the book *Make Your Own Neural Network* by Tariq Rashid is recommended. ⏮

200023-02

4 SALE @ ELEKTOR.COM

→ Sipeed MAix BiT Kit for RISC-V AI+IoT
www.elektor.com/sipeed-maix-bit-kit-for-risc-v-ai-iot



Figure 10: The object is identified and classified.

Web Links

[1] Maixduino description: https://wiki.sipeed.com/en/maix/board/maixduino.html

[2] Sipeed forum: https://bbs.sipeed.com/

[3] Maixduino circuit diagram: http://dl.sipeed.com/MAIX/HDK/Maixduino/Maixduino-4.30/

[4] MobileNet demonstration: https://github.com/sipeed/Maixduino/tree/master/libraries/Maix_KPU/examples/mobilenet_v1

[5] MobileNet model data file: http://dl.sipeed.com/MAIX/MaixPy/model/mobilenet_0x300000.kfpkg

Advertisement

HAMMOND MANUFACTURING®

Hand held enclosures standard and waterproof

More than 5000 different enclosure styles:
hammfg.com/electronics/small-case

01256 812812
sales@hammond-electronics.co.uk



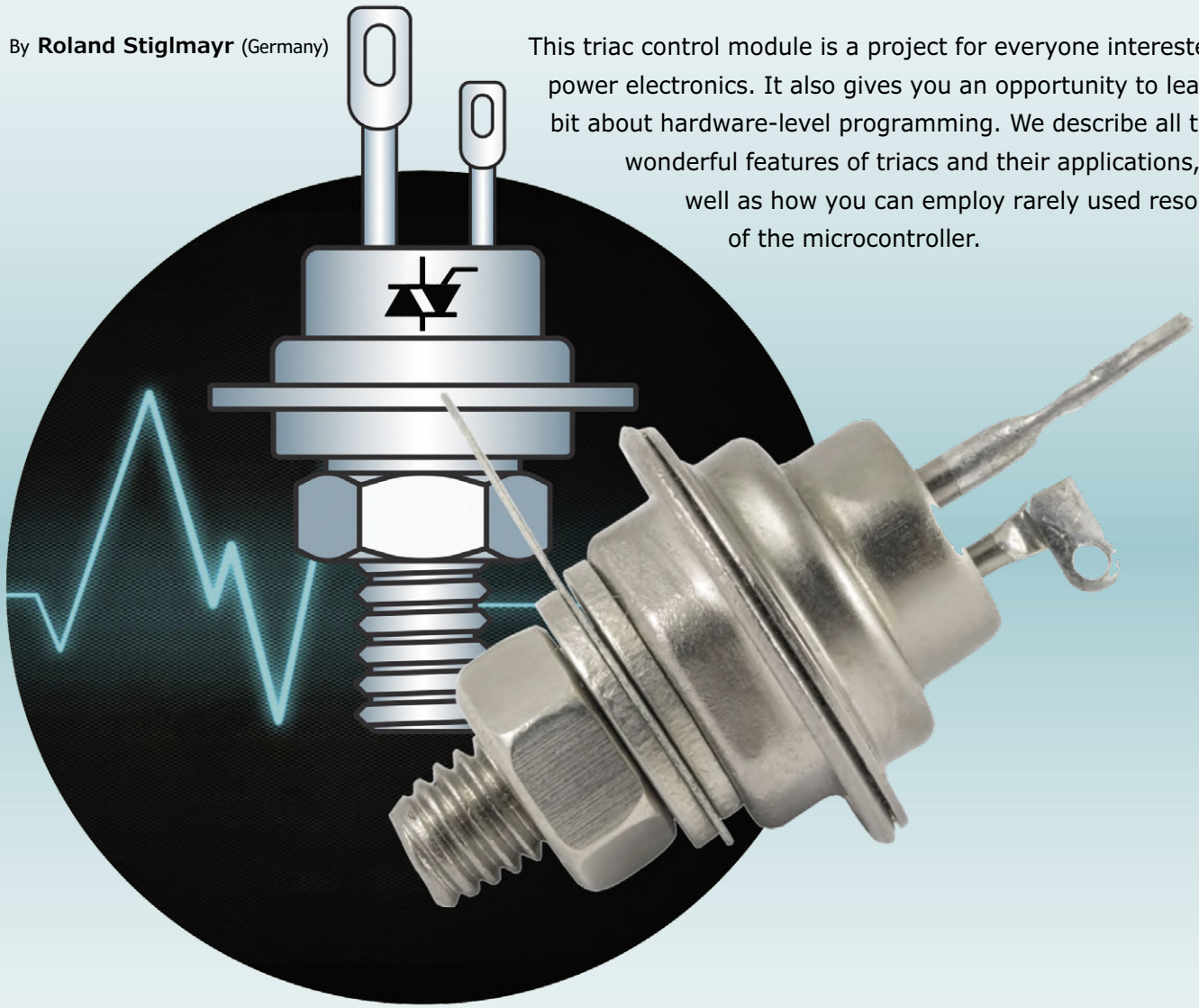


Universal Triac Control Module with ATmega

Switch and dim a variety of loads

By **Roland Stiglmayr** (Germany)

This triac control module is a project for everyone interested in power electronics. It also gives you an opportunity to learn a bit about hardware-level programming. We describe all the wonderful features of triacs and their applications, as well as how you can employ rarely used resources of the microcontroller.



Everyone involved in electronics knows more or less what thyristors and triacs are, how they work and how they are used. However, these components offer so many possibilities that only specialists are able to fully exploit their potential. Nowadays thyristors are mainly used in special applications with high voltages (above 1 kV) and strong currents (many kiloampères), so this article only focuses on triacs.

First we look at the basic operation of triacs and the relevant parameters. Then we describe the circuitry of the control module. **Figure 1** shows the author's finished prototype. Many functions are possible with this combination of an ATmega microcontroller and a triac. Depending on the firmware, the module can be used as a simple switch or remote switch, a timer, a dimmer, or a soft-start module for switching inductive loads such as transformers or motors.

Basics

A triac is a semiconductor switch that conducts current in both directions when it is switched on, which means it is an AC

switch. The load current flows through the two electrodes T1 and T2 (here 'T' stands for 'terminal'). These electrodes are also designated MT1 and MT2 (for 'main terminal'), and in the past they were also called A1 and A2 (for 'anode'). The third electrode is the gate (G), which is the control lead. **Figure 2** shows the Elektor circuit diagram symbol for a triac (on the left) and a simple equivalent circuit (on the right).

A triac remains in the off state until the current flowing from the gate (G) to T1 is high enough to switch it on. Once the triac is switched on, it remains in the on state until the current between T1 and T2 drops below a threshold value, which depends on the type. Then it returns to the off state. In operation with an AC voltage, this means it switches off at every zero crossing of the current, which is not necessarily in phase with the voltage, unless it is constantly kept on by the gate current. Accordingly, a triac can be triggered by a short, low-energy pulse on the gate, but it cannot be directly forced to switch off.

T1 is the reference point (reference voltage) for the gate G. There are four different directions of current flow, depending on the polarity of the voltages on the gate and T2 relative to T1. These are shown by the four-quadrant diagram in **Figure 3**. The X axis represents the polarity of the voltage on the gate relative to T1. The Y axis represents the polarity of the voltage on T2 relative to T1. The resulting four quadrants are labelled with Roman numbers in Figure 3. Positive values are shown in red, and negative values in blue. On the data sheets, the triac parameters are stated for all four quadrants because they are often different.

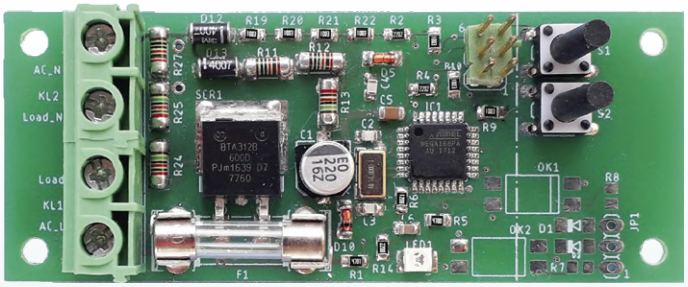


Figure 1: The author's prototype of the control module.

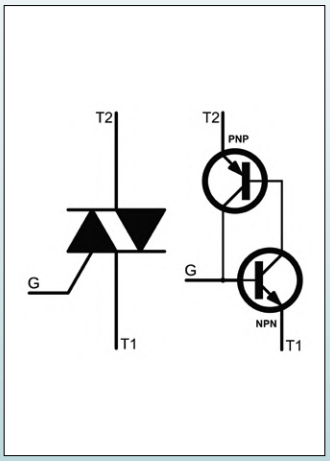


Figure 2: Elektor schematic symbol for a triac (left) and an equivalent circuit for the first quadrant, composed of PNP and NPN transistors (right).

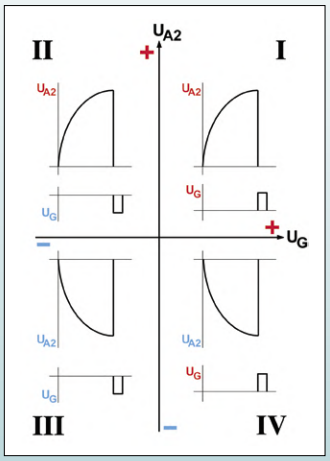


Figure 3: The four quadrants result from the polarities of the voltages on the gate and T2 relative to T1.

Parameters

To trigger a triac into conduction, the gate current must be higher than the threshold value, which is designated as the gate trigger current I_{GT}. At the end of the control pulse, the triac only stays on if the resulting load current is higher than the latching current I_L. Otherwise the triac switches off after the end of the control pulse. The trigger currents and other currents are specified for each quadrant separately on the data sheets. If the load current through a switched-on triac drops below the holding current I_H, the triac switches off again.

With an inductive load there is phase shift between the current and the voltage, with the amount of the shift depending on the inductive component of the load. Consequently, the voltage on the load is not necessarily zero at the zero crossing of the current. As a result, the voltage on T2 can jump to a high value when the triac switches off. This can cause problems. If the voltage rate of rise is higher than the limit value dV_{COM}/dt, the triac will be switched on again. Depending on the load, this can be damaging. An excessive rate of rise can be prevented by connecting a RC network called a snubber across T1 and T2. There are also 'Hi-Com' triacs available that are significantly less sensitive in this regard.

The dV_D/dt parameter is also relevant. If the limit value of this parameter is exceeded on a triac in the off state, it will also switch on. If the resulting current flows for exactly one half-wave, a transformer switched on this way can enter saturation and suddenly present a low impedance. In many cases the triac, and possibly the tracks on the PCB, will not survive such an incident. Here again, a snubber or a Hi-Com triac can remedy the situation.

The parameter dI_T/dt, which is the limit value for the maximum current rate of rise after triggering the triac, is also critical. If it is exceeded, the triac will fail with a short between T2 and T1. This results from non-uniform current density around the gate electrode, leading to local overheating. As the value for dI_T/dt is especially low in the fourth quadrant, this operating mode should be avoided. The dI_T/dt issue is relatively minor if switching occurs at the voltage zero crossing, although this is only true for predominantly resistive loads. If necessary, the rate of rise of the current can also be limited by connecting a small air-core choke ahead of the triac.

The parameter dI_{COM}/dt defines the limit value for the rate of rise of the current at the zero crossing of the current. If this value is exceeded, the triac will not switch off but instead remains on. The triac can also be triggered falsely if the repetitive peak off-state voltage $\mathbf{V}_{DRM}$ is exceeded. This can be remedied by connecting a varistor between T2 and T1 to limit the maximum voltage.

The last parameter that needs to be explained is the overcurrent protection factor I^2t, which is the integral of the square of the current over a 10 ms period. This limit corresponds to the maximum energy absorption of the triac. It must always be taken into account in the dimensioning of the fuse.

All other quantities listed on the triac data sheet are largely self-explanatory. If you want to design your own triac circuits, you should now be able to use these parameters to select the best triac for your application.

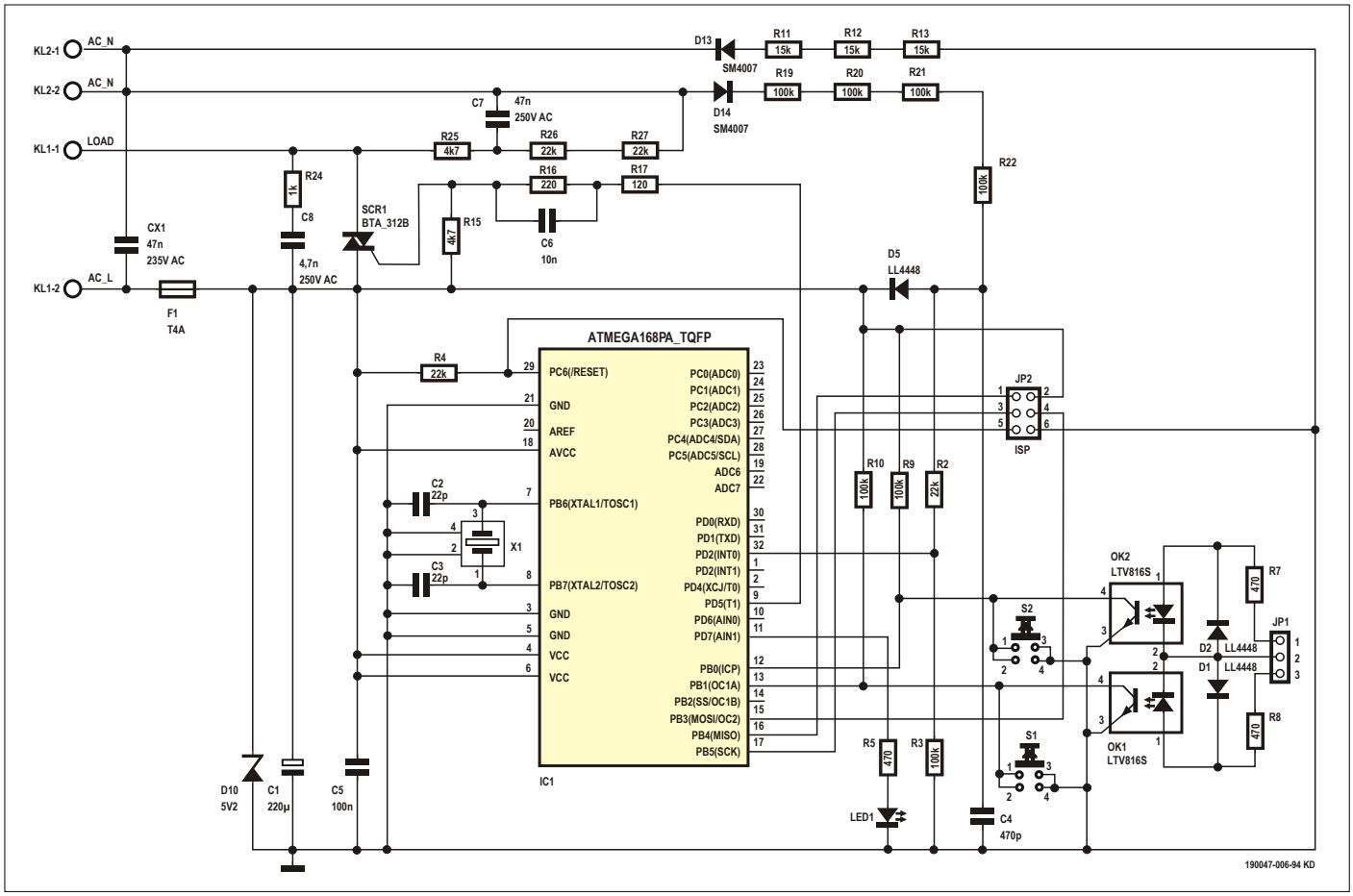


Figure 4: The circuit diagram of the control module. Crystal X1 has a frequency of 11.0592 MHz.

Triac control module

Now let's get down to practice. Thanks to its intelligence, the universal triac control module can act as a multi-purpose simple switch, a timer with delayed switch-off or a dimmer, or be used to control inductive loads. That's a lot of versatility from a single module. If you understood 'intelligence' to mean 'microcontroller', you got it right. Here we chose an ATmega device.
With regard to triac selection, the main criteria are the current to be switched and the maximum voltage on the triac. With operation from a single-phase AC mains, around 600 V is sufficient (230 V × √2 plus a safety margin). For industrial applications it is better to use triacs with 800 V reverse voltage rating, due to the high transient voltages in industrial environments. The current actually possible in practice depends not only on the rating of the triac, but also on the manageable power dissipation, and thus on the cooling. The power dissipation is equal to the product of the forward voltage and the switched current.
The minimum load current in the application must also be considered. Both I_L and I_H must be clearly exceeded, as otherwise the load will not be switched on cleanly or the current will not flow continuously. The 600 V devices BT134-600E (4 A) and BTA312B (12 A) are good general-purpose devices that are suitable for many simple purposes.

Powering the module

Supplying power to the module is not trivial. A small switching power supply would be nice, but would need a lot of space. A series capacitor would be simpler, but MKT capacitors with the necessary capacitance and voltage rating are not exactly cheap. A series resistor is even simpler. We want the triac to operate in quadrants II and III. This requires a negative gate trigger voltage relative to the T1 electrode, which is connected directly to the AC mains. This is most easily achieved with a half-wave rectifier. However, the feasibility of this is directly dependent on the current consumption of the circuit.
The MCU and the trigger energy required by the triac are the main considerations in this regard. The MCU does not have much to do and can therefore spend most of its time in low-power sleep mode. The average trigger energy is low because only short trigger pulses are needed. The question is how low we can make the current consumption. The MCU is in Extended Standby Sleep Mode most of the time and is only awakened to generate the trigger pulses.

The worst case is high-frequency triggering without zero crossing detection, with a trigger signal generated every 278 µs. Then the ATmega needs 1.9 mA in practice, with a V_{CC} of 4.7 V. With a peak gate current of 11 mA for each 10 µs pulse and a period of 278 µs (3.6 kHz), the average gate current is 400 µA. The total current is therefore approximately 2.3 mA. This determines the value of the series resistor and its power dissipation. With half-wave rectification, the effective value is $I_{pk}/2$ and the average value is I_{pk}/π. This yields:

I_{pk} = 2.3 mA × π = 7.3 mA

R = 325 V / 7.3 mA = 45 kΩ

$I_{eff} = I_{pk} / 2$ = 3.7 mA

The power dissipation of the resistor R is:

$P_{dis} = I_{eff}^2 \times R = (3.7\ mA)^2 \times 45\ k\Omega = 0.62\ W$

If R is divided into three individual 1 W resistors, then the power dissipation and the applied voltage do not present any problem. There is also an issue with switch-on. As long as the MCU is in reset mode, it draws a relative large current and V_{CC} rises only very slowly. This can be solved by setting the BOD level to 2.7 V and immediately activating sleep mode for 250 ms. This is explained later on.

Safety

You should never ignore the danger of a module operating directly from the AC mains. With AC mains operation, you shouldn't connect any measuring instruments, and definitely not a PC. Things like this should be left to qualified electricians, and isolation transformers and other equipment are mandatory. Be sure the module is protected against touch contact when it is connected to the AC mains. It should always be fitted in an insulating case.

For tests or modifications, the module must be completely separated from the AC mains and powered from a 5 V supply. A 50 Hz signal with an amplitude of 4 V from a square-wave generator, injected at C4, can be used as a test signal for zero crossing detection.

Hardware

In the circuit diagram shown in **Figure 4**, power for the MCU and the peripherals is provided by the series resistors R11-R13, half-wave rectification with D13, and filtering with C1. The MELF resistors used here for R11-R13 have a high breakdown voltage and power rating. Due to the high impedance of the power supply, it effectively approximates a current source, allowing the Zener diode D10 to easily provide a constant voltage.
The MCU clock is controlled by crystal X1 to ensure precise timing of the trigger pulses. The network formed by diode D12 and the voltage divider R2, R3 and R19-R22 generates a signal suitable for zero crossing detection, which is clipped to V_{CC} by diode D5.
The gate current is limited by R16 and R17. Fast triggering of the triac is provided by C6 in parallel with R16. A snubber formed by R24 and C8 is connected across T2 and T1 of the triac to prevent false triggering. The triac used here is a Hi-Com type, so the combination of a small capacitance and relatively high resistance is sufficient.

The optional auxiliary load, consisting of R25-R27 and C7, is a special feature. It is only required if the external load has high inductance, such as a transformer. These components are usually not fitted. Capacitor CX1 suppresses RFI.
The module is operated using the pushbutton switches S1 and S2. This can also be done with optocouplers OK1 and OK2 connected in parallel with the switches. Switches with long actuator pins should be used to ensure adequate safety distances. LED1 acts as a power indicator. The microcontroller

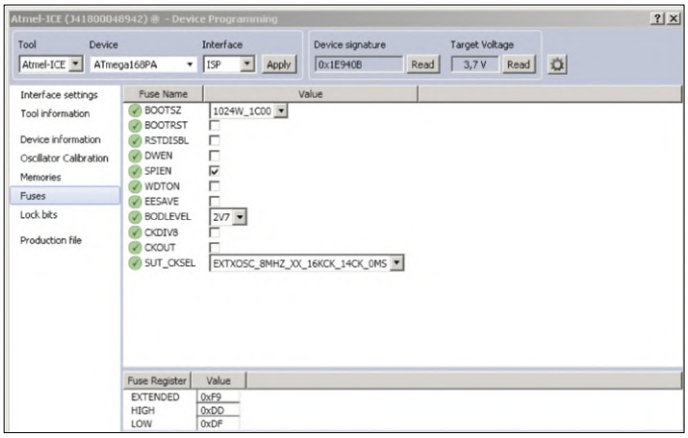


Figure 5: How to set the fuses when you program the ATmega the first time.

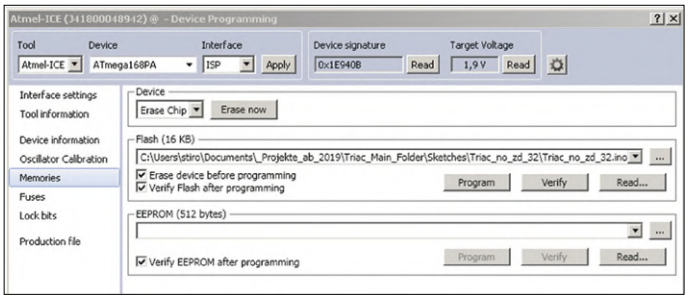


Figure 6: Settings for flashing the firmware with an ISP programmer.

(IC1) is programmed through the ISP interface on JP2. Fuse F1 at the AC mains input protects the circuit in the event of a short circuit.

Software

The project software [1] was written using the Arduino IDE. However, the author did not use the Arduino functions because they are too slow. The program created by the author is very fast and operates in real time. The sketches were there therefore intentionally not ported to Atmel Studio.
Unfortunately, it is not possible to use the boot loader to program the microcontroller. So much current would be needed at switch-on that the supply voltage would not rise. This means that the fuses must be set to disable the boot loader. The microcontroller can be programmed directly from Atmel Studio using a suitable ISP programmer.
For each app, the project folder contains the corresponding sketch and the hex file. In the initial programming, it is essential to program the fuses as shown in **Figure 5**. Then the desired hex file can be loaded from the project folder and flashed in the MCU (see **Figure 6**). If a sketch is modified, it must be recompiled (verify and compile). The IDE status field shows the folder containing the generated hex file. An example of a typical file path is

```
C:\Users\name\AppData\Local\Temp\buildxyz.tmp/
   Triac_...ino.hex
```

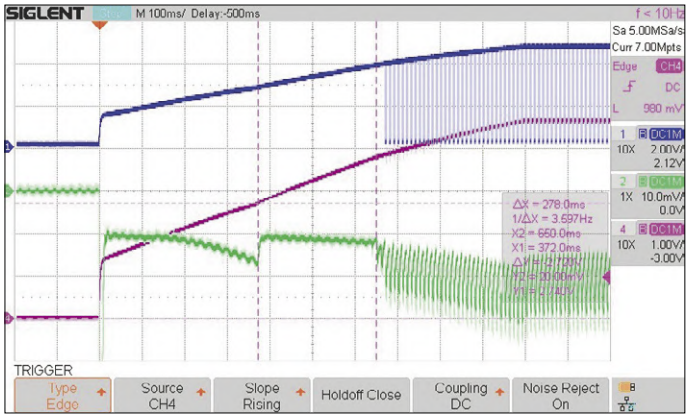


Figure 7: Relationships during switch-on. Ch1 (blue): gate pulses; Ch2 (green): current; Ch4 (red): V_{CC}. X1, Y1: Start of WDT and sleep mode, BOD level 2.7 V; X2: WDT timeout, wake-up.

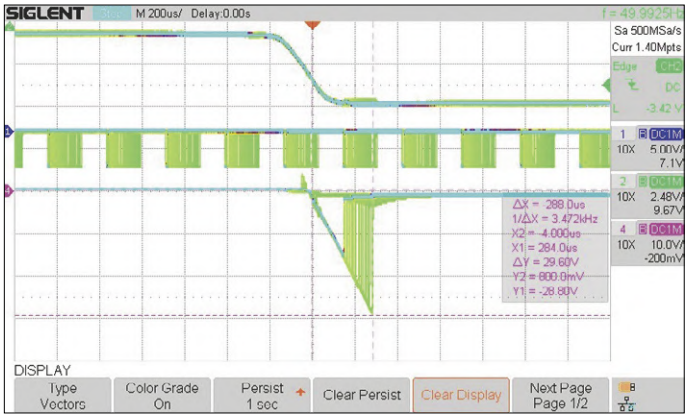


Figure 8: Voltage on the triac with uncorrelated trigger pulses over a period of 1 s. Top trace (Ch2): mains zero crossing; middle trace (Ch1): gate pulses; bottom trace (Ch4): voltage on triac.

As the folder name and build number are reassigned every day, the file also has to be updated daily in Atmel Studio. If the folder *AppData* is not visible in File Explorer, select *Show hidden files and folders* under *View/Options*.
Then load the desired hex file in Atmel Studio and program the ATmega microcontroller.

Warning: For programming, the module must be fully separated from the AC mains and powered by a safe low-voltage supply.

Listing 1

```
void setup()
{
 //after reset, start with sleep mode to stabilize VCC.
 //prepare the watchdog for wakeup

   asm ("WDR");              //clear wd-timer
   WDTCSR= 0x1c;             //set WDCE, WDE to enable setting
   WDTCSR= 0x1c;             //wd-timer 0.25s
   WDTCSR= 0x14;             //clear WDE
   WDTCSR= (WDTCSR | 0x40); //set WDIE, enable WD-interrupt
   interrupts();             //enable interrupts

   SMCR= 0x0f;               //prepare xtd-standby
   asm ("SLEEP");            //enter sleep mode
   SMCR= 0x00;               //finish, not necessary
   noInterrupts();           //disable interrupts, after wakeup

   MCUSR= (MCUSR & 0xf7);    //clear WDRF, wd-reset active
   WDTCSR= WDTCSR | 0x18;    //set WDCE, enable setting WDE
   WDTCSR= 00;               //clear WDIE, disables wd-timer

     . . .

}
```

Applications

All apps have the same initialisation procedure. On power-up, the BOD level of 2.7 V determines the end of the reset interval. After the following program start, the watchdog timer is activated and the program is put into sleep mode. This allows V_{CC} to rise quickly and reliably (**Figure 7**). The WD timeout triggers the WD interrupt and terminates sleep mode. Then execution of the actual program starts.
The sketch excerpt in **Listing 1** shows how setting the WD timer and sleep mode is implemented. The listing contains part of the code of the initialisation procedure. First the watchdog is set and activated. Then the microcontroller is put into sleep mode. The WD timeout triggers the WD interrupt, which wakes up the microcontroller.
A separate sketch is available for each example app. Each app has its own distinct functions, which can be selected using pushbuttons or jumpers. More information is available in the comments of the individual sketches.

A simple switch and timer app

The *Sketch Triac_no_zd_32* app manages without zero crossing detection, so it illustrates the basic programming of the module and its features. The module is operated from a 5 V AC power adapter for programming and testing. If you are interested in hardware-level programming of the WD timer, the various sleep modes, the interrupts, the timer functions or the I/O commands, you should study the code in detail. All of the software can be downloaded free of charge from the Elektor web page for this article [1].

With this app the module can act as a switch or as a timer, with the actual function selected by a pushbutton. With the additional operating mode 'Autostart', the timer starts running immediately after power-up and the load is switched off when the timer period expires. This function can be used to limit how long a

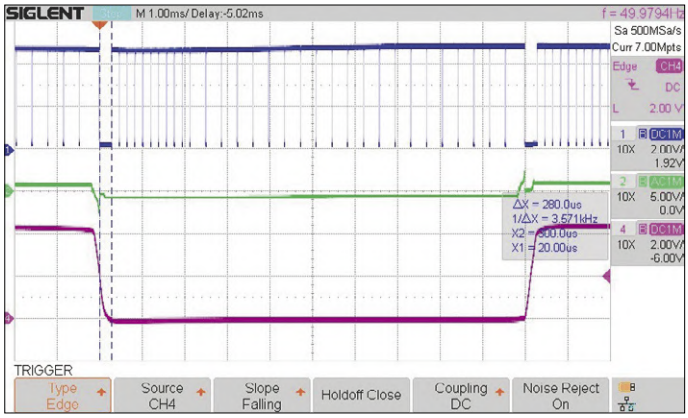


Figure 9: Switching on a LED lamp at the zero crossing of the mains voltage. Ch1 (blue): gate pulses; Ch2 (green): voltage on triac; Ch4 (red): zero crossing signal.

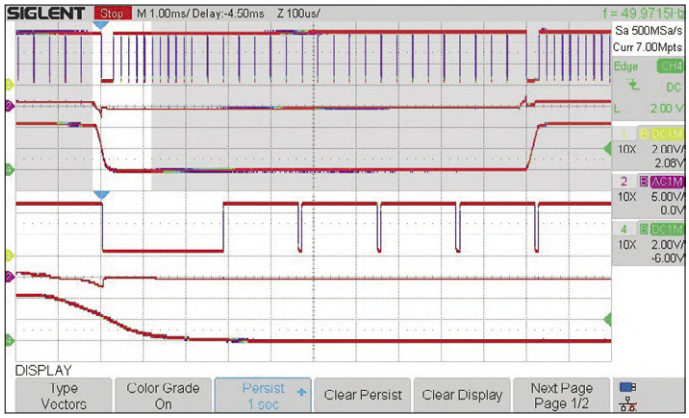


Figure 10: Switching on a LED lamp at the voltage zero crossing. Waveforms over 1 s. Ch1: gate pulses; Ch2: voltage on triac; Ch4: zero crossing signal.

lamp remains on. The switch-off time is stored in the program. In the on state, the program generates periodic 10 µs pulses with an period of 278 µs to drive the triac. The control pulses are not referenced to the AC mains frequency, so the switching times are not synchronized to the mains voltage. In the worst case, the load will only be switched on 278 µs after the zero crossing. At this point the mains voltage will have already reached 29 V, resulting in voltage spikes that can cause EMC problems. To illustrate this, the oscillogram in **Figure 8** shows what happens over an interval of 1 second. The trigger pulses are not correlated to the zero crossings, so they wander with respect to them. The voltage over the triac varies according to the offset between the zero crossing and the trigger pulse.

This is remedied in the next practically oriented application, in which the zero crossings are detected so that triac triggering is synchronised with the zero crossings. Unlike the following measurements, the reference voltage here is V_{CC}. T1 of the triac is at V_{CC}, so here you see the control pulses on the gate with negative polarity.

Switch and timer with zero crossing detection

Unlike the previous app, here a signal derived from the AC mains voltage is used to detect the zero crossings of the mains voltage. This signal is applied to the INT0 input, where its falling edges trigger interrupts that mark the start of the first half-wave. The end of this half-wave and the start of the next half-wave are determined by Timer 2. This avoids complicated hardware for precise detection of both zero crossings.

For reliable triggering with low load currents, the first trigger pulse of each half-wave is extended to 280 µs. It is following by two more 10 µs trigger pulses. This ensures that the I_L of the triac is exceeded even with high-impedance loads. **Figure 9** shows this relationship. The short bumps (approximately 3 V) in the voltage on the triac at the end of each half-wave are particularly noteworthy. They result from the relatively low current through the connected LED lamp, which drops below I_H at the end of each half-wave. However, this has little impact on the noise level.

Figure 10 shows the superimposed traces over an interval of 1 second. The control pulses are synchronous with the mains voltage.

Dimmer

Dimming the brightness of a lamp is a typical application for a triac. The lamp power is adjusted by phase angle control. Here the switch-on point of the load (phase angle) is varied with respect to the zero crossing point of each half-wave of the mains voltage. Dimmers can cause a lot of EMC problems because they generate voltage waveforms with high amplitudes and steep edges, depending on the phase angle. The worst case is switching at the crest of the mains voltage (with a phase angle of 90°). When the triac is triggered, the voltage on the triac quickly drops from the crest value to zero (**Figure 11**). Depending on the load, this results in very high currents and high noise levels. An upstream RFI filter (for example, from a discarded dimmer) is therefore essential.

Many LED lamps do not work with conventional dimmers, due to the design of their power supply. This module also has difficulties with quite a few LED lamps. You should therefore use dimmable lamps.

An interesting feature of this triac control is that after the first pulse, additional pulses are inserted until the end of the half-wave. This allows the module to be used with relatively complex loads. At full power, the first pulse after the zero crossing is extended to 190 µs. This ensures that I_H is exceeded

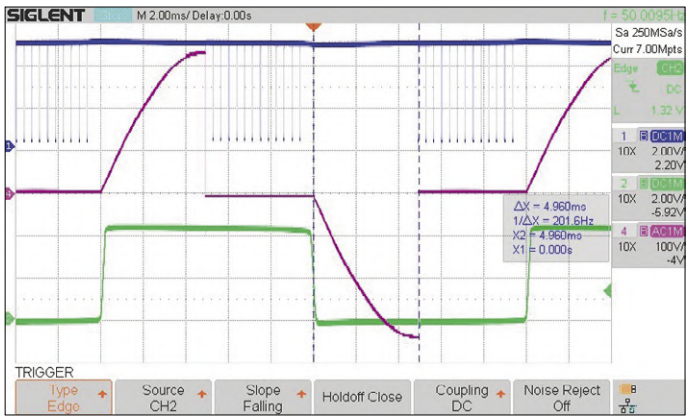


Figure 11: Dimmer with phase angle control. Ch1 (blue): gate pulses; Ch2 (green): zero crossing signal; Ch4 (red): voltage on triac.

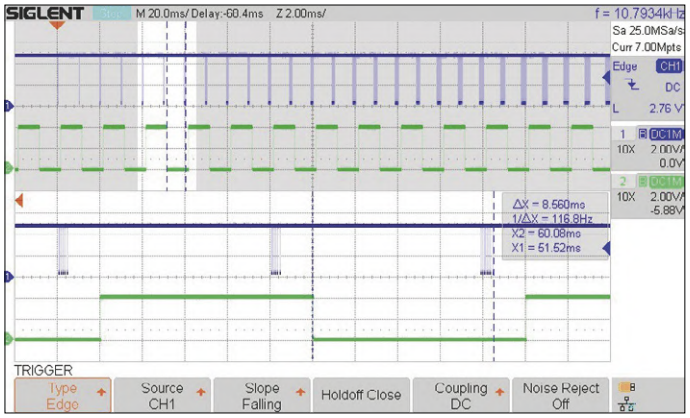


Figure 12: Control pulses after a restart. Ch1 (blue): gate pulses; Ch2 (green): zero crossing signal. The expanded area at the bottom shows the increase from five to seven pulses.

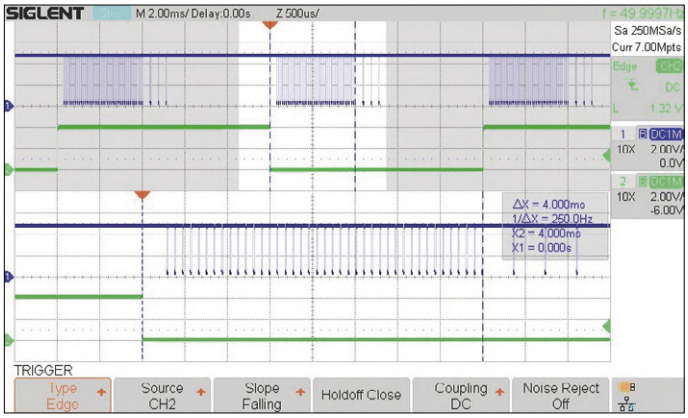


Figure 13: Control pulses in the settled state with expanded area below. Ch1 (blue): gate pulses; Ch2 (green): zero crossing signal. Filler pulses are generated for 5 ms after each zero crossing.

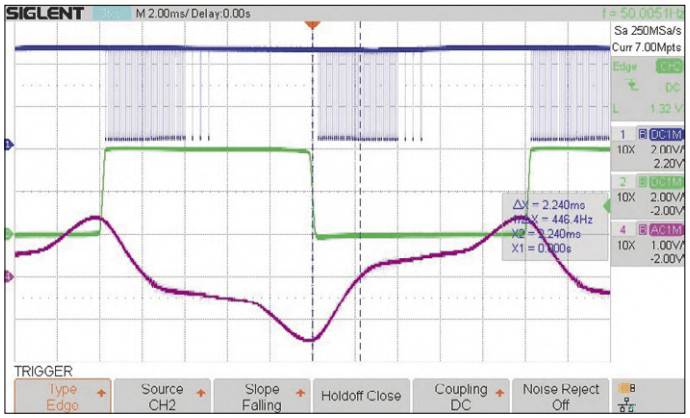


Figure 14: Transformer current with no load. Ch1 (blue): gate pulses; Ch2 (green): zero crossing signal; Ch4 (red): transformer current.

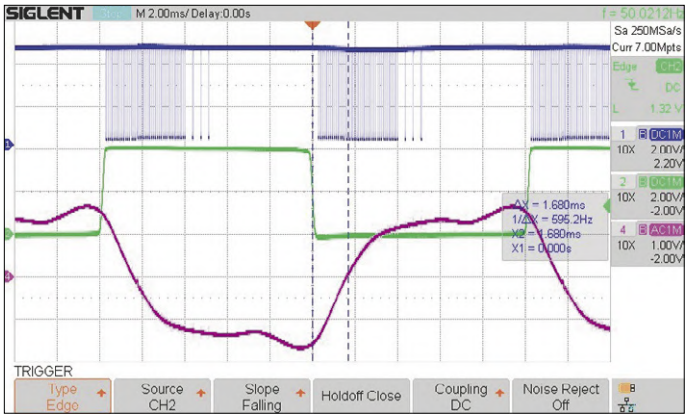


Figure 15: Transformer current with a small load. Ch1 (blue): gate pulses; Ch2 (green): zero crossing signal; Ch4 (red): transformer current.

even with high-impedance loads.
The output power can be set with the two pushbuttons or with optocouplers.

Soft start for inductive loads

Inductive loads, especially transformers, can draw very high switch-on currents. Here switching at the voltage zero crossing is the most difficult situation. The residual magnetisation of the core can also cause problems, because the switch-on current waveform may be offset from the zero line. As a result, the flux density in the core (the hysteresis loop) is shifted in one direction and can easily enter the saturation region. When that occurs, the winding inductance suddenly drops to zero and the rising current is only limited by the resistance of the winding and other elements. Ring-core transformers are especially prone to this. They should therefore be fused at a significantly higher level than what is required solely for the load current.
Sometimes a varistor or series resistor with a time-delay relay is used to limit the switch-on current until a symmetrical magnetic field (and correspondingly clean AC voltage) is established. In theory, this problem can also be encountered if the phase angle is slowly increased. When switching on the transformer, you should start with a phase angle of around 180° and then slowly reduce it to 0°. This is exactly what is done in this app. **Figure 12** shows the pulse sequence after a restart. The first trigger pulse is generated approximately 8.6 ms after the zero crossing. Another pulse is added ahead of it after each full wave, until the time is reduced to around 0.3 ms. **Figure 13** shows the situation after settling down. Within the first 3 ms, the load current goes through zero due to the phase shift. In order to achieve a continuous current flow, it is necessary to put the control pulses close together in this interval.
Figure 14 shows the transformer current with no load. The zero crossing signal shows the relationship to the mains voltage. Here you can see the magnetisation current, which reaches a maximum at the voltage zero crossing. The remaining part results from the iron and copper losses of the transformer. The phase lag of the current is not exactly 90°, due to losses and nonlinearities in the core. **Figure 15** shows the transformer current with a load. The load current and magnetisation current are superimposed. The load reduces the phase shift.
A prerequisite for faultless operation is fully symmetrical control of the triac. The control pulses must have the same offset to the zero crossing for both half-waves, so that the phase angle

About the Author

Roland Stiglmayr is an electronics engineer specialising in information technology, with 40 years of professional experience in R&D in the fields of control engineering, RF engineering, mobile radio, fibre optics engineering and data transmission. He is currently active as a consultant.

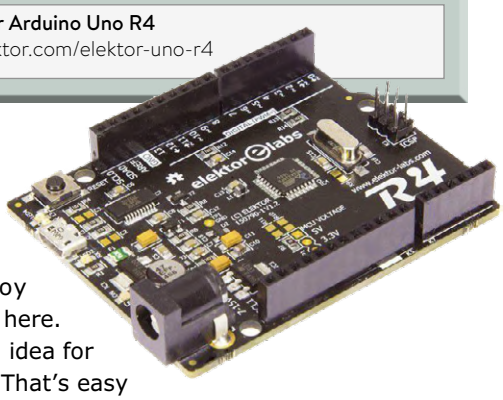
of the current is symmetrical. I_{GT} and especially I_H are significant for this. Reliable triggering is ensured by the RC network, which strengthens the leading edge of the gate current pulse. The transformer current must be greater than I_H for the triac to remain conducting. Ring-core transformers are difficult in this regard due to their high inductance. This problem can be solved by a small additional resistive load (R25-R27, optional). Some ring-core transformers even require an additional external load. Particularly with transformers, driving the triac with pulses alone was a challenge in the programming of the firmware. The various tricks can be easily understood from the detailed comments in the code.

Web Link

[1] Project support page: www.elektormagazine.com/190047-04

4 SALE @ WWW.ELEKTOR.COM

→ Elektor Arduino Uno R4
www.elektor.com/elektor-uno-r4



We hope you enjoy the apps described here. Maybe you have an idea for a new application? That's easy to do by modifying the firmware.

190047-04

Advertisement

Join the Elektor Community

Take out a GOLD Membership now!



GOLD MEMBERSHIP

- ✔ 6x Elektor Magazine (Print)
- ✔ 6x Elektor Magazine (PDF)
- ✔ 3x Elektor Industry Magazine (PDF)
- ✔ Access to Elektor Archive
- ✔ Access to over 1,000 Gerber files
- ✔ Elektor Annual DVD
- ✔ 10% Discount in Elektor Store
- ✔ Exclusive Offers

GREEN MEMBERSHIP

- ✔ 6x Elektor Magazine (PDF)
- ✔ 3x Elektor Industry Magazine (PDF)
- ✔ Access to Elektor Archive
- ✔ Access to over 1,000 Gerber files
- ✔ 10% Discount in Elektor Store
- ✔ Exclusive Offers

Also available:

The all-paperless GREEN Membership!

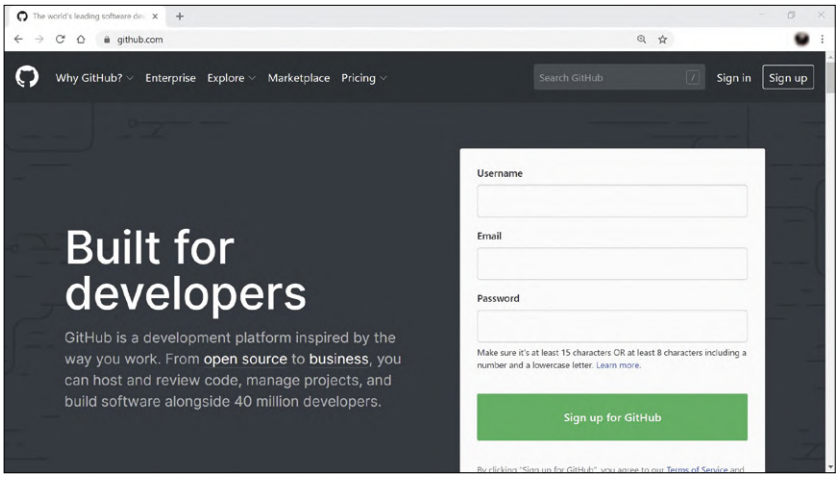
www.elektor.com/member

HOW-TO Download something from GitHub

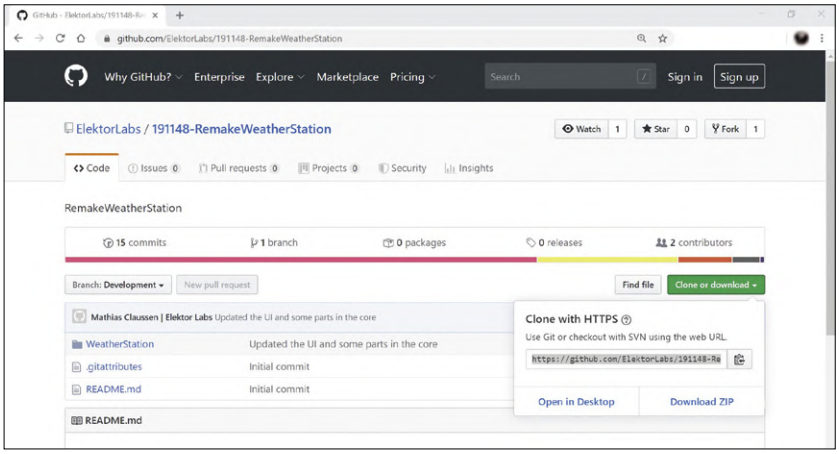
GitHub for Dummies

By **Clemens Valens** (Elektor Labs)

If you use microcontrollers, play with Arduino or do other things that involve open-source software, you're sure to have come across GitHub.



Although anyone should be able to use it, Github is intended for professional software developers and therefore has lots of functions that most people never need.



The green 'Clone or download' button is only visible in the root of the repository.

GitHub is *the* place for people to share source code for all sorts of open-source applications ranging from simple single-file Arduino sketches to huge cloud applications consisting of many repositories containing thousands of files, but also other things like electronics design files or simply text documents.

What is GitHub?

GitHub is a hub for *git*, where 'Git' is a versioning system, meaning it keeps track of changes made to any document it's aware of. The main purpose of 'git' and GitHub is to allow groups of people to work together on the same software project. Because GitHub is a tool for professional- and production-quality software development, it has lots of functions and options that most people never need, and that make it complicated to use.

How to download from GitHub?

At Elektor too we publish code on GitHub and tell people to download it from there. We then often get the question "*How do I download something from GitHub?*" This seems like a silly question to the experienced user, but it's true, you cannot just download the files you see listed in a repository. There is no Save or Download button, and when you click on a file, it gets opened in a browser, not downloaded. If you right-click on a file and then save it, you end up with a file containing HTML code, and not the contents you expected.

Raw mode

Actually, you can download a file if you open it in Raw mode first and then right-click to save it as a file. Sadly that's undoable in case there are a lot of files.

The green 'Clone or download' button

There is an easy download button, but it is only visible when you are in the root of the repository. Once you are there things become easy: just click the green button labelled 'Clone or download'. However, you might get confused by the surprising options about 'cloning with HTTPS', or open it on your desktop, or download ZIP. Here you just click 'Download ZIP' and ignore the other stuff. You do not need to sign in or have an account.

Arduino & GitHub

The name of the file you downloaded may end with '-master' and when you unpack it, you will get a folder with the same name. Most of the time this doesn't really matter, but it does

when the download happens to be an Arduino sketch. For the Arduino IDE the name of the .INO file must be the same as the name of the folder. The solution is simple: remove the '-master' part by renaming the folder.

Usually you do not have to do this when the sketch is inside **a folder nested inside the .ZIP file** because the nested folder will have the same name as the .INO file.
If the download is an Arduino library, then the Arduino IDE knows how to deal with it. You can import it using the 'Add .ZIP library' option from the 'Sketch' 'Include library' menu.

Releases

Many projects on GitHub are evolving continuously. If you download the .ZIP file as we did before, you get a so-called 'development snapshot' that may not work or may not do what you expect from it.
For this reason, one tends to publish stable versions, so-called releases, that work and have been tested. These releases are often hosted on another website, for instance on the project's official website, but GitHub also offers space for them. Check the Releases tab to see if there are any, and if there are, download the most recent one or another version or file you need.
You might also want to check the Releases tab if you're not interested in the source code of the project, but only in precompiled binaries to run on your platform. This is often the case when compiling the project is too complex for you or even impossible because you don't have the tools for it.

Cloning a repository

Finally, you can use the official GitHub method to download a project and that is by cloning it. In order for this to work you must first instal a git client on your computer. There exist clients with nice graphical user interfaces and clients that you control from the command line. In either case you will need the complete URL of the repository you want to clone. You can get it from the download button, again, but this time by copying the URL in the 'Clone with HTTPS' box. Click the copy button next to it to do this.

Command line (CLI) cloning

In the command-line interface, first navigate to the folder where you want your clone of the repository, then type the command 'git clone' and paste the URL of the repository behind it and hit enter. If all is well, the cloning process will start. Example:

```
git clone https://github.com/ElektorLabs/180468-ESP32-Weatherstation.git
```

Open in desktop

In the desktop interface, from the menu select 'File' 'Clone repository…', browse to the destination folder and paste the URL you copied before in the box labelled 'Repository URL', then click Clone.

Cloning is easier than downloading

Once you know how to do it, then cloning is probably easier than downloading. You don't have to unpack anything and you can clone directly into the folder where you want it. And you don't get the '-master' added to the repository's name either.

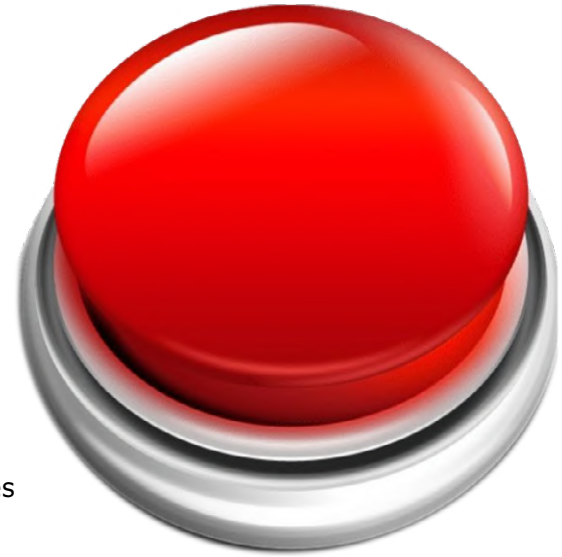
200047-01

Advertisement



My IoT Button:
A Button for the Web
Part 2: Prototyping with board and cloud

By **Dr Veikko Krypczyk** (Germany)



The basic idea of an IoT button is that you use minimalistic hardware to communicate with a remote service in the cloud. For this it is necessary to establish a connection to a local network. Small, low-cost microcontroller boards, such as the ESP8266 NodeMCU, are ideal for this task. Here we describe the relevant aspects with regard to the IoT device and present some ideas for your own projects. The nice thing is that it's all relatively easy to do, and the positive experiences come quickly. This creates encouragement for developing your own projects.

The Internet of Things (IoT) makes all imaginable devices smart and connects them to the outside world through the Internet. In the first article of this series, we outlined the basic architecture of an IoT landscape consisting of sensors, actuators, and a server backend. We saw that you usually do not have to set up your own server infrastructure, since there are many options available in the cloud. You communicate with them through the Internet. Things are really simple if you also adhere to the established standards. RESTful communication is widely used for requests. The client, which in this case is the IoT device, can send data to services (on a server) or request data from services. In the second article of this series, we look at the hardware and software aspects of the IoT device.

Selecting the hardware

What hardware should you use? There are basically several options — for example, a single-board computer such as the **Raspberry Pi**. Along with WLAN, the RPi has enough process-

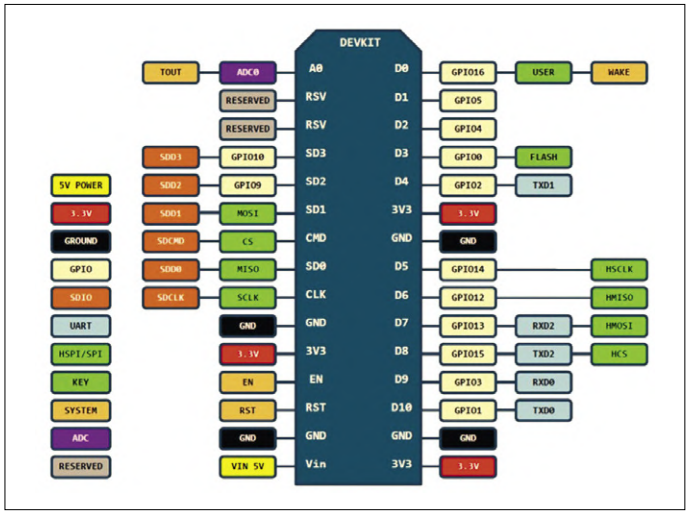


Figure 1: Pin assignments of the ESP8266 NodeMCU microcontroller board.

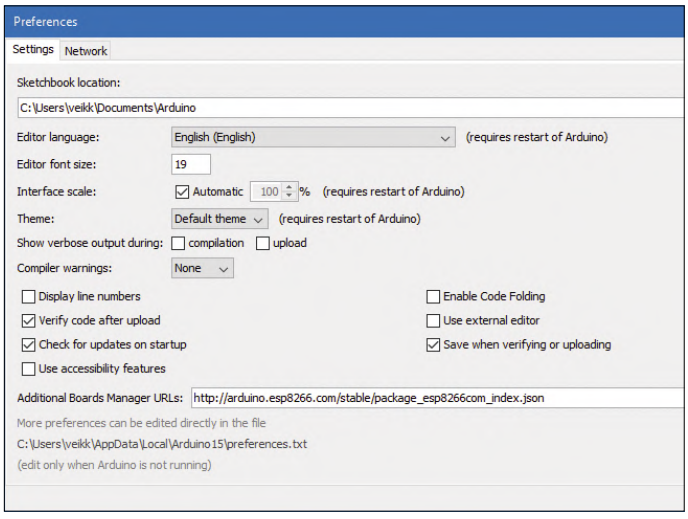


Figure 2: Arduino IDE configuration for programming the ESP8266.

ing power for all of the applications. If you want to use multiple buttons, sensors and actuators in the same place with a variety of functions, you can connect all these devices to the GPIO pins of the Raspberry Pi. Then the RPi can establish a connection to the cloud service via the network. However, the Raspberry Pi is physically too large for relatively small tasks, and it needs a continuous supply of power from an AC mains adapter.

So let's look at the **Arduino Uno** instead. As is well known, it is not a complete computer, but instead a microcontroller board. However, the hardware and software possibilities are extensive and there are countless libraries and example applications available. Programming is convenient in the Arduino IDE. However, the Arduino board does not support WLAN, so you have to add an expansion board to get this capability.

A much easier solution is to use the **ESP8266 NodeMCU** microcontroller board (see [1] and the inset). It has all the necessary functions integrated, in particular a WLAN interface (**Figure 1**).

You only have to connect a pushbutton switch between an I/O pin on the board and GND. Later on, pressing this pushbutton will be communicated over the WLAN. A status LED is already present on the board, so you don't need any other components. Power can be taken from the USB port of the computer, at least for now.

Setting up the ESP8266 NodeMCU

Just like the Arduino boards, the ESP8266 microcontroller can be programmed from the Arduino IDE. First you have to connect the board through a data cable to the USB port of the PC, and you may need to install the driver for the CH340 chipset [2]. If you do not already have the Arduino IDE on your computer (in which case you can skip this part), you need to install it now [3]. You also have to update the IDE with an add-on. Under *File Preferences*, enter the link address [4] at *Additional Board Manager URLs* on the *Settings* tab (**Figure 2**). Close the dialog window with *OK*.

Now you have to do the actual installation in the *Tools* menu, where you should find the entry *Generic ESP8266 Module*. Select the *Tools* menu, click on *Boards Manager* at the top of the pop-up menu, enter *NodeMCU* to limit the search range, and then perform the installation (**Figure 3**).

Once this is done, you can test the board with a simple 'Hello World' program (**Listing 1**). For this, copy the code and then load it into the microcontroller with *Sketch|Upload*. In this small routine, the LED on the board is alternately switched off for 3 seconds by the instruction `digitalWrite(LED, HIGH)` and switched on for 1 second by the instruction `digitalWrite(LED, LOW)`.

You also don't need much code to query a pushbutton (in this case connected to pin D7). For example, the following code section checks whether the switch is closed in order to trigger the subsequent actions:

```
int button=D7;
status=digitalRead(button);
if (status==LOW)
{ …… }
```

Now you have to give the ESP8266 access to the Internet. The hardware for this is already present on the ESP8266 board, in the form of the tiny grey-green WLAN module (**Figure 4**).

Listing 1. 'Hello World' (blinking LED) on the ESP8266.

```
#define LED D4
void setup()
{
     pinMode(LED, OUTPUT);
}

void loop()
{
    digitalWrite(LED, HIGH);
    delay(3000);
    digitalWrite(LED, LOW);
    delay(1000);
}
```

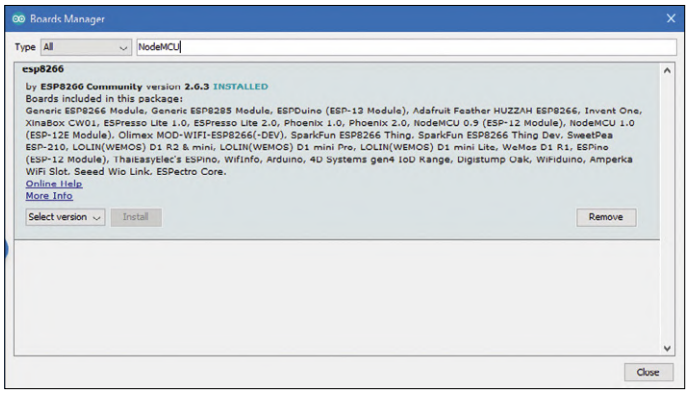


Figure 3: Installation of the libraries for the ESP8266 NodeMCU board.

The ESP8266 NodeMCU board

The ESP8266 NodeMCU is a microcontroller board with an ESP8266 chip clocked at 80 MHz and up to 4 MB of flash memory. The ESP8266 NodeMCU board also has WLAN module and a USB interface, including a USB to UART converter (CH340G). This little board has the following technical features:

- Supports WPA/WPA2
- Chip: ESP8266
- Processor: Tensilica L106 clocked at 80 MHz
- System architecture: 32 bit, Flash memory: up to 4 MB
- Supported protocols: SPI, UART, I2C, I2S, IR remote control, PWM, GPIO
- Logic level: 3.3 V
- Current drain: <10 µA in sleep mode; up to 200 mA under load
- Supply voltage and programming via micro USB port: 5 V
- Programming chip: CH340G
- Size: 5.5 × 3.1 × 1.2 cm; Weight: 42 g

The above specifications make the ESP8266 NodeMCU board a very suitable platform for small and especially low-power IoT devices.

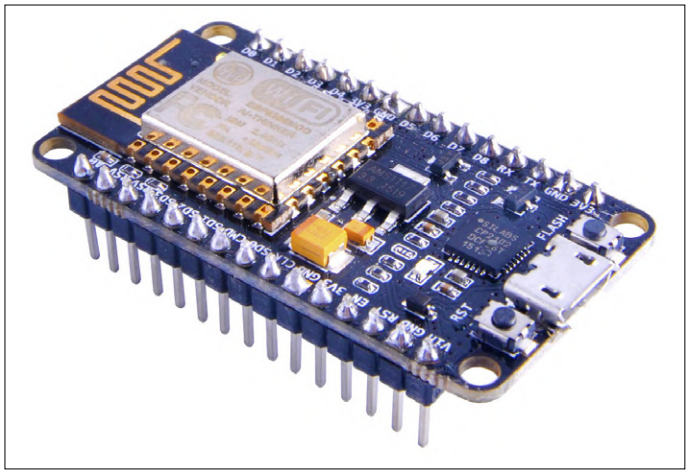


Figure 4: ESP8266 NodeMCU board with WLAN module.

Listing 2. Establish a link to the Internet.

```
include <ESP8266WiFi.h>
void setup()
{
    Serial.begin(115200);

    WiFi.begin(<SSID>, <Password>);

    while (WiFi.status() != WL_CONNECTED);
    {
        delay(1000);
        Serial.print("Connecting ...");
    }
}
```

If you want to use the WLAN functionality of the ESP8266, you have insert the line `#include ESP8266WiFi.h` in the header of the source code to include the corresponding library. The library expects the right data for establishing a connection to the Internet: the SSID and the password. For now you can do put these in the source code as static data, but later on you will want to allow users to configure them. Establishing the WLAN connection is remarkably simple (**Listing 2**).
Now you should proceed with the design under the assumption that when an event is received, a message is sent via the network. A REST endpoint acts as the receiver of this network communication, as described in the first article of this series. This endpoint accepts the data or message and can further process it and initiate a response. With a simple IoT button, the only data that has to be sent is the 'button press' event. With suitable expansion of this application, you could also sent additional data as parameters of the REST request. Now let's see how it works.

Data in the cloud

A REST endpoint is required in order send data to the cloud, which means to a remote server. You do not have to program this yourself; you can instead use existing services. Most of these cloud services are free of charge for tests and personal use. Let's proceed experimentally and step by step. At [5] you can create a free REST API for test purposes, which can be accessed from the microcontroller board via the WLAN. This works with just few mouse clicks, even without prior login. Have a look under [6]; there you can see that we have already created an endpoint (**Figure 5**). To test whether and how this endpoint can be reached from the microcontroller, you can send a GET request to the service when the button is pressed. This is implemented in the program loop (**Listing 3**).

The following line in the source code checks whether the microcontroller is connected to the Internet:

Listing 3. Send a GET request to the REST endpoint.

```
void loop()
{
  if (WiFi.status() == WL_CONNECTED)
  {
     status=digitalRead(button);
     if (status==LOW)
     {
      digitalWrite(LED, LOW);

      BearSSL::WiFiClientSecure client;
      client.setInsecure();
      HTTPClient https;
      https.begin(client, "https://iotbutton.free.
                           beeceptor.com");
      int httpCode = https.GET();
      if (httpCode == 200)
      {
        String payload = https.getString();
        Serial.println(payload);
      }
      else
      {
        Serial.print("Error: ");
        Serial.println(httpCode);
      }
      https.end();

     }
     else
     {
      digitalWrite(LED, HIGH);
     }

  }
}
```

```
if (WiFi.status() == WL_CONNECTED)
{ …… }
```

This means that all the enclosed source code lines are executed only if there is a connection to the Internet.
The next step checks whether the button has been pressed:

```
status=digitalRead(button);
     if (status==LOW)
```

If it has, an HTTP client is initialised, which enables the option to send HTTPS commands. A library is required for this, and it must be included in the header with `#include`. We set the value `https://iotbutton.free.beeceptor.com` to indicate the URL for our REST endpoint. This is followed by a simple GET request to the endpoint, and the result is received as a character string by the line

```
String payload = https.getString();
```

As a direct response to pressing the button, the LED on the board should light up when the button is pressed and go dark when the button is released (see above). You can check online to see whether the data actually arrived in the cloud. Go to the endpoint on the Beeceptor web page [6] and follow the requests on the endpoint (**Figure 6**).
Bear in mind that only 50 requests per day can be sent free of charge to the endpoint. During our experiments we exceeded this number, with the result that we received an error message with the code 429 in response to the GET request. As long as you do not exceed the maximum number of requests, the status code 200 (okay) and the data of the REST request will be sent back to the microcontroller. In the serial monitor of the development environment (*Tools*/*Serial Monitor*), you can keep track of the responses received by the microcontroller (**Figure 7**). This proves that pressing the button sends a REST request over the Internet. Now you have a working IoT button design. **Listing 4** shows the entire source code. The places that need to be adapted to your individual situation are marked in bold.

Current consumption and sleep mode

We opted for the ESP8266 microcontroller in particular due to its low current consumption. In practice you can assume the current consumption values given in [7]. With a simple Arduino program the current consumption is approximately 50 to 70 mA, rising to around 80 to 170 mA in WLAN mode, and for a few milliseconds the board can draw as much as 400 mA. The possibility of considerably reducing the power demand in deep sleep mode is therefore attractive. According to the data sheet and Internet sources, values in the low microampere range are possible, which means a simple battery (ordinary or rechargeable) should be able to power the board for many months. The supply voltage, for example from a powerbank, can be connected to the +5 V and GND pins. There are two forms of deep sleep mode:

- *Software deep sleep*: The ESP8266 automatically wakes up after a specific time. For this, GPIO16 (D2) must be connected to the RST pin. After the set time, a signal is sent to automatically restart the ESP processor. The maximum time duration with this option is approxi-

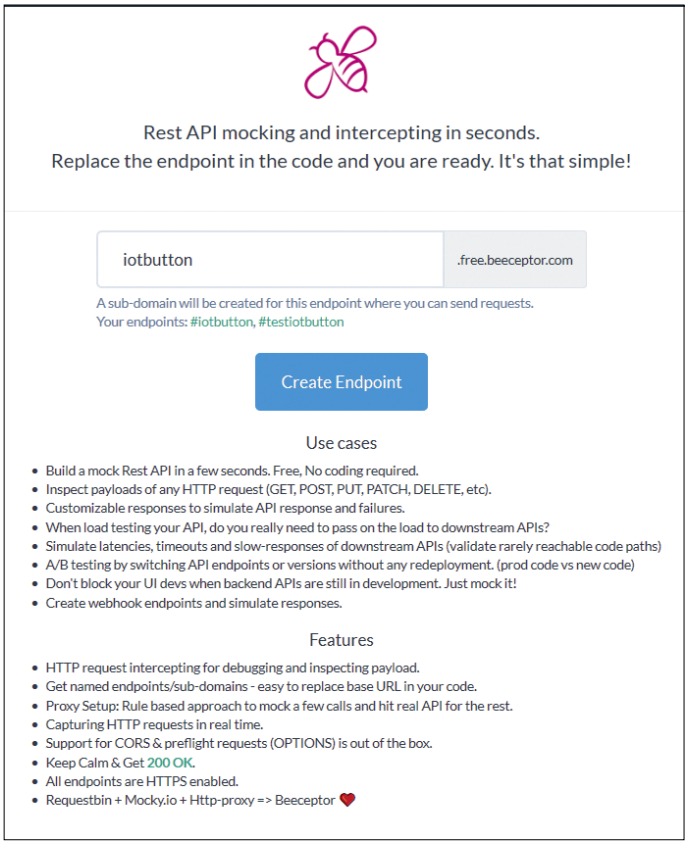


Figure 5: REST endpoint for tests in the Beeceptor cloud.

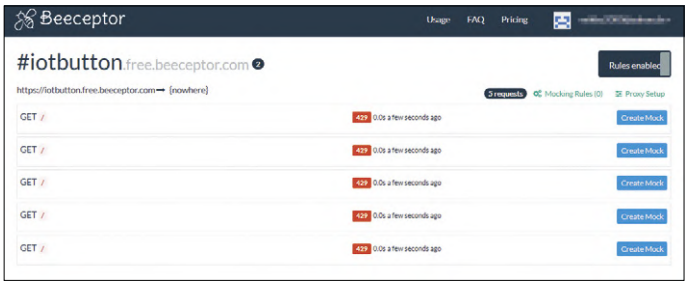


Figure 6: Logged requests to the REST endpoint in the Beeceptor cloud.

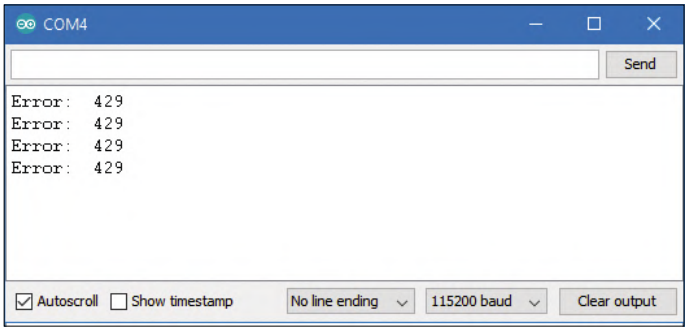


Figure 7: The responses of the REST endpoint are logged on the serial monitor. Here there is a 429 error message due to exceeding the number of allowed requests.

Listing 4. Send a GET request via REST.

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#define LED D4
int button=D7;
int status=1;
void setup()
{
    Serial.begin(115200);
    pinMode(LED, OUTPUT);
    digitalWrite(LED, HIGH);
    WiFi.begin(<SSID>, <Password>);
    while (WiFi.status() != WL_CONNECTED)
    {
        delay(1000);
        Serial.print("Connecting ...");
    }
}

void loop()
{
  if (WiFi.status() == WL_CONNECTED)
  {
     status=digitalRead(button);
     if (status==LOW)
     {
      digitalWrite(LED, LOW);
      BearSSL::WiFiClientSecure client;
      client.setInsecure();
      HTTPClient https;
      https.begin(client, "https://iotbutton.free.
                           beeceptor.com");
      int httpCode = https.GET();
      if (httpCode == 200)
      {
        String payload = https.getString();
        Serial.println(payload);
      }
      else
      {
        Serial.print("Error: ");
        Serial.println(httpCode);
      }
      https.end();

     }
     else
     {
      digitalWrite(LED, HIGH);
     }
  }
}
```

Listing 5. Send a GET request via REST and go into deep sleep mode.

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>

#define LED D4
void setup()
{
    Serial.begin(115200);
    pinMode(LED, OUTPUT);
    digitalWrite(LED, HIGH);
    WiFi.begin(<SSID>, <Password>);
    while (WiFi.status() != WL_CONNECTED)
    {
        delay(1000);
        Serial.print("Connecting ...");
    }
}

void loop()
{
  if (WiFi.status() == WL_CONNECTED)
  {
      digitalWrite(LED, LOW);
      BearSSL::WiFiClientSecure client;
      client.setInsecure();
      HTTPClient https;
      https.begin(client, "https://iotbutton.free.
                           beeceptor.com");
      int httpCode = https.GET();
      if (httpCode == 200)
      {
        String payload = https.getString();
        Serial.println(payload);
        digitalWrite(LED, HIGH);
        delay(2000);
        ESP.deepSleep(0);
      }
      else
      {
        Serial.print("Error: ");
        Serial.println(httpCode);
      }
      https.end();
  }
}
```

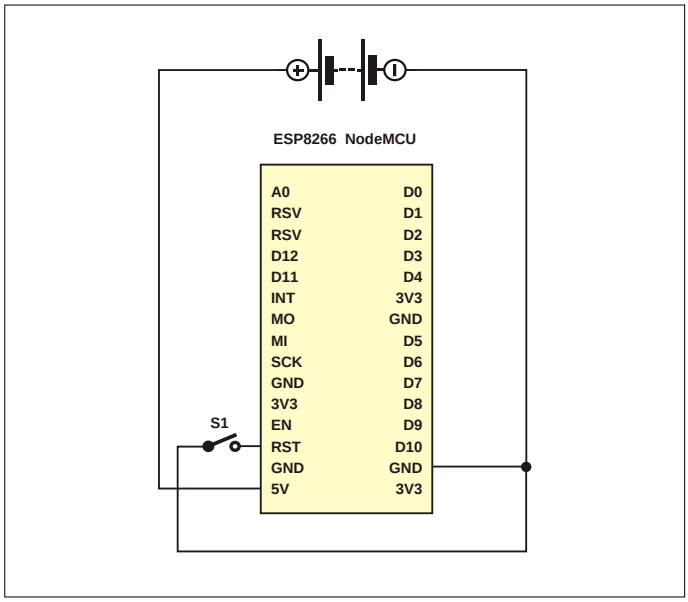


Figure 8: The circuit diagram consists of the microcontroller board and a single pushbutton.

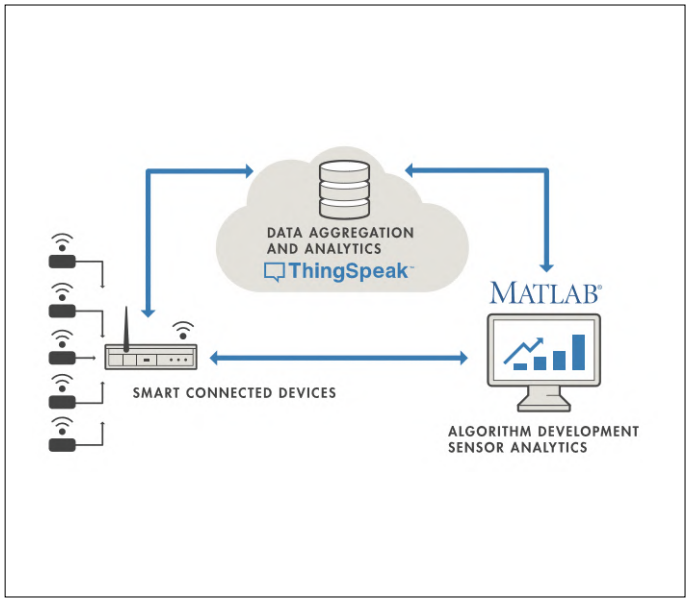


Figure 9: Collaboration between the IoT device, IoT cloud and other services, using ThingSpeak as an example.

mately 71 minutes. This option is a good choice in scenarios where periodically measured values are transmitted. The instruction for this is: `ESP.deepSleep(SLEEP_TIME * 1000000)`

- *Interrupt deep sleep*: The ESP8266 does not automatically restart, but instead is awakened from deep sleep mode by an interrupt. This form of deep sleep mode is initiated by `ESP.deepSleep (0)`. In this case there must be no connection between GPIO16 and RST. When the event is triggered, a Low signal is sent to the RST pin. This restarts the ESP8266, and the firmware is executed.

For the IoT button, this means it is no longer necessary to connect the pushbutton to a GPIO pin. Instead it is connected between the RST pin and ground, as shown in Figure 8. Consequently, there is no need to check for a button press in the code. Instead, the microcontroller is put into sleep mode after the data has been sent successfully over the REST interface (**Listing 5**). Compile the source code (sketch) and load it into the microcontroller. After starting up, the microcontroller connects to the network, sends a message, and goes into sleep mode. When the microcontroller is awakened by the pushbutton (RST), this process is repeated from the beginning. You can check online whether the GET request has reached the REST API.

Back in the cloud

The basic IoT button implementation is ready now. The actions to be taken when the microcontroller message is received can be configured in a few further steps, because there are also various cloud services available for these scenarios – such as *Microsoft Azure IoT* [8][9][10] or *ThingSpeak* [11]. The proce-

Web Links

[1] ESP8266 NodeMCU user manual: www.handsontec.com/pdf_learn/esp8266-V10.pdf

[2] CH340 driver: https://sparks.gogo.co.nz/ch340.html

[3] Arduino IDE: http://www.arduino.cc/en/main/software

[4] Board manager URL: http://arduino.esp8266.com/stable/package_esp8266com_index.json

[5] Beeceptor: https://beeceptor.com/

[6] IoT button console: https://beeceptor.com/console/iotbutton

[7] ESP8266 current consumption: https://arduino-hannover.de/2018/07/25/die-tuecken-der-esp32-stromversorgung/

[8] Microsoft IoT cloud: https://azure.microsoft.com/en-en/services/iot-hub/

[9] ESP8266 in the Azure cloud: https://sandervandevelde.wordpress.com/2019/05/07/connection-a-cheap-esp8266-to-azure-iot-central/

[10] NodeMCU in the Azure cloud: https://www.thingforward.io/techblog/2018-05-22-connecting-nodemcu-to-microsoft-azure-iot-hub-part-1.html

[11] ThingSpeak: https://thingspeak.com/

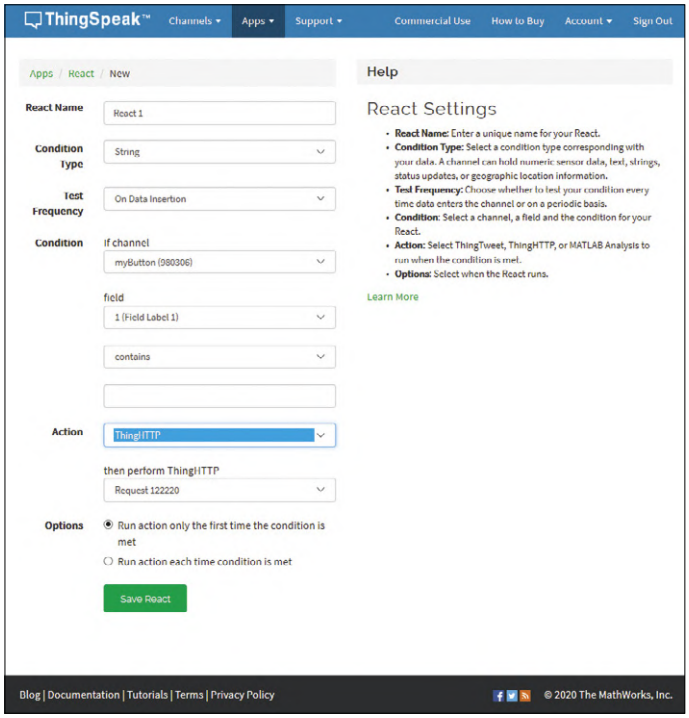


Figure 10: Configuring a REST endpoint (channel) at ThingSpeak.

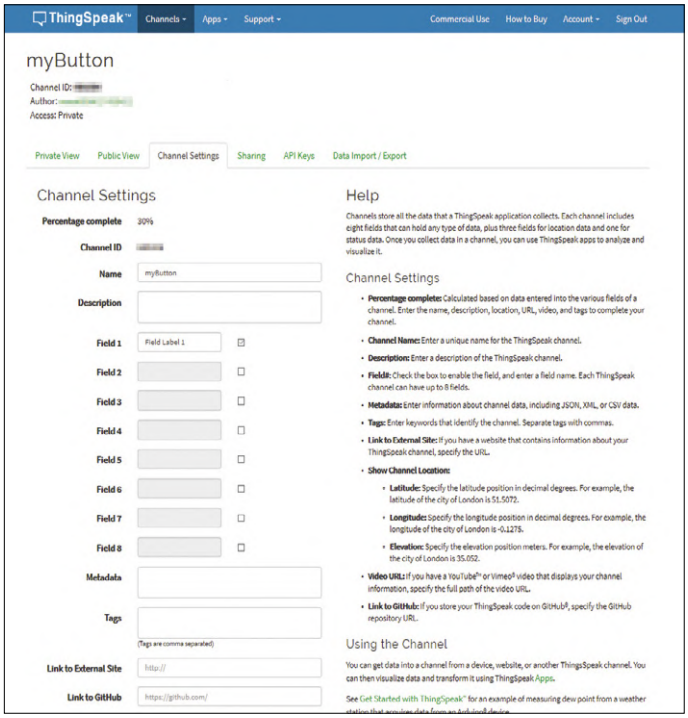


Figure 11: At ThingSpeak, the actions for a channel are determined by apps.

dure essentially consists of configuring an endpoint directly in the cloud service and then contacting it with the IoT device via the Internet. Then the desired actions can be initiated through the service (**Figure 9**).

You can set up a free account for the ThingSpeak service, which only requires verification of an email address, and then create a new REST endpoint (channel).

Fill in the detailed form, which is explained to the right (**Figure 10**), or first give your newly created endpoint a name. Each channel is given a unique URL and the ability to not only receive requests but also send additional data by means of URL parameters. Although no additional parameters are needed for a simple IoT button, the channel read and write keys (as well as example application for how to include the keys in the GET requests) can be found under *My Channels/API Keys*. Once a channel is correctly configured, you can use the menu item *Apps* to define which actions should be triggered (**Figure 11**) — for example, sending an email via a web service or posting a tweet on Twitter. Here the IoT backend acts a sort of intermediary.

Hazard warning: IoT devices that communicate with services on the web need access to these services. For this purpose, they have to identify themselves, for example with a user name and password or with unique features (keys). These values must be stored in the source code. Be sure to keep this information protected, for example when copying the source code or publishing it on the Internet (for general use).

Conclusion and outlook

Experimenting with the ESP8266 microcontroller is fascinating. Few if any additional components are necessary to implement IoT scenarios. Not much more work would be needed to make the IoT project ready for production, or even turn it into a commercial application. This mainly involves minor mechanical peripherals (enclosure) and the power supply (permanent battery connection). Extensions are also conceivable. For example, replacing the pushbutton by a motion detector would allow the device to generate an alarm and communicate it via the Internet when an unauthorised person approaches a secured property. The actual work is done online through the configuration in the cloud.

(190303-B-04)

4 SALE @ WWW.ELEKTOR.COM
→ ESP8266 NodeMCU
www.elektor.com/17952

TMS0280 Speech Synthesizer

Peculiar Parts, the series

By **Neil Gruending** (Canada)

In 1976 Texas Instruments (TI) wanted to design a device that would pronounce a word to be spelt out on a keyboard as a way to teach spelling. It needed to be inexpensive, so it needed to store as many words as possible in about 256 kbits of memory. It also needed to be portable, so the size and power consumption was also very important. By 1978 TI had managed pack the apparent performance of a several-thousand-dollar computer into the $50 device that we all know as the *Speak & Spell* thanks to the TMS0280 speech synthesis IC. TI already had the TMS1000 microcontroller that we looked at in the previous installment but is wasn't powerful enough to do speech synthesis on its own. For that TI turned to their team that included Larry Brantingham, Paul Breedlove, Richard Wiggins and Gene Frantz. After a lot of research, they determined that consumers associated a dull monotone voice with computer generated speech which worked in their favour to greatly simplify the synthesis requirements. The result was the TMS0280 which used linear predictive coding to generate speech at a 10-kHz sample rate from audio data encoded at 1200 bits per second.



Figure 2: TI Speak & Spell. Author attribution: Bill Bertram 2006. No changes made. CC-BY-2.5 — Attribution: https://creativecommons.org/licenses/by-sa/2.5/deed.en

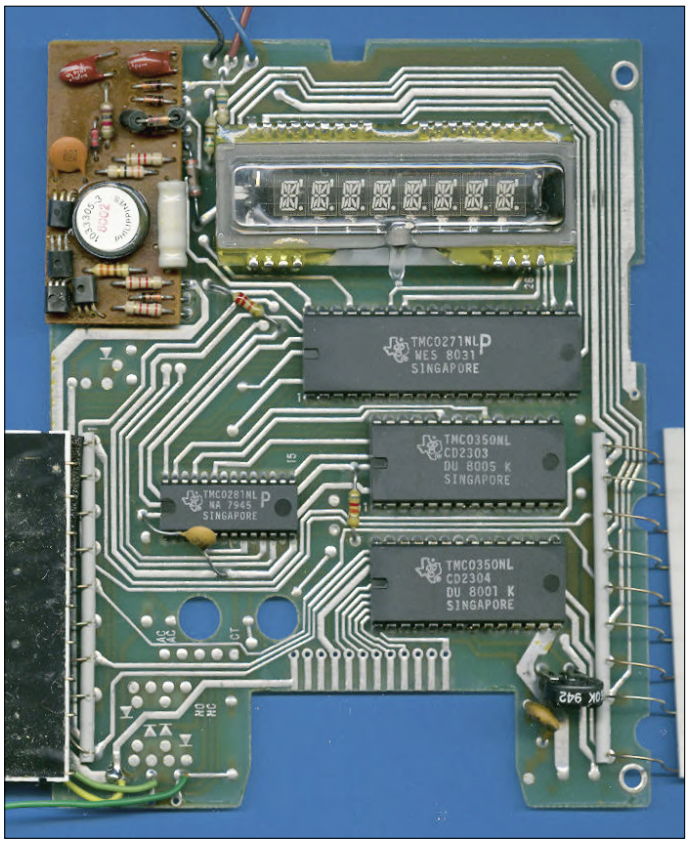


Figure 1: TMC0280 labelled TMC0281 in a Speak & Spell. Source: Datamath.

The encoded data was stored as data frames that contained information like energy, pitch, and reflection coefficients that were needed for the speech algorithms. Sometimes the data frames were hand-generated in the lab but usually the encoding parameters were generated by large computer programs that analyzed real speech data. Either way, the goal was to create encoded data that could be clearly understood and sounded as close to the original speech as possible.

The TMS0280 contained a digital signal processor with a multiplier, an adder implementation of a two multiply lattice filter and a masked ROM for the program memory. The ROM allowed the data rate, excitation and filter characteristics to be changed as needed for different production variations. It also contained an oscillator, interface logic and a 200-mW D/A convertor for the audio output.

Now obviously TI hasn't produced the *Speak & Spell* for about 30 years but there's a strong enthusiast community modifying them. Probably the most famous one was the *Speak & Spell* that E.T. used to call home in the movie *E.T. The Extra-Terrestrial* — TI even made a special themed language cart based on the movie. So maybe it's time to take that *Speak & Spell* that's been hiding in a box and see what you can do with its TMC0280.

190383-C-01

Interactive

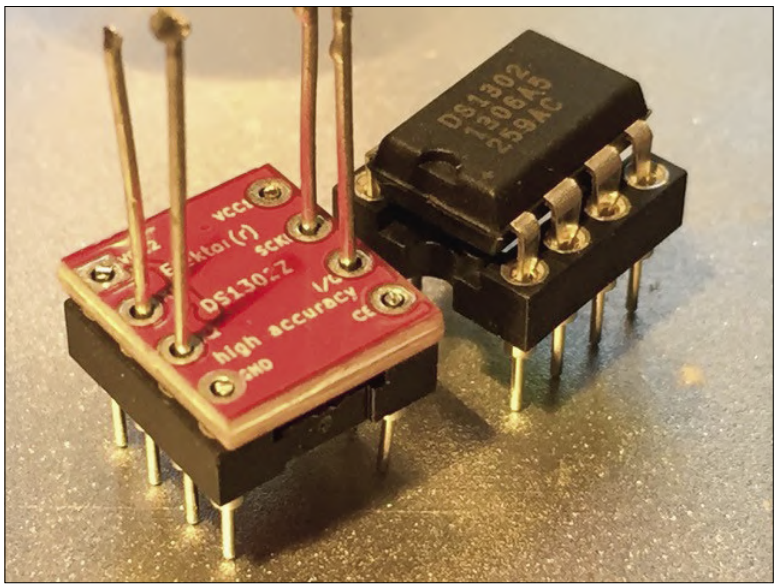
Corrections & Updates || Questions & Answers

By **Clemens Valens** (Elektor Labs)

Updates of and additions to projects published in ElektorLabs Magazine spiced up with tips & tricks, tech advice, and answers to reader's questions.

Upgrade for the 3-way Display Alarm

Elektor Labs magazine 3/2019 (May & June), p. 52 (180620)



It's me again with a reader tip from the field. The article states that the RTC BoB is too tall to simply replace the DS1302 (IC2) of the clock. It then goes on to mention three possible ways to modify the clock to make the BoB fit. I have found a fourth possibility which I personally consider to be the best because it does not require any changes to the clock at all as it fits in place of the existing IC. The trick is not to use the pinheaders shown, but to use cut off component wires, plenty of which should be around in the electronics enthusiast's work area.
Once the BoB is equipped with its three components, place it — component-side facing down — on top of an 8-pin IC socket. Then stick eight wires from above through the board's holes into the contacts of the socket, as shown in the photo. Using a pair of small, flat pliers is better for your fingers and makes it easier to ensure that all pins go down all the way into the contacts. Cut the wires flush with the little PBC and solder them. Carefully lever the BoB out of the helper socket and insert it into the socket for IC2 on the clock PCB. As you can also see on the photo, the RTC is now even lower than the original IC.

Hans Schneider (Germany)



Flicker-free LED Dimmer

Elektor Labs magazine 5/2019 (September & October), p. 12 (190062)

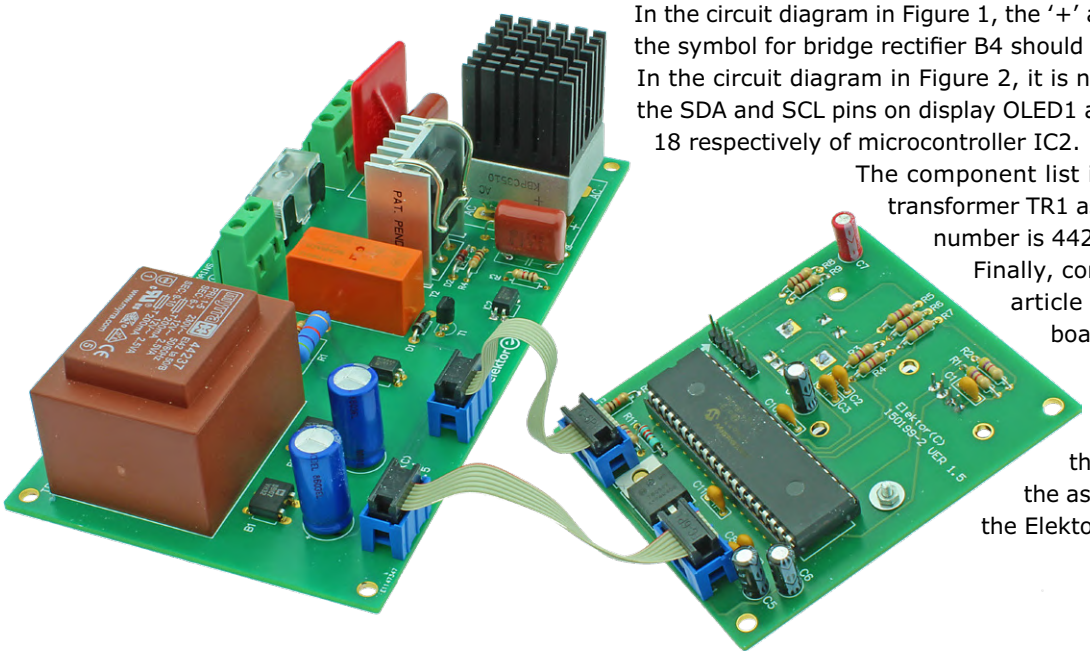
Even though I enjoyed reading the article, I felt that it was not complete as it doesn't mention that the two advantages of the method, i.e. less flicker and better EMC, are offset by two disadvantages.
First of all, the colour of an LED depends on the operating current. The principle proposed by J. Waegebaert changes the operating current and thus the colour of the light produced. This can be disturbing in, for example, professional photography. When operating with rectangular-pulsed current (PWM), the individual current pulses always have the same value. Therefore, the resulting colour of the light does not change in this operating mode, and remains constant over the entire dimming range.
Second, smooth dimming in the lower brightness range

is not guaranteed. Not all power LEDs have a defined behaviour in the lower current range. There are manufacturers who specify a minimum current 10% up the nominal current (for example). With pulsed current it is still possible to dim smoothly down to zero.
In practice one will have to find a suitable compromise for each application.

Daniel E. (Switzerland)

Single-phase 1-kilowatt AC Motor Drive

Elektor Labs magazine 1/2020 (January & February), p. 62 (191142, 150199).



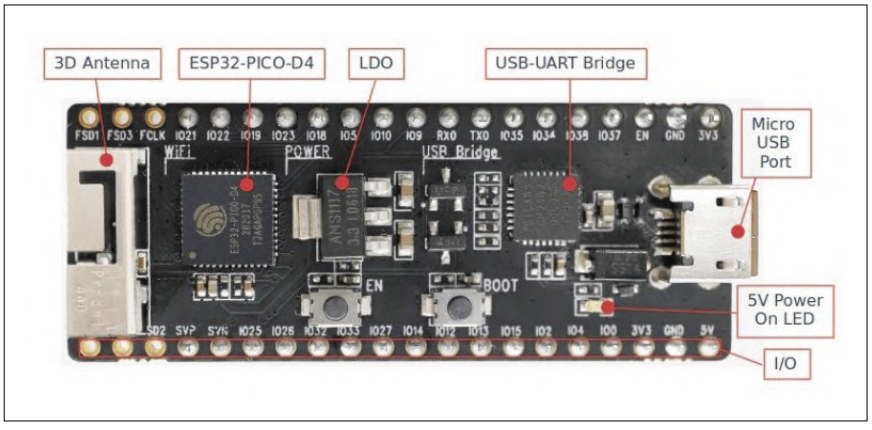
In the circuit diagram in Figure 1, the '+' and '–' symbols shown inside the symbol for bridge rectifier B4 should be swapped.
In the circuit diagram in Figure 2, it is not immediately evident that the SDA and SCL pins on display OLED1 are connected to pins 23 and 18 respectively of microcontroller IC2.
The component list incorrectly identifies power transformer TR1 as a Myrra 44327; the correct number is 44237.
Finally, contrary to what is said in the article text, the power electronics board is double-sided.

None of the above advice has any consequence for the operation of the project, or the associated PCBs available from the Elektor Store.
www.elektor-labs.com/3980

ESP32-PICO-Kit v4.1 Cannot Be Programmed With Arduino IDE?

Q: The two ESP32-PICO-Kit modules I bought last month have a version number of 4.1. The modules I bought earlier are version 4. The older modules can still be programmed with the Arduino IDE. With the new ones however I was able to upload just one sketch, and then uploading stopped working. The uploaded sketch ran correctly. I have tried different baud rates. What can be the problem?

S. Kepper (Germany)



A: The two modules that you sent us can be programmed on our test system without any problem. However, we did discover that there appears to be a driver or other software problem with certain combinations of Windows 10 and USB controllers (AMD Ryzen platform). Unfortunately, we have no control over these components. As a workaround, the behaviour can be improved by using an old USB 2.0 hub.

Elektor Labs

Epilogue: Many thanks for your reply. Your answer incited me to take a closer look at the USB ports on my PC. It has both USB 2.0 and USB 3.0 interfaces. I have found that I can/may not use USB 3.0 under Windows 7 for the newer ESP32-PICO-Kit 4.1 modules. Maybe this discovery is also interesting for other users. Why programming the module the first time on a USB 3.0 port worked, remains a mystery.

S. Kepper (Germany)

190379-C-01

BASIC for the ESP32/ESP8266 (2)

An hourglass with the ESP8266 and Annex WiFi RDS

By **Peter Neufeld** (Germany)

In the previous edition of Elektor, Annex WiFi RDS was described as a platform for programming ESP8266/ESP32 controllers in BASIC. Now we are talking about a practical application: the firmware for an auto-rotating hourglass — not in C, C# or C++, but in fast, straightforward and simple BASIC.



Figure 1: On the finished hourglass a four-digit digital clock lights up on the left and on the right a real and illuminated hourglass is turned by a hidden servo at the exact time.

The BASIC development environment Annex WiFi RDS [2] described in *Elektor* March & April 2020 [1] is now available as a practical entry project. The ESP8266 modules, available in various designs for little money, open up endless possibilities for the electronics engineer projects of many levels, not only thanks to their WLAN capability, but also because they can be easily connected "wirelessly" to smartphones, tablets and PCs. Since there are also various sensors and actuators that can be easily connected to the microcontroller for little money, do-it-yourself construction could actually be a real pleasure again — if it weren't for the rather complicated programming of modern SoCs.

You can choose between different programming languages, various IDEs, libraries, drivers, resources and compilers. Or should you sneak past them and take some unfamiliar code that is difficult to understand and somehow transfer it to the module? It gets really complicated if you want to control your project via web interface with a smartphone. The development environment called Annex WiFi RDS makes such a project much simpler by using the BASIC programming language. This in turn leads to quick achievements.

A simple project

To realize a digital clock with the ESP8266 that displays the weather forecast at the same time is a matter of a few BASIC lines, as example projects on the Annex website [3] prove. I came up with a simple idea with special elegance: the combination of a digital and an analogue display. In the project "ESP8266 Sand Clock", a small one-minute hourglass located next to a conventional four-digit digital clock, is to be turned over precisely in time.

But that's not all: the trickling sand could be illuminated. Four NeoPixel LEDs are used as background lighting. Of course,

the display's brightness should be adjusted to the surroundings, to which an LDR is added as a light sensor. If you add a 1-wire temperature sensor, you can even show the room temperature on the display.

Of course, the hourglass should also be controllable via WLAN using the integrated web interface. If you think of the code and the many potential bugs, sometimes it's just a heroic approach forever scheduled to *mañana* ;-).

But because the 'real thing' is much easier thanks to BASIC, the finished hourglass shown in **Figure 1** doesn't need as much software work as one might think.

Browsing the extensive, practice-oriented help file for Annex WiFi RDS, including the contents of the corresponding website, gave me some ideas for new projects. Despite its power, BASIC is easy to master even for casual programmers. Thanks to the Annex Rapid Development Suite, software development is considerably simplified compared to other current solutions.

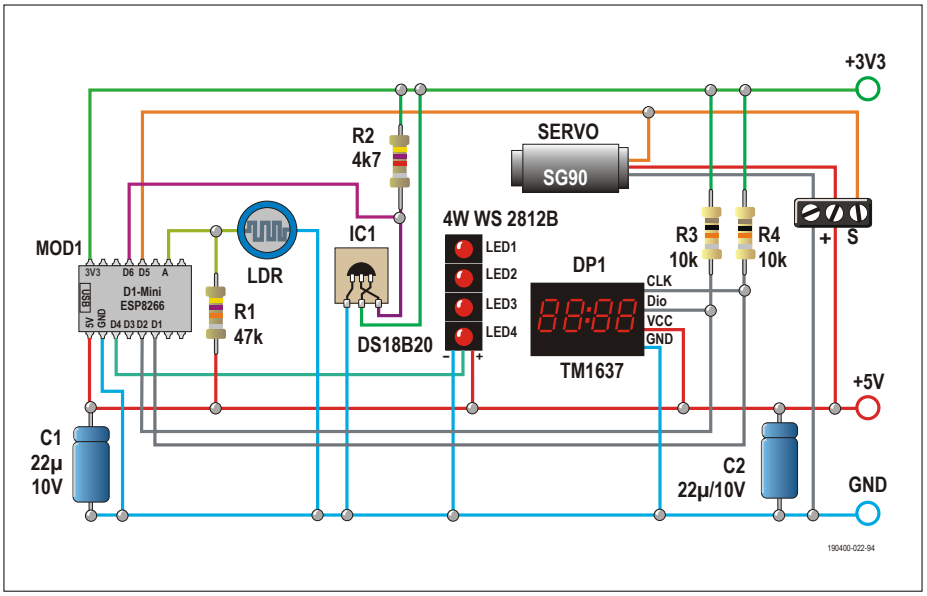


Figure 2: The electronic hourglass circuit is really this simple: in addition to the ESP8266 module, four resistors, two capacitors, four LEDs, a temperature sensor, a display and a servo are required. A parts list is not necessary.

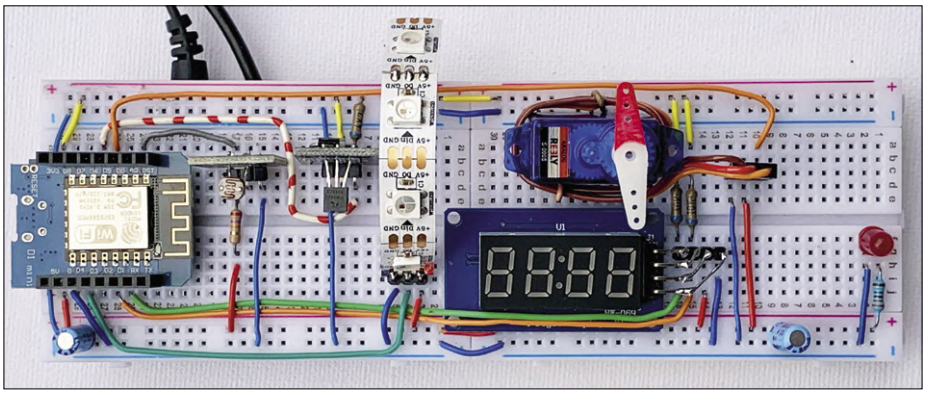


Figure 3: The test setup of the hourglass on a breadboard according to the circuit of Figure 2.

The electronics

The D1-Mini controller board used offers an ESP8266 module and, thanks to its pins, is suitable for experiments on a breadboard, as all relevant SoC connections are brought out. Via its USB interface and auto-flash capability, the module is easily and quickly loaded with firmware using the Annex Toolkit [4] without hardware intervention. Don't forget to set the correct time zone on the CONFIG page and enter the BASIC script in the autostart line there (the path of my active script is always "*/program/default.bas*").

For my tests I built the circuit from **Figure 2** on a breadboard. The result can be admired in **Figure 3**.

If the supply voltage is connected via the micro USB socket, +5 V and +3.3 V are also brought out and both voltages are additionally loadable (within limits), thanks to a Schottky diode and a 3.3 V voltage regulator inserted. The currents required for the peripherals used here do not push this to the limit. If you want to be absolutely sure, you can power the servo separately from +5 V. The two electrolytic capacitors between +5 V and ground absorb the current peaks of the servo.

Please note that the inputs of an ESP8266 can handle a 3.3 V level maximum. Therefore almost all external pull-up resistors are connected to +3V3. The internal voltage divider of the analogue input ensures that voltages from 0 to 3.3 V can be processed and that even 5-V swings do not cause problems. The voltage divider with the LDR should be dimensioned in such a way that with full lighting, no more than 500 mV appears at the input. If the analogue input remains unconnected, 0 V is applied here and the display is consequently set to full brightness.

Caution when connecting the temperature sensor DS18B20! The unit with module board has a different pin configuration than the sensor in the TO housing.

BASIC software

After one-time installation of the firmware on the ESP8266 using the Annex Toolkit [4] and following the instructions in [5], you only need a web browser to access the module's web server. New code can be quickly written in the editor or inserted into default.bas by using copy & paste.

In the Annex help file (F2 key in the editor) you will find explanations, pin assignments and script snippets for the TM1637 LED display, the analogue servo, the NeoPixel LEDs and various commercially available actuators and sensors as well as for the different ESP modules. The online help also contains a helpful

Listing 1. Preamble of hourglass.bas

```
'#################### HOURGLASS ########################
' 4Digit digital clock combined with 10 minutes hourglass
  VERSION$      = "V3.3"
' 10/2019  Peter.Neufeld@gmx.de   min:ANNEX_1.39b6
' - ESP8266-Module: WEMOS-D1-Mini
' - Display: 4digit 7segment TM1637
' - Servo: analog servo (SG90)
' - Background light: 4 NeoPixel-LEDs
' - Temperature sensor: DS18B20
' - Light sensor: LDR @ analogue input
' GPIO-Mapping (PIN-Label) for WEMOS and NodeMCU boards
D0=16:D1= 5:D2= 4:D3= 0:D4=2
D5=14:D6=12:D7=13:D8=15:D9=3:D10=1
SERVO_PIN   = D5  'Servo @ GPIO14=D5
TM_DATA     = D2  'TM1637 Data @ GPIO4=D2
TM_CLOCK    = D1  'TM1637 Clock @ GPIO5=D1
TM_BRIGHT   = 7   'TM1637 brightness
TURN_TIME   = 10  'Turn hourglass every xx minutes
SERVO_LEFT  = 180 '1. Position of hourglass
SERVO_RIGHT = 0   '2. Position of hourglass
SERVO_POS   = 0   'actual position of hourglass
BLINK       = 0   '255= visible colon @ TM1637
ADC_DARK    = 450 'max. ADC value for dark LDR,
'            depends on LDR and Pull-up
LED1_STATUS = 0 'Background LEDs: 0=automatic, 1=manually
STATUS$     = "Modus: AUTOMATIK"
ADC_IN = 0:
```

section on the basics of the language. In **Listing 1** the start of the used BASIC script is presented. **Listing 2** contains the code of the deliberately simple web interface, which can be seen in **Figure 4**.

This project quickly resulted in about 140 lines of BASIC. Since they are provided with sufficient comments, you can follow such a script even if you didn't write it yourself or if you want to extend or improve it in a few weeks. The complete code is available for download on the Elektor website for this article [6].

Listing 2. The deliberately simple web interface.

```
WEB_PAGE:
cls
a$ = "<center><h1> - S A N D U H R - "+ VERSION$ +" - </h1>"
a$ = a$ + "<br>" + textbox$(t$,"cssTB")
a$ = a$ + METER$(SS,0,60,"cssMET")
a$ = a$ + textbox$(TEMP$,"cssTB")+"<br><br>"
a$ = a$ + "<br>R: "+ slider$(R, 0,255)+ textbox$(R,"cssTB")
a$ = a$ + "<br>G: "+ slider$(G, 0,255)+ textbox$(G,"cssTB")
a$ = a$ + "<br>B: "+ slider$(B, 0,255)+ textbox$(B,"cssTB")
a$ = a$ + "<br>"+ LED$(LED1_STATUS)
a$ = a$ + "<br>"+ textbox$(STATUS$)
a$ = a$ + "<br>"+ BUTTON$("Umschaltung auto/manuell",MAN_AUTO)
a$ = a$ + cssid$("cssTB"," width:70px;text-align:center")
a$ = a$ + cssid$("cssMET"," transform:rotate(-90deg);")
html a$
return
```

In the constant change from opening the file default.bas, editing, storing (!), starting, debugging, debugging, extending, saving, starting etc. common errors in the code are named and marked by the interpreter. They are then usually corrected in no time. The BASIC code can be readily understood by casual programmers if meaningful variable names are assigned and subroutines are used. BASIC is just easy to read. Automatic code highlighting improves readability and facilitates syntax checking. For syntax problems, the F2 key helps with its context-sensitive search in the online help file. There is also a small green check mark in the editor window that leads to the syntax check.

The BASIC script is initially quite linear. After initialization of some variables and a short movement of the hourglass servo, the IP address of the Annex web server is shown in portions on the display. This helps a lot when accessing the server with a web browser, especially if the router has assigned a different IP address.

The main routine is processed once per second by a timer. Hours and minutes are processed to suit the display, and the subroutines for light control and backlighting, the display of time and temperature, and every ten minutes the servo control are called sequentially. I abandoned my original plan to turn the hourglass once per minute when I noticed how loud a model-making servo is when it turns the hourglass in the living room. So I decided to use a ten-minute hourglass, which turns the glass slowly over approximately four seconds and thus more quietly and less frequently. If you like that, you can also use a faster hourglass to achieve a one- or five-minute rhythm via the variable `TURN_TIME`.

Web interface

When a web browser first accesses or re-accesses the ESP8266 web page, or when variable values used change, the HTML content of the ESP8266 web page is prepared and updated in the `WEB_PAGE` subroutine (see Listing 2). The web interface shows the current time and temperature. The simple LED background animation behind the hourglass is replaced by a manual RGB colour selec-

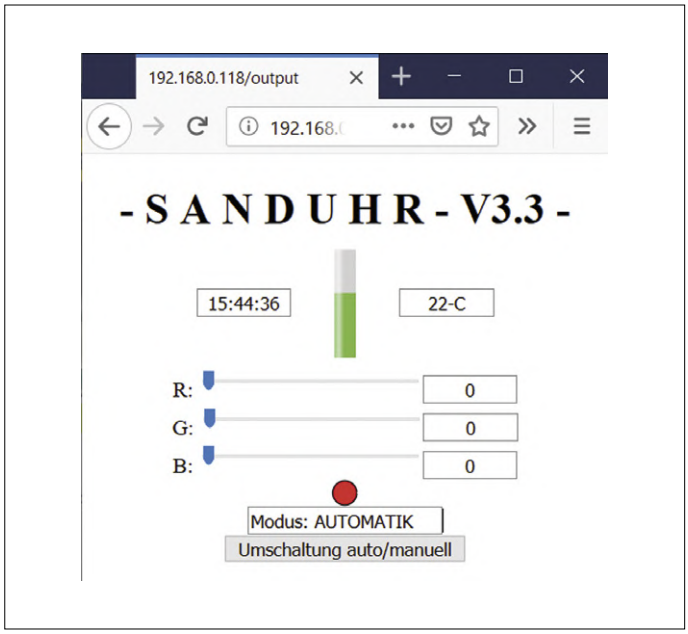


Figure 4: A simple web interface can be realized with a whiff of BASIC code. *Sanduhr* = hourglass; *Modus* = mode; *Umschaltung* = toggle.

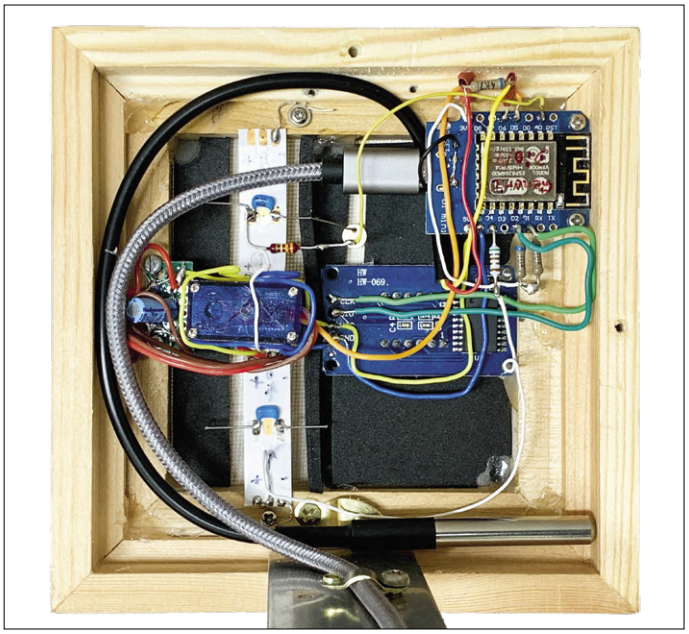


Figure 5: This is how the electronic hourglass looks from behind: not pretty, but working, assembled in flying-wire style.

tion via pushbutton. In automatic mode, the RGB controllers and display fields follow the continuously calculated values of the animation live.

The website is only intended to demonstrate here by way of example how the Annex firmware in the background handles the bidirectional communication with the web browser. The Annex-supported use of CSS (Javascript, AJAX and web sockets are also supported) means that there are virtually no limits to experiments in communication with (almost) any web browser. There are also many examples of this on the Annex website — especially in the files that can be copied from the Annex Toolkit to the */program* directory on the ESP module during firmware installation. It's worth browsing! An interesting script can be inserted, saved and started quickly in *default.bas* in the Annex Editor window with a text editor (e.g. *GEANY.exe*) for testing via "crtl+a" (select all) as well as copy & paste! Complete sections of the script can be quickly converted into comment lines from the menu and vice versa.

Moving to the final housing

The tested circuit was finally installed without any circuit board or breadboard in a small wooden frame, covered with a translucent canvas on the front side Figure 5). Thus only the active, luminous LED display segments are visible and the LDR light sensor can also absorb sufficient ambient light. The servo axis for the hourglass was guided through a small hole in the screen. With appropriately cut sections of foam rubber plates or black paper the light transmission of the front material is limited to the necessary zones on the display, behind the hourglass and on the LDR.

Other frames with plexiglass fronts (e.g. from the range of a large take-away furniture store) could also be used as an alternative. The transparent front can then be backed on the inside with a transparent paper or a suitable photo print. ⏮

190400-B-04

4 SALE @ WWW.ELEKTOR.COM

→ **WeMos D1 mini Pro**
www.elektor.com/wemos-d1-mini-pro-esp8266-based-wifi-module

→ **Book: IoT Home Hacks with ESP8266**
www.elektor.com/iot-home-hacks-with-esp8266

→ **PDF file: ESP32 & ESP8266 Compilation**
www.elektor.com/esp32-esp8266-kompilation-en

Web Links

[1] BASIC for the ESP32 and ESP8266 (1): www.elektormagazine.com/190400-03

[2] Annex WiFi RDS: https://sites.google.com/site/annexwifi/home

[3] Example projects: https://sites.google.com/site/annexwifi/projects

[4] Toolkit: https://sites.google.com/site/annexwifi/downloads

[5] Manual: https://sites.google.com/site/annexwifi/home/first-steps

[6] Article web page: www.elektormagazine.com/190400-B-04

Developer's Zone

Tips & Tricks, Best Practices and Other Useful Information

By **Clemens Valens** (Elektor Labs)

FROM IDEA TO PRODUCT

PART 5

Even though it seems that our new product design advances rather slowly, it's unlikely to move quicker in the real world. Setting up a business takes time. The previous installment left us with a Bill of Materials (BoM) and a netlist, and so we can now start working on a printed circuit board (PCB) for the project. Or several, as needed. Please note that PCB designing is highly complex and cannot be treated in any depth in this short article. Please refer to [1] for more information on, and suggestions for, PCB design work.

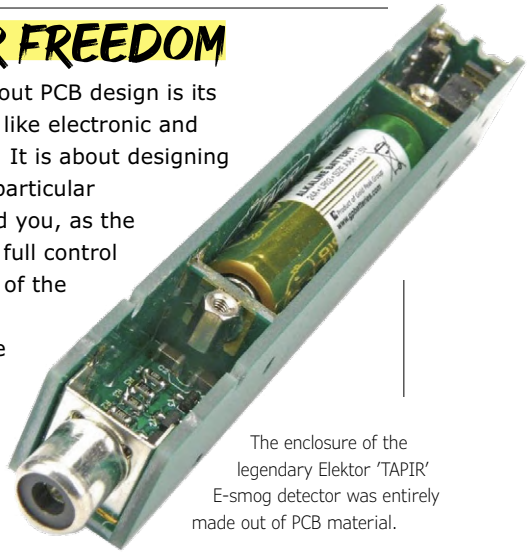
START AT THE END

One of the first things we did in this article series was creating a mock-up of our future product. With this dummy in mind, we designed the electronic circuitry to put into it, and consequently we only selected components that fit inside and are adapted to the product's intended working environment. Now we are going to add one or more PCBs to the mix, and they will have to fit as well. Therefore, before you start placing parts and routing traces you should think about the board's outline and mounting holes, or other ways of keeping the PCB in place inside its enclosure.

Often people simply start with a rectangular outline, place the parts as closely to each other as the routing allows, and then shrink the outline to get the smallest board as possible. Next, the board is etched, populated and tested and finally put into the enclosure. It then turns out that there either are no or insufficient mounting holes, or they are not in the right place, and not of the correct diameter. Furthermore, connectors are at the wrong side and cooling is not possible. In short, you created a mechanical nightmare, and the board is up for revision. Most of this could have been avoided if the product's future enclosure had been on the desk of the PCB designer.

ENJOY YOUR FREEDOM

The interesting thing about PCB design is its combining of disciplines like electronic and mechanical engineering. It is about designing a mechanical part with particular electronic properties and you, as the designer, have (almost) full control over it. So do make use of the possibilities PCB design offers you. A PCB can be part of the enclosure, it can be made from aluminium, it can be flexible, it can have almost any shape. Because the boards you see in Elektor are often rectangular doesn't mean that they must be like that.



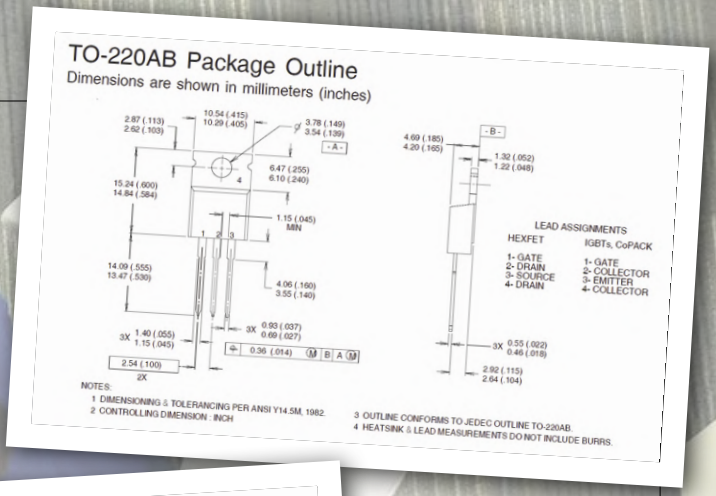
The enclosure of the legendary Elektor 'TAPIR' E-smog detector was entirely made out of PCB material.

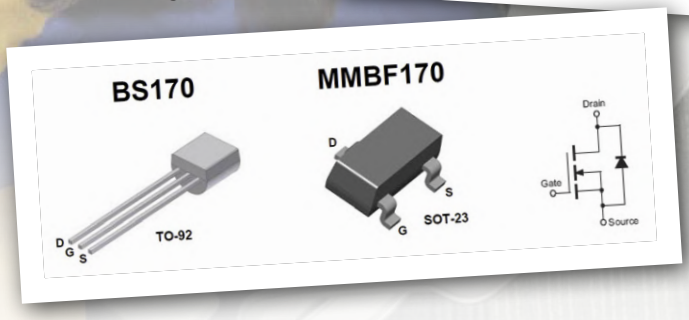
DESIGN FOR PRODUCTION

It's tempting to build in safety margins when designing a footprint; you know, make a pad slightly larger than specified, a hole a bit wider, that sort of thing. This sounds like a good idea, but it may very well result in problems during automated manufacturing. Oversized SMD pads can make parts drift out of position or rotate during reflow soldering. Or the excess solder paste may cause short circuits. Holes with a diameter much larger than the lead passing through it may not wave-solder properly.

CAN YOU TRUST YOUR PARTS?

Besides ignoring mechanical compatibility, another very common PCB design problem are wrong footprints. The problem is similar to the one we came across during schematic capture. If you stick to passive components in 0805 packages and use SOT-23 transistors and diodes only you will probably be okay, but chances are that you will have to design at least one custom footprint because your CAD software doesn't have it in its libraries. Repeating the wise words from the previous installment: "*Searching the Internet may be a quick solution, but can you trust clemo12345? Buying them from a commercial schematic/PCB CAD part supplier may be better, but will cost money. In any case, you should carefully check every component before dreaming of relying on it. This is also true for standard library parts as libraries may contain errors.*"





It's surprising how confusing datasheets can be, when it could have been so limpid.

Besides downloading the datasheet of every part of your design, it is also highly recommended to get samples of each and every one of them before commencing the board design. It is surprising how unclear datasheets can be about diameters (pin or hole?), bottom or top view, inches or millimetres, pin numbering, etc. And they may contain errors too. Having the real thing in your hand will save a lot of mistakes.

TALK TO THE ASSEMBLY HOUSE

Besides soldering problems due to incorrect footprints, component placement may also hinder the assembly process. A large metal part next to a tiny plastic one may result in bad soldering of the first, or damaging the second due to the uneven heat requirements. Part positions may not be compatible with the pick and place machine, requiring, for instance, the addition of snap-off tabs to the board. There are also parts that have special soldering requirements. Most of these problems can be solved by the people at the PCB assembly house, but if you consult them before finalizing the PCB, the board may turn out cheaper and faster to produce, as well as more reliable.

BOARD ITERATIONS ARE EXPENSIVE

Chances are high that the first version of your PCB will need a few tweaks. To allow you to keep track of different board iterations, mark and label them clearly. Do not forget to update the version number on the board every time you change it. It's also recommended to write down the changes and their causes. Volume quantities tend to increase as confidence in the product grows. This makes the product cheaper, but it makes modifications more perilous. Discarding ten prototypes is probably less expensive than replacing one resistor with a wrong value on 1,000 boards pulled from a production run. No longer remembering exactly why that resistor value was changed doesn't make things any easier.

Web Link

[1] PCB Design is Not Easy: www.elektormagazine.com/160397

BEWARE OF MIRRORED FOOTPRINTS!

The mirrored-footprint error is so common that it merits having its own paragraph header. Explaining it is not necessary, you know what I mean; just stop making this mistake. End of paragraph.

WHAT'S NEXT?

Our product is almost ready, or is it? We now have a few prototypes to test and to play with. Testing, you said? Yup. Testing for functionality, for certification, for quality — there is a lot of testing to do. So how do you do that? And what are the implications? Will this article series never end? ⏮

190369-C-01

Cricket Thermometer and Dolbear's Law



At the risk of repeating myself, temperature may well be the world's most surveyed physical quantity. Several methods exist to measure temperature. In electronics the most common ways to do it are:

- **thermistor**: a device whose resistance increases (positive temperature coefficient or PTC) or decreases (NTC) with temperature;
- **Resistance Temperature Detector (RTD)**: exploits the precisely known relationship between temperature and resistance of a material, typically a metal like platinum;
- **thermocouple**: a junction of two different metals that produces a temperature-dependent voltage as a result of the thermoelectric (Seebeck) effect;
- **semiconductor-based sensors**: the voltage across a diode or PN junction decreases by approximately 1 to 2 mV per degree Celsius;
- **infrared thermometer**: infers temperature from thermal radiation.

The last technique is often used to measure human body temperature by pointing a 'temperature gun' at the insides of an ear. Up to this point nothing new, you say, discreetly suppressing a yawn, but did you know that you can use the ear itself to determine the temperature?

As it turns out, the chirping sound produced by a cricket is temperature dependent. Similar to estimating the distance of a thunderstorm by counting the seconds between a lightning flash and the thunder that follows it, the temperature of the air can be estimated by counting the number of chirps produced by a cricket in one minute. This correlation was reported in 1881 by Margarette W. Brooks, but the credit went to Amos Dolbear after he published an article entitled "The Cricket as a Thermometer" in 1897. The green tree cricket officially known as *Oecanthus fultoni* has been known as the thermometer cricket ever since.

Calculating the temperature T from the cricket chirp rate is easy, especially in Fahrenheit:

$$T = N_{15} + 40\ [°F]$$

Here N15 is the number of chirps in 15 seconds. Converted (and rounded) to Celsius gives:

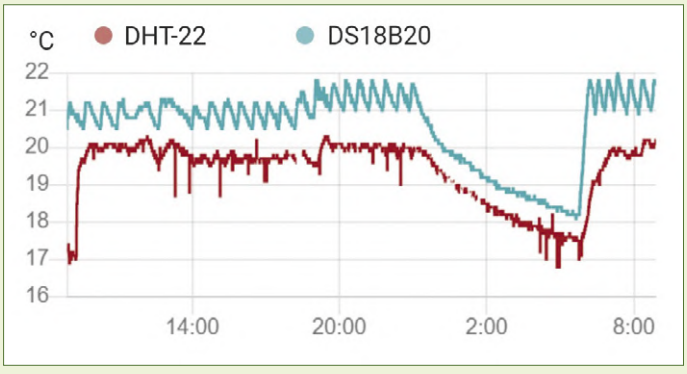
$$T = N_{8} + 5\ [°C]$$

Here N_8 is the number of chirps in 8 seconds. The cricket thermometer is fairly accurate between 5 and 30°C.

Cricket Emulator

Just for fun, using an Arduino Uno, a passive buzzer and a DS18B20 one-wire temperature sensor, I inverted the process to emulate a cricket based on the ambient temperature. Now this looks like a silly project and, to be honest, that's how it started. However, once I had it up and running, I quickly noticed how sensitive the human ear is to rate *changes*. In the end I think this emulator could make for a useful temperature indicator in applications where it is not practical to keep an eye on a display.

Full project details can be found at: www.elektor-labs.com/3945

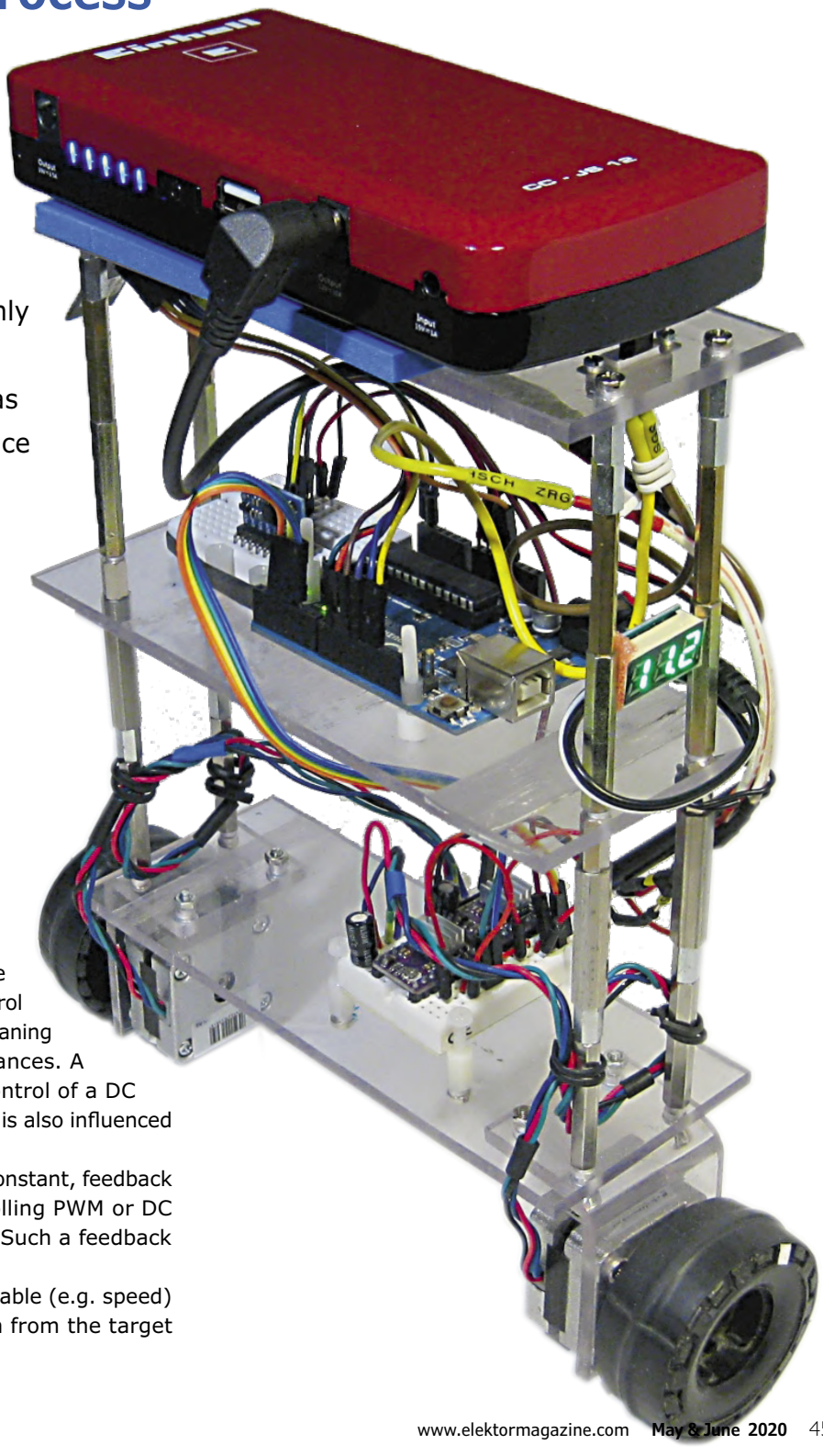


Not all temperature sensors are created equal. This 24-hour log shows the outputs from a DHT-22 and a DS18B20 sensor placed next to each other.

HOMELAB PROJECT

BalBot
a Self-Balancing Robot
Modern acceleration sensors ease the construction process

By **Dr Günter Spanner** (Germany)



Regulations are not only prevalent in technology, but are also commonly found in nature. The construction of self-balancing robots can serve as a preliminary stage of androids, since movement on two parallel wheels requires a control similar to that of the walking upright. In this project, an Arduino Uno and an IC comprising an acceleration sensor and a gyroscope are employed.

In contrast to technology and science, in everyday life a distinction is rarely made between regulation and control. A mere control system does not monitor the output value, meaning that it can change due to external disturbances. A simple and typical example is the speed control of a DC motor by PWM. Here, the speed of the motor is also influenced by load fluctuations.

Now to the control: If a speed is to be kept constant, feedback is required, for example to adjust a controlling PWM or DC voltage depending on the speed deviation. Such a feedback system characterizes a control loop.

Control therefore means that the output variable (e.g. speed) is captured and, in the event of a deviation from the target

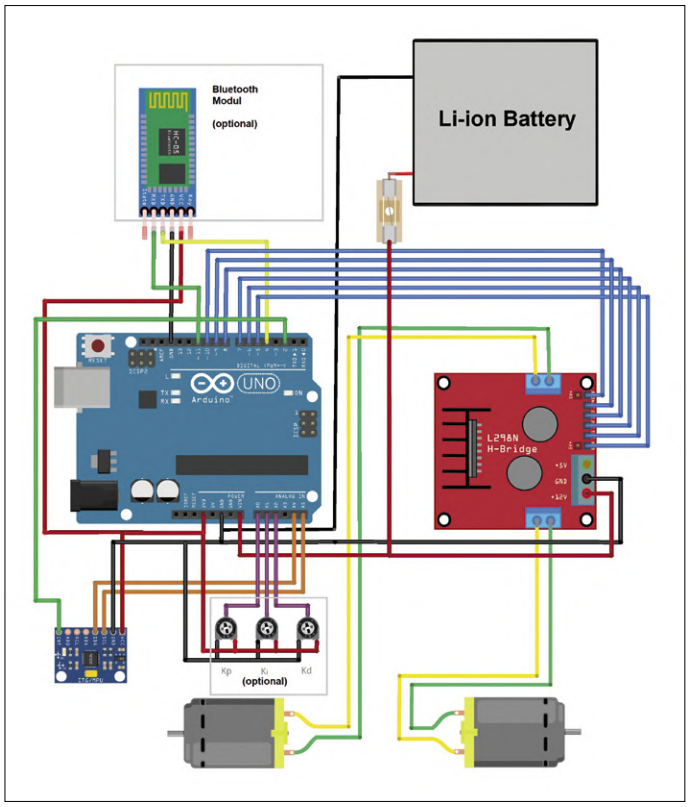


Figure 1. Wiring diagram of the BalBot. The Bluetooth module and the potentiometer are optional.

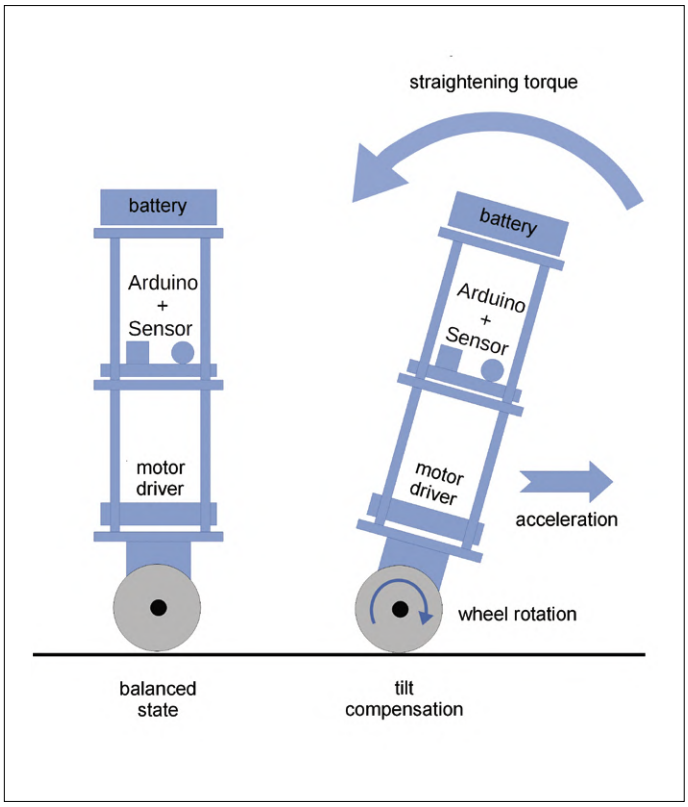


Figure 2. Principle of self-balancing.

COMPONENT LIST

Arduino Uno
L298 Motor driver (or module)
MPU6050
2 DC motors, e.g. PGM37DC12 or similar
Batteries (7 pcs NiMH or 2 pcs LiPo)
2-amp fuse
If applicable: power supply switch

value, the manipulated variable (here the PWM) is changed accordingly. The central terms are:

- Actual value: *x*
- Required value: *w*
- Control deviation: *e* = *w* - *x*
- Manipulated variable: *y*
- Disturbance variable: *z*

These variables are sufficient to describe a controller.
There are also controller types with different control characteristics. The *PID controller* is very common and covers many cases. It has a proportional (*P*) as well as an integral (*I*) and a differential (*D*) path. By adjusting its parameters, many control engineering problems can be solved with high precision.

Balance on one axis

An interesting example for the application of such controls is a self-balancing robot. In principle, this is an inverse pendulum on wheels. Due to its instability, the inverse pendulum is a classic example for the application of an active control loop. The well-known Segway scooter is a one-axle, two-wheeled, electric vehicle. The person to be transported stands between the wheels virtually on the axle. In principle, single-axle vehicles are unstable. So a control system provides the necessary balance. Driving it is fun. The popularity of such scooters then triggered the development of self-balancing robots.

To reproduce a scooter according to the Segway principle is not really difficult. There are even complete kits. However, it does not necessarily have to be such a scooter to deal with the underlying control technology: For this purpose, it is sufficient to build a smaller, self-balancing robot, as described below.

In order for a self-balancing robot to maintain its balance, its wheels must permanently counteract the tendency to tilt. This requires a dedicated control loop, a sensor and actuators. The IC MPU-6050 is perfectly suited as a sensor element, containing an acceleration sensor and a gyroscope. This allows very precise measurement of acceleration and rotation in the three three-dimensional axes. Since such chips are used in many devices such as cameras or smartphones, mass production makes them low in price. An Arduino Uno is all you need to calculate the data. The motors coupled to the wheels are the actuators.

If the MPU6050 is connected to an Arduino (see bottom left in **Figure 1**), the correct function can be checked with a test program. The download archive accompanying this article [1] contains „MPU6050_test“ for this purpose. After loading the test program the serial monitor of the Arduino IDE can be started. Six lines show the reaction of the sensor values to rotation or movement. If correct motion data is displayed, the motor driver can be connected to the Arduino. The type L298N (the IC or a module equipped with it) is appropriate as driver for small DC motors. The power source is either seven 1.2 V NiMH (= 8.4 V) or two 3.7 V LiPo batteries (= 7.4 V). Figure 1 shows how the individual components and modules (see parts list) are interconnected.

Balancing algorithms

As already mentioned, a suitable control loop is required. The position of the robot corresponds to the actual size. It should deviate as little as possible from the vertical (target size). A PID controller is ideal for this. Our robot named BalBot achieves an astoundingly stable dynamic balance with optimally adjusted K_p, K_i and K_d parameters. The target value or the vertical is first fixed in the software. **Figure 2** illustrates how this works in practice.
The MPU6050 module permanently records the current position of the robot. Its values are processed by the PID algorithm in Arduino. The control sketch carries out the calculations for controlling the motors. If the robot is in danger of tipping over in one direction, the resulting tilting moment is compensated by a corresponding compensation movement of the wheels. The robot remains vertical even if it is pushed slightly. The corresponding sketch BalBot.ino is also included in the free download archive.

Its code is divided into six parts:

1. Initially the libraries are loaded. If not included in the Arduino IDE by default, they can be got via Internet [2] [3]. Then the motor connections are specified. The motors can be operated both forward and reverse by means of EN (enable) and IN (input). The motor power is controlled by PWM.
2. The values required for operating the MPU6050 are subsequently defined. These are essentially the standard settings. Details can be found in the IC datasheet [4].
3. Now the PID parameters are specified. They are of central importance. The following three values define the control behaviour: K_p (range 0-1000; default value 400), K_d (range 0-100; default value 30) and K_i (range 0-500; default value 200).
4. These values must be adapted to the real conditions of the respective hardware. Differences in mechanical construction (due to differences in motors, wheel sizes, etc.) can be compensated here. Notes on the adjustment of these parameters can be found in the next section.
5. The required interfaces are now activated in the setup. The serial interface is only used to output debug information. This code can be omitted in the ’final release’. The MPU6050 module is then started and checked for function - an error message is output if necessary. The specified offset values can usually be accepted without modification, since the gyroscope automatically calibrates itself.

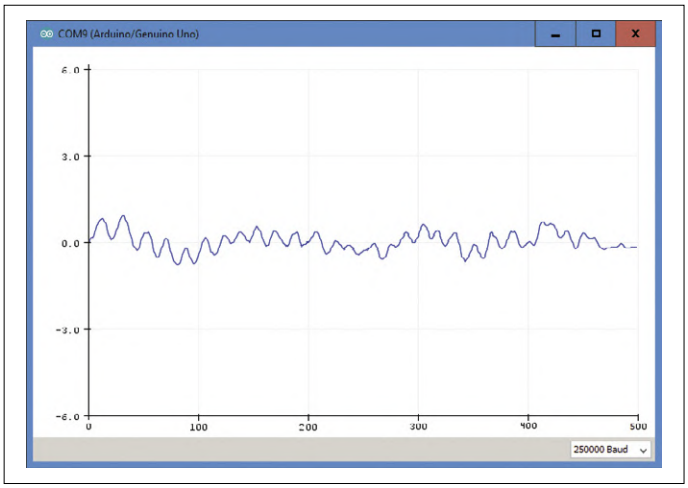


Figure 3: Tilt angle curve in the serial plotter integrated in the Arduino IDE.

6. In the main loop, only the MPU signal is read out, and the control signal for the motors is computed using the PID controller pid.Compute() and forwarded to the motor driver with the following call:

```
motorController.move(PIDout, MIN_ABS_SPEED);
```

In addition a check is carried out to determine whether the FIFO buffer of the MPU6050 is overflowing. This should not happen during normal operation. If necessary, the buffer size can be adapted.
The line:

```
Serial.println(ypr[1] * 180/M_PI);  // actual tilt
```

outputs the tilt angle. In the Arduino IDE serial plotter it can be tracked continuously. If regular, sinusoidal value curves appear, that is a clear indication for the oscillation of the control. In this case the control parameters need to be adjusted. In most cases it is helpful to reduce the Kp value. Ideally, the line should only show small, irregular signal deflections, such as those caused by random disturbances (air movements or uneven floors). **Figure 3** shows the random compensating movements around the rest position.

Parameter optimization and adjustment

The optimization of the PID values could also be done by means of a simulation for example with MATLAB. In the professional field, this procedure is standard for complex controllers.
But with BalBot, the classic ’trial and error’ approach can also be used. Here, the individual parameters are gradually adjusted until the best possible control is achieved:

1. First, the three parameters K_p, K_i and K_d are set to zero.
2. Next, K_p is slowly increased step by step. If K_p is too low, the robot will tip over because the corrections are not fast or large enough. If K_p is too high, the robot will perform violent back and forth movements without reaching equilibrium. K_p must be adjusted so that the robot moves back and forth only slightly.

3. Upright standing with balancing movements is now achieved. Often a certain tendency to oscillate still occurs: the robot quickly oscillates or quivers around the ideal equilibrium position.
4. Now the K_d value is adjusted in a way to reduce the oscillations. In the ideal case, the robot now remains in an upright position almost motionless. If K_p and K_d are properly adjusted, the robot will remain upright even if it is lightly bumped.
5. Finally, the K_i parameter is set. Even with optimum K_p, a certain tendency to oscillate still occurs in the event of major disturbances. It can even lose its balance. K_i is then ideally set when position stabilization occurs quickly after a disturbance.

If all three parameters are optimally set, the robot snaps into a nearly vertical position as soon as the controller's catch range is reached. It then stands upright as if held by a ghost hand and compensates for even larger disturbing influences with ease. If no balance can be achieved, the motor connections may have been mixed up. In this case, the robot does not accelerate in the direction of the tilting moment, but in the opposite direction. **Figure 4** shows the finished robot in an upright, self-balancing position.

Since setting the parameters by software is quite complicated, there is also a simpler variant. An Arduino has several analogue inputs. Three potentiometers can be connected to these, allowing the values for P, I and D to be set directly. The software only has to contain the query for the potentiometer values (see Listing: **Potentiometers**).

'Potentiometers' program listing

```
void readPIDTuningValues()
{   int potKp = analogRead(A0);
    int potKi = analogRead(A1);
    int potKd = analogRead(A2);
    kp = map(potKp, 0, 1023, MIN_P, MAX_P) / 100.0;
    ki = map(potKi, 0, 1023, MIN_I, MAX_I) / 100.0;
    kd = map(potKd, 0, 1023, MIN_D, MAX_D) / 100.0;
}
```

'Bluetooth' program listing

```
void readPIDTuningValues()
{   int potKp = analogRead(A0);
    int potKi = analogRead(A1);
    int potKd = analogRead(A2);
    kp = map(potKp, 0, 1023, MIN_P, MAX_P) / 100.0;
    ki = map(potKi, 0, 1023, MIN_I, MAX_I) / 100.0;
    kd = map(potKd, 0, 1023, MIN_D, MAX_D) / 100.0;
}

content=blue.readString();

#if LOG_BT
     Serial.print("content: ");Serial.print(content);
     Serial.print(" - setpoint: "); Serial.print(setpoint);
     Serial.print(" - rotation: "); Serial.println(rotate);
#endif

if(content[0]=='F')
       // forward
else if(content[0]=='B')
       // backward
else if(content[0]=='L')
       // left
else if(content[0]=='R')
       // right
```

So the parameters can be optimized during operation via the potentiometers (see *BalBot_Pots_BT.ino* in the download). The connection of the optional potentiometers is already considered in Figure 1.

Remote control by smartphone

Building a self-balancing robot is an interesting learning challenge. If it all works, this initially pleasing result is no longer an exciting one in the long run. A remote control could change that. It would allow BalBot to be moved wirelessly in all directions. Since nearly everyone (including the electronics engineer) has a smartphone with a Bluetooth interface and many of them run under Android, such a smartphone can be used to control the robot.

All that is needed is to hook up a Bluetooth module to the Arduino. An HC05/06 receiver, which is already included in Figure 1, is well suited.

To use it, the standard Bluetooth library of the Arduino IDE must be integrated and the sketch must be supplemented accordingly (**Bluetooth listing**).

This extension is also already included in the sketch BalBot_Pots_BT.ino. The smartphone only needs the app „Arduino Bluetooth" from Circuit Magic. The app can be downloaded for free from the Playstore [5]. After installation and connection with the HC05/06 module the

app sends the necessary control commands to the Arduino. **Figure 5** shows the user interface of the app. The app is controlled using the arrow keys on the left side of the screen.

Conclusion & Outlook

The robot presented here can be assembled with a minimum of effort. Besides the drive motors only an Arduino, a motor driver and an acceleration/gyroscope module are required. The adjustment of the control parameters is considerably simplified by additional potentiometers. BalBot can even be moved remotely within your own four walls via Bluetooth.

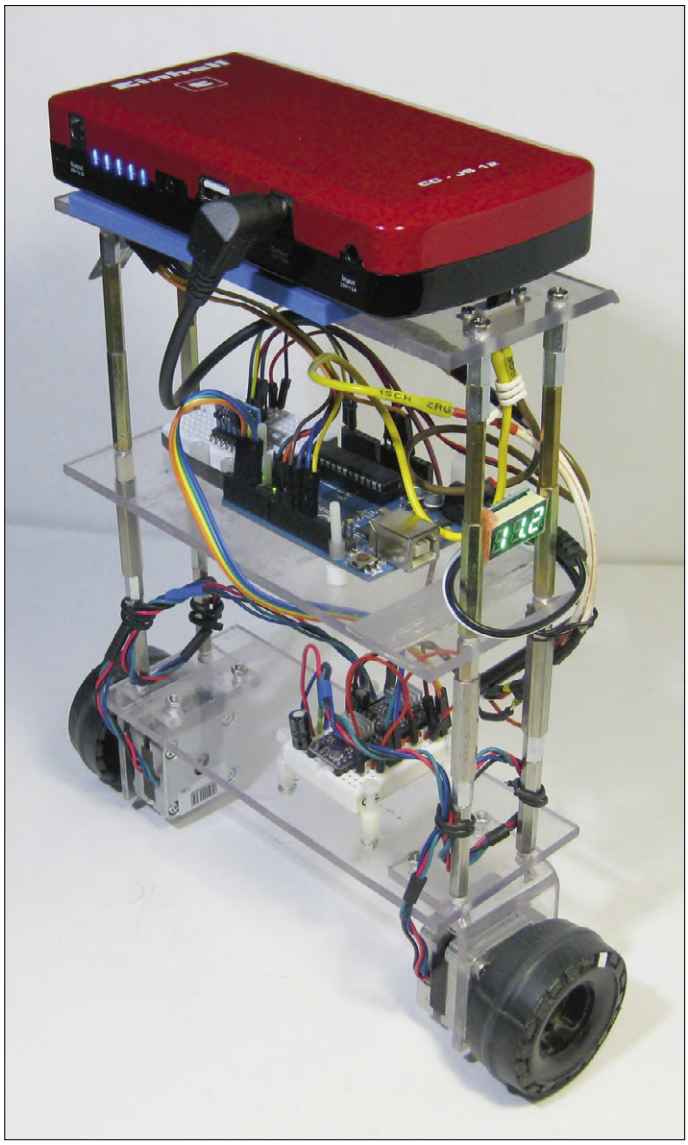


Figure 4: BalBot in action.

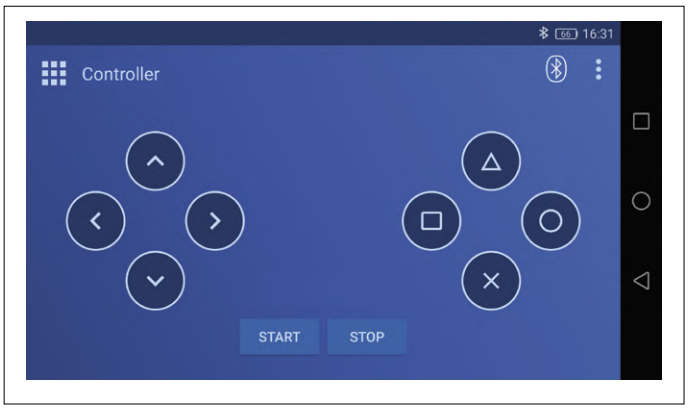


Figure 5: Android app for controlling BalBot via Bluetooth.

A further improvement of balance and handling, especially in remote-controlled operation, could be achieved by so-called rotary encoders. This allows the speed of rotation of the wheels to be recorded. This information can be used to further improve stability when driving forwards and backwards. In practice, the remote-controlled BalBot occasionally loses its balance if only the values of the MPU6050 are available. Unfortunately, this sensor can only measure acceleration and rotation, but not the speed directly.

Another alternative is to use stepper motors. Counting the steps provides information similar to that obtained by encoders. Besides replacing the motors, the L298 must of course be replaced by suitable stepper drivers (such as the DRV8825 driver). ⏮

200074-02

Web Links

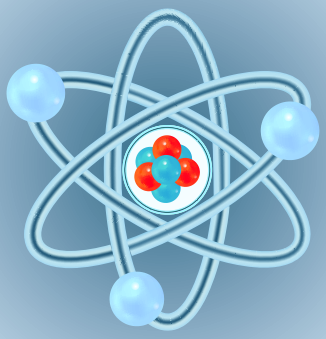
[1] Article support page: www.elektormagazine.com/200074-02

[2] PID_v1.h: https://github.com/br3ttb/Arduino-PID-Library/blob/master/PID_v1.h

[3] MPU6050_6Axis_MotionApps20.h: https://bit.ly/37T7T1c

[4] MPU6050 datasheet: www.invensense.com/products/motion-tracking/6-axis/mpu-6050/

[5] 'Arduino Bluetooth' app: https://play.google.com/store/apps/details?id=com.circuitmagic.arduinobluetooth&hl=en



Starting Out in Electronics (3)

Easier than imagined!

By **Eric Bogers** (Elektor Netherlands)

In the previous episode we got acquainted with Ohm's Law and did some elementary calculations. Now it gets a bit more complicated — but no fear: we don't need higher mathematics...

First a small update to the previous episode. There, we mentioned that a decimal prefix is often written instead of the decimal point: 8.2 kΩ becomes 8kΩ2 and 5.6 nF becomes 5nF6. There we noted casually that the µ ('micro' or 10^{-6}) is at least tricky to find on our US/UK keyboards. In fact the key combination is ALT+0181, but that is clumsy. That's why instead of the µ, a 'u' is often written and then we get: 4.7 µF = 4.7 uF = 4uF7 or even 4u7.

Series connection of resistors

Over to business as usual. What happens when we connect two resistors in series, as outlined in **Figure 1**? If we consider a resistor as a component that impedes the current flow, we could reason as follows: the current is obstructed by R1, and the small amount of current that manages to pass that resistor is obstructed again by R2. So it is obvious that the total resistance increases when connecting resistors in series. Yes we know, this reasoning is rickety but it does make some sense of what is happening.

Now let's see exactly what happens. Thanks to Ohm's Law, we know that the voltage *U* across the series connection of R1 and R2 (which we call *R*) equals

$U = R\,I$

Now resistors are not mysterious components — no power is lost in them. So the current *I* in the series circuit is equal to the current leaving the series circuit, and so both resistors pass the same current. Each of the resistors drops a certain voltage (which in turn follows from Ohm's Law), and because resistors are purely passive components, no voltage is added along the way (and no voltage disappears either), so the sum of the two partial voltages *U*1 and *U*2 is equal to the total voltage *U*. Armed with that knowledge, we can write:

$U1 = R1\,I$

$U2 = R2\,I$

And since $U1 + U2 = U$:

$U = (R1\,I) + (R2\,I) = (R1 + R2)\,I$

And it follows that the total resistance *R* equals the sum of resistors R1 and R2. Okay so you had guessed that and it's obvious too, but of course it's nice to know for sure.

We leave it up to the reader to check the above by means of a numerical example (e.g. take a battery voltage of 15 V, R1 = 10 Ω and R2 = 20 Ω).

Parallel connection of resistors

Resistors are not only connected in series ('sequentially') but also in parallel ('side by side'), as outlined in **Figure 2**. First eliminate R2 for a moment; then we know that a certain current will flow through R1. Now we add R2; a certain current will flow through it as well. Therefore, when connected in parallel, the total current increases; and since the voltage across the resistors remains the same, that means the total resistance has become smaller!

Looking at Figure 2 again, two things are clear: both resistors R1 and R2 have the same voltage *U* across them. Consequently, the sum of the currents through the individual resistors is equal to the total current *I*: we are dealing with purely passive components, so no current disappears or gets added along the way. If we call the total resistance of the parallel circuit *R* again, we get:

$U = R\,I$

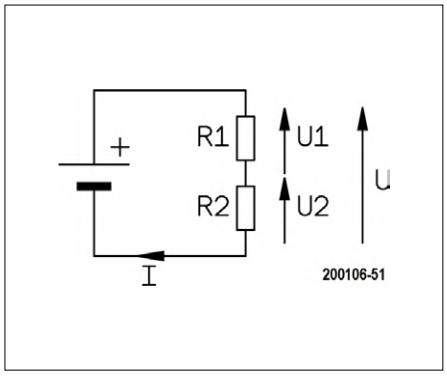


Figure 1: Series connection of two resistors.

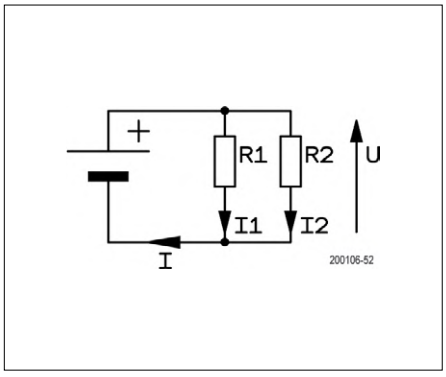


Figure 2: Parallel connection of two resistors.

Now we apply Ohm's Law to the individual resistors:

U = R1 $I1$ $I1 = U$ / R1

U = R2 $I2$ $I2 = U$ / R2

With $I1 + I2 = I$, we write

$I = (U / R1) + (U / R2) = U / R$

and from that:

$1 / R = (1 / R1) + (1 / R2)$

Rearranging:

$R = 1 / [(1 / R1) + (1 / R2)]$

We leave it up to the attentive reader to check this outcome by means of a numerical example.

Mixed circuit

In daily electronics practice we are not only dealing with 'pure' series and parallel circuits, but also with 'mixed circuits' such as those of Figure 3. There we see a series connection of a single resistor (R1) with a parallel connection of two resistors (R2||R3). (Note: a parallel circuit is often indicated by two parallel bars ||). Let's look at a numerical example here: R1 = 10 Ω, R2 = 20 Ω and R3 = 30 Ω. Although it is possible to write down a single relationship for the total resistance, such an approach is likely to incur mistakes and arithmetical errors. It is better to calculate the total resistor in two steps: first the parallel circuit and then the series circuit. For the parallel circuit of R2 and R3 we write:

$1 / R_{parallel} = (1 / R2) + (1 / R3)$

Entering the example value yields:

$1 / R_{parallel} = (1 / 20\ \Omega) + (1 / 30\ \Omega) = (3 / 60\ \Omega) + (2 / 60\ \Omega) = 5 / 60\ \Omega$

In reverse: $R_{parallel} = 60\ \Omega / 5 = 12\ \Omega$

This resistor is connected in series with R1; for the total resistance R of this mixed circuit, we arrive at $R = 10\ \Omega + 12\ \Omega = 22\ \Omega$.

A few remarks

As you can now easily check for yourself, the following rules of thumb apply for guesstimates and quick estimates:

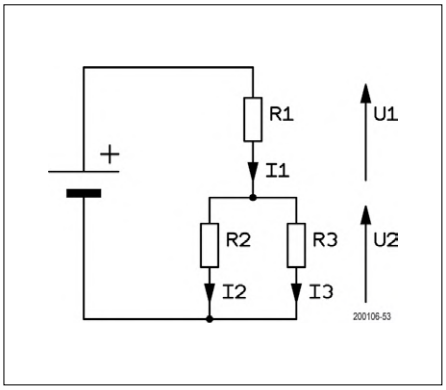


Figure 3: Simple mixed circuit.

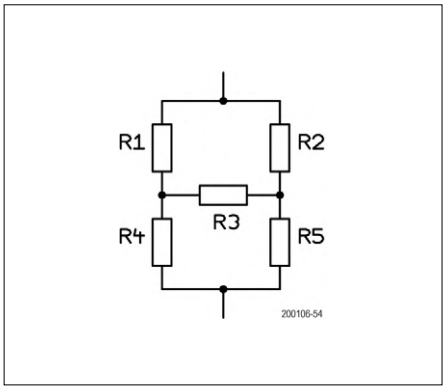


Figure 4: The H-circuit, a dreaded occurence in exams.

- With series connected resistors, the total resistance is always greater than the largest of the series connected resistors;
- with parallel connected resistors, the total resistance is always smaller than the smallest of the parallel connected resistors.

For example, if we have a series connection of a very large and a very small resistor (say 100 kΩ and 10 Ω), then in everyday electronics practice we don't calculate that exactly, but just say that the total resistance equals the largest resistance, so 100 kΩ. And with parallel connection of the same resistors the same holds: we save ourselves the trouble of calculation and say that the total resistance equals the smallest resistance, so 10 Ω. The difference with the actual values is so small that we shouldn't have to worry about that. And another remark close to the end: above we've limited ourselves to series and parallel connection of two resistors each, but mutatis mutandis everything we've reasoned and calculated also holds for series and parallel connection of more than two resistors:

- series connecton of n resistors: R_{series} = R1 + R2 + ... + Rn
- parallel connectioon of n resistors: $1 / R_{parallel}$ = (1 / R1) + (1 / R2) + ... (1 / Rn)

Back to Figure 2. As a result of it we noticed that the sum of the currents $I1$ and $I2$ equals the total current I. That seems very obvious, but in physics (and electronics is no different than applied physics) something is only obvious when it is irrefutable proven. And physicist Gustav Kirchhoff did that back in 1845. Kirchhoff's First Law states that the sum of the currents in a node equals the sum of the currents from that same node.

The H-circuit - a forward view

Armed with the knowledge we have now acquired, we can calculate simple mixed circuits such as the ones from Figure 3 without many problems. But take a look at **Figure 4** - a mixed circuit of the trickier kind. In this so-called H-circuit, we cannot find combinations that we can treat separately as series or parallel circuits. How can we do that? How do we deal with such a thing?
Think about this; in the next episode we will come back to this (keyword: star-delta transformation or Y-Delta transformation). And then we will also deal with alternating currents and voltages. ⏮

200106-02

Small Circuits Revival

Capita Selecta from the Elektor Project Suggestions Box

By **Eric Bogers** (Elektor Netherlands)

Small circuits not using SMD components or special tools, that can be built on stripboard and invite experimentation and improvisation – these are frequently a springboard to a career in electronics. Here we again present a selection from the Suggestion Box of your favourite magazine.

Idea: Michael A. Shustov (Russia) and Andrey M. Shustov (Germany)

Energy-efficient Light-sensitive Relay

In the first episode of this series (Elektor January & February 2020) we described a few variations on the subject of 'energy-efficient relay' from the father & son team Shustov. These are actually also variations of the well-known stairwell switch circuit. The Shustovs haven't been sitting idle: **Figure 1** shows a version that has been expanded with a light-sensitive transistor, so that the lamp is not only turned on by pressing one of the buttons SB1 through SBn, but can also be turned on from a distance using, for example, a torch.

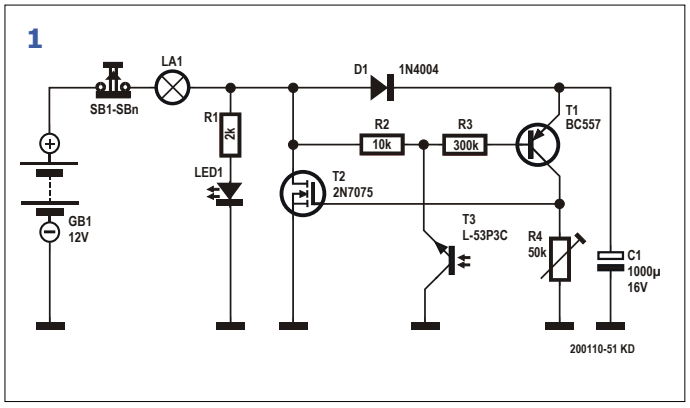
If we forget about T3 for the moment, then the circuit operates exactly as described in the first episode. In the rest state the series-connected pushbuttons SB1 through SBn are all closed (for the sake of simplicity only one is drawn). A small current can flow through the lamp via R1 and LED1 to ground — the LED will light up to indicate that the circuit is 'armed'. At the same time, capacitor C1 is charged via D1. By the way, the current through the lamp is not high enough for it to light up. (And it will be clear that this has to be an **incandescent** lamp).

As soon as a button is pushed there is no longer any current through the lamp and the base of transistor T1 is pulled to ground (via R3, R2, R1 and LED1). The transistor will start to conduct and C1 will discharge via R4. As a consequence, the voltage across R4 becomes high enough for T2 to turn on so that enough current will flow through the lamp for it to light up (of course once the pushbutton is released).

This state is stable for as long as the voltage across R4 is high enough for T2 to continue conducting. Once capacitor C1 has discharged sufficiently so that T2 turns off, the initial state will be restored; after a short time C1 is charged again via D1 and the circuit is armed once more.

Transistor T3 does actually the same thing as the pushbuttons: in the dark this light-sensitive transistor is off and the circuit is in the rest state. When light strikes the transistor (for example from a torch) it will switch on and pull the base of T1 to ground — and then the circuit behaves in exactly the same way as after pressing one of the buttons.

This method makes the circuit appropriate for application in a cellar, for example, where it is normally dark (!) — you do not have to search around for the light switch and only have to strike a match or something...



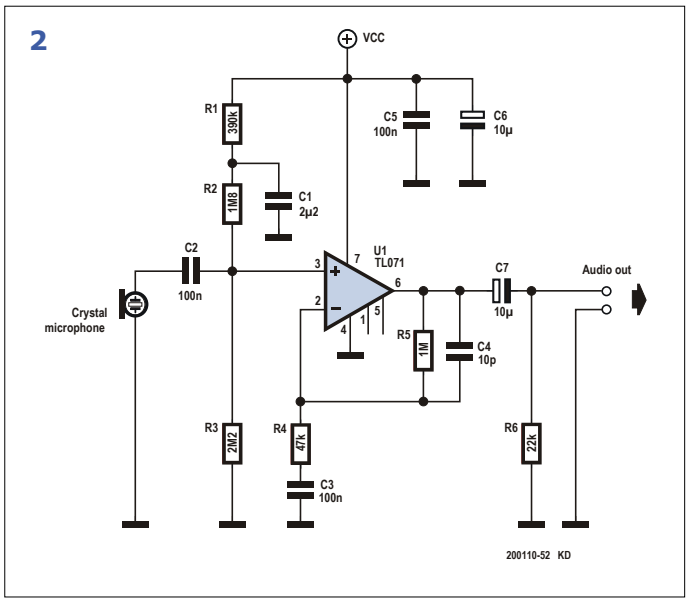


Idea: Joseph Kreutz (France)

Preamplifier for Crystal Pickup Elements

You will know them — the so-called 'stereo consoles' from the seventies of the previous century? These large wooden monsters with on one side a radio attached to an amplifier (no transistors, only tubes) with bulky mechanical selector pushbuttons and large volume and tuning knobs on either side of a beautiful 'multi-band' tuning scale. And on the other side a gramophone for 33, 45, and 78 rpm records?

That gramophone… Belt drive was a rarity, direct drive still



had to be invented. The drive used a rubber wheel between the motor shaft and the inner rim of the turntable. And of course this gramophone was equipped with a crystal element (sometimes called a ceramic element).

Such an element generated a relatively high output voltage, but had to be loaded with a high impedance. The latter was not a problem for the valve equipment of the time, of course. However, it is a problem now, because the 'phono' input of a modern amplifier (assuming that it still has one, in this CD era) has a normalized input impedance of 47 kΩ and that is much too low with all its consequences; furthermore, all modern phono amplifiers are designed for signal levels in the order of millivolts. This is guaranteed not to work well.

If, in spite of all this, you would like to experiment with old 'crystal gramophones', we need a preamplifier/impedance converter. Fortunately such a thing is easily realised, as **Figure 2** demonstrates. The heart of the circuit is a high-impedance FET opamp type LT071. With resistors R1 through R3 the input impedance is matched as good as possible to the microphone or element. The opamp is used to build a standard, non-inverting amplifier with a bandwidth that is limited by capacitors C3 and C4 from about 33 Hz to a little less than 16 kHz. The gain is about 21 times and is suitable for connecting to a CD input of a modern amplifier.

There is not much more to say about this simple circuit, except that it is not only suitable for crystal pick-up elements, but also for the crystal microphones that were often supplied with the tape recorders of that era (and is already indicated in the schematic).

Idea: Wolfgang Borst (Germany)

External Watchdog for Microcontroller Systems

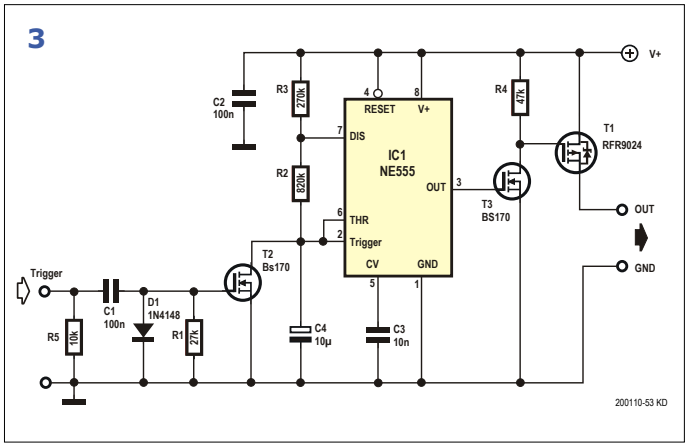
A microcontroller that runs 24 hours per day, 7 days per week, can latch up for whatever reason (note: *can*, it fortunately only happens rarely). In these cases, the normal reset signal generated by the internal watchdog may not be sufficient to escape from this situation. On these rare occasions only one thing will do: turn off the power supply voltage completely and then turn it back on.

And this is exactly what the small circuit of **Figure 3** does: it is a small external watchdog that, if it doesn't receive any trigger pulses from the microcontroller for about 10 seconds, will turn off the power supply voltage for about 5 seconds and then turn it on again.

The circuit is built around a 'good old' 555 — one of the oldest workhorses in electronics and still a favourite component when some timing is involved.

It is not difficult to understand the operation of the circuit: the regular trigger pulses from the microcontroller (note: the software that is running on that controller has to generate these, but that will not be a problem for any slightly accomplished programmer) will turn on transistor T2 each time, which results in the discharge of timing capacitor C4 of the 555. The 555 is configured as an astable multivibrator.

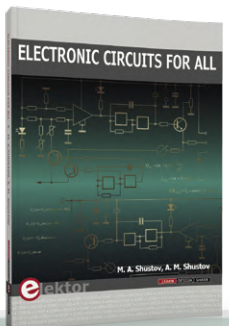
As a result the voltage across the capacitor cannot normally reach the trigger threshold necessary to activate the 555. The



output (pin 3) of the 555 remains high and the power supply voltage to the microcontroller is turned on via T3 and T1.

However, when the trigger pulses remain absent for more than 10 seconds, the voltage across C4 will be high enough for the 555 to toggle: the output becomes low and T1 will turn off via T3 and the power to the microcontroller is cut off. After about 5 seconds the 555 will switch back and the power supply is switched back on. ⏮

200110-02



4 SALE @ WWW.ELEKTOR.COM
→ Electronic Circuits For All
www.elektor.com/electronic-circuits-for-all

Tamper-Evident Box

Send data securely through postal services

By **Luka Matić** (Croatia)



In secure communication systems based on one-time pads (OTPs), messages are encrypted and decoded with a random key. To make this work, both the sender — usually called Alice — and the receiver — better known as Bob — need to use the same key. Alice generates the key, but how can she send it to Bob in a secure way? It is this chicken-and-egg problem that the project described in this article is trying to solve.

Quick Specifications

- Open source design without hidden tricks
- Capacitive box-opened detection
- Incident light box-opened detection
- Travel time verification
- Temperature attack detection
- Infrared communication
- Reusable

This project is a (logical) sequel to my TRNG [1] and OTP Crypto Shield [2] and tries to solve the problem of securely exchanging one-time pad (OTP) keys stored on, for instance, an SD card. I started out with the idea of designing a tamper-proof device, but then learned that securely erasing data from a flash-memory-based device like a solid-state drive (SSD) or SD card is practically impossible [3]. Furthermore, even SRAM memory exhibits some parasitic

burn-in effects, allowing to recover the data it contained [4]. With this in mind I decided not to build a box that's tamper-proof, but one that will indicate it was tampered with.

Principle of operation

Alice stores a freshly generated OTP key on a microSD card and puts it in the tamper-evident box. She closes the box and arms it by storing a challenge-and-response string in it by means of a non-contact (infrared) link. She then mails the box to Bob. When he receives it, he asks Alice for the challenge string. He can use any 'non-secured' channel for this — e.g. phone or email —, as only Alice knows this string and Bob has the box. Using the optical link, Bob unlocks the box by entering the challenge string. The box will respond with either the response string or with a warning about a potential tampering event. Only Alice can confirm the validity of the response string. If the response string is invalid or if the box gave a tamper event warning, Bob simply discards the SD card and will not use the potentially unsafe OTP key for secure communications with Alice.

If the box is opened without first being disarmed using the challenge string, or if it is tampered with in some other way, it will securely delete the challenge-and-response strings from its memory. Eavesdropping Eve or malicious Mallory may intercept the box to read and copy the SD card and forward the box to Bob, but they can't re-arm the box with the same challenge-and-response sequences as only Alice knows them, and so Bob will eventually find out.

Hardware description

Refer to **Figure 1** for the complete schematics of the tamper-evident box. Microcontroller IC1 is the heart of the device. Every two seconds it wakes up from power-save mode for less than 50 ms to check several parameters. If these parameters indicate a tampering event, the MCU securely erases (zeroizes) its SRAM contents, something

PROJECT DECODER

- Secure communication
- encryption
- IR communication
- AVR
- entry level
- intermediate level
- expert level
- 3 hours approx.
- (SMD) soldering iron, AVR programmer
- €50 approx.

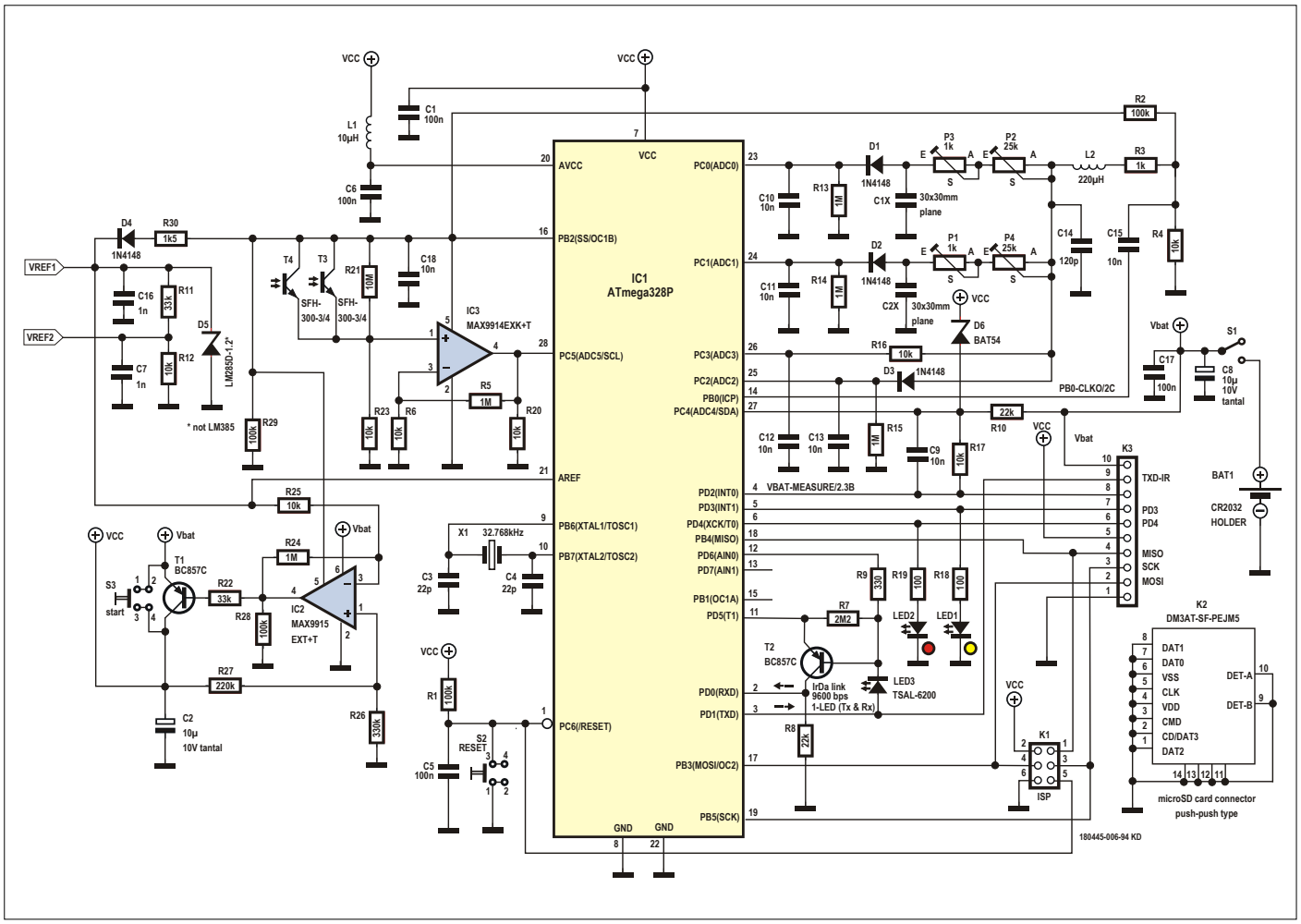


Figure 1. The tamper-evident box uses phototransistors and open-plate capacitors for detecting tampering.

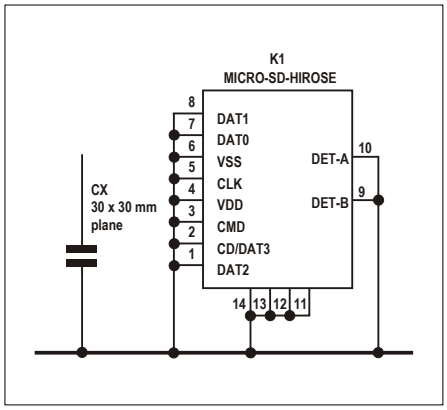


Figure 2. The open-plate capacitors C1x and C2x are 30 x 30 mm plates of PCB material with a copper plane on one side. A microSD card connector can be mounted too.

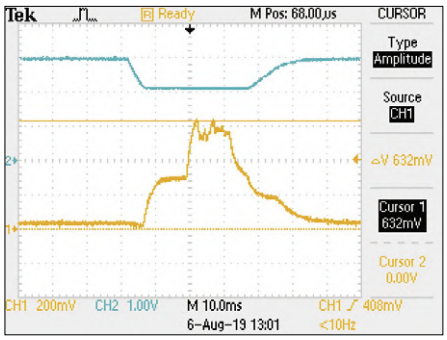


Figure 3. To save power the box wakes up only every two seconds for about 50 ms. The voltage regulator is activated (top trace) to measure vital tamper parameters. The bottom trace shows the current consumption.

that is not possible with flash memory or EEPROM.

Opening of the box is detected in two ways:

- Change in capacitance of the box (it must be a metal box). C1x and C2x (**Figure 2**) are open-plate capacitors 'protecting' the SD card placed in between them. Their other plate is the metal (!) enclosure of the box. A change in their capacitance indicates tampering.
- Light detection. The box is sealed and 100% lightproof; inside the box it's completely dark. Phototransistors T3 and T4 detect infrared and visible light as soon as the box is opened.

For entering the challenge-and-response codes an infrared IrDA-like link is used. LED3 ensures bidirectional communications. The maximum range is around one centimetre, but this is more than enough. The circuit is powered by one CR2032 battery with a nominal voltage of 3 V. It will last for more than one month (**Figure 3**), which should be enough even for the slowest snailmail carriers. A fresh battery typically starts at 3.3 V, and the circuit will work with a supply voltage as low as 2.1 V. This is made possible by the low-dropout (LDO) voltage regulator built around IC2, which produces a stable 2.0 V at V_{CC}.

D5 provides a precise voltage reference for the MCU's analogue-to-digital converter (ADC). The LM285 is an extended-temperature-range variant of the LM385, allowing the circuit to operate safely with sufficient precision from −20 °C up to +60 °C i.e. temperatures that may be experienced during its papermail *voyage*.

The MCU is clocked from its internal 1-MHz RC-oscillator. To keep energy consumption to a minimum, this clock is only turned on at two-second intervals for a duration of about 50 ms. The low-power 32.768 kHz clock based on watch crystal X1 runs continuously and times these wake-up moments. It is also used to tune the MCU's UART clock on each wake-up, and it provides the timebase for precisely measuring the duration of the box's journey.

R3, L2 and C14 filter the 1-MHz square wave derived from the MCU clock output on PB0, to obtain a sinewave that can be used to accurately measure the C1x and C2x capacitances. Both peak and mean values are measured, and the ratios of the AC voltage on C1x and C2x with respect to the AC voltage on C14 get calculated. A significant change in one of these ratios indicates tampering.

LED1 and LED2 indicate various states of the box. For instance, LED1 blinks when a correct challenge code was entered, and LED2 blinks when it was incorrect. They both blink once after power-up. LED1 flashes once on every MCU wake-up as it is used to help the LDO output to quickly reach 2 V. If LED2 blinks once on every MCU wake-up, then a tamper event was detected.

Switch S1 is the box's power switch, and S2 is the MCU's reset button. Pushbutton S3 allows to boot the circuit in case the battery voltage is a bit on the low side.

Note that the PCB has components on two sides. That's in order to hide a part of the circuit from a malicious investigator, making it more difficult to find good spots for placing probes or interfering otherwise without removing the PCB. The open-plate capacitor on the bottom side will detect this.

The IR communications adapter

An infrared (IR) link is available for communicating with the hermetically sealed tamper-evident box. This link is not particularly fast (9,600 baud) and has a rather short-range (1 cm), but it works without the need to open the box. The adapter (**Figure 4** and **Figure 5**) that makes this possible consists of a small microcontroller, an ATtiny45, equipped with an IR transceiver almost identical to the one of the box. The circuits differ slightly due to the low-power constraints for the box that do not apply to the IR adapter.

The user side of the adapter has a serial port. Since entering the challenge-and-response codes is a security-critical procedure, the IR adapter can connect to a computer in various ways, e.g. through a USB-to-serial converter (cable), over a 'real' RS-232 port, and even over Ethernet if you add a WIZnet WIZ107 Ethernet-to-serial adapter. These options allow dedicating a cheap notebook PC that never ever connects to the Internet to the box.

Adjusting the capacitive detectors

For proper operation of the capacitive tamper detection the trimpots P1, P2, P3 and P4 need to be adjusted. Power up the box (LED2 will blink once, followed by LED1) and place the IR adapter's IR LED in front of LED3 of the box at less than 1 cm. Connect the IR adapter to a computer running a serial terminal (9600n81). After typing 'U' (0x55) in the terminal, you will see this data block arrive every two seconds:

```
Box unlocked. Status-1. Define
   challenge string.
Capacitor Cx1 voltage ratio: 0.703
Capacitor Cx2 voltage ratio: 0.703
Phototransistor voltage [V]: 0.181
Box temperature [degC]:      +22
Battery voltage [V]:         2.900
```

For maximum measurement sensitivity trimmers P1 through P4 have to be adjusted in such a way that the combined resistance [P2+P3] equals the reactance C1x, and the sum [P1+P4] equals

reactance C2x. This sounds complicated, but the status data contains voltage ratios to help you here. They must be close to $1/\sqrt{2}$, i.e. 0.707. Because the box itself acts as a capacitor plate for the open-plate capacitors C1x and C2x, the voltage ratios must be correct with the box's lid in place. Use P2 and P4 for coarse adjustments and P1 and P3 for fine tuning.

Temperature measurements

The MCU has a built-in temperature sensor. Its precision is low — some ±5°C — but good enough to detect abnormal temperatures and to log the approximate minimum and maximum temperatures during the box's journey. The temperature is measured by reading ADC channel 8, which is the MCU's internal thermometer. Although designed to work with the MCU's internal voltage reference (1.1 V and not very stable), if we throw in some math (an affine function), an external voltage reference (D5) can be used to improve things. This way much better precision than stated in the MCU's datasheet can be obtained. If needed, the slope of the affine function can be modified inside the program, in the function measure_signals().

The temperature sensor needs calibrating to make it useful. The 'TP' command is available for this, to be sent from the terminal. Its syntax is 'TPsxx' where 's' is the sign '+' or '−' (the sign is mandatory), and 'xx' is the offset (as a 2-digit value, i.e. '00' to '99'). As an example, if the 'Box temperature' value is given as +40°C when the ambient temperature really is +25°C, then you should send 'TP−15' from the terminal. Sending 'TP+00' will set the offset back to zero. The offset is stored in the MCU's internal EEPROM.

User manual for spies

Alice creates a file of random numbers with her TRNG and copies it to a never-used SD card. She inserts a new battery in the box and switches it on with S1. Because the battery is fresh, it will not be necessary to press S3 to make the device boot. Then she puts the SD card into the socket and closes the box. With the IR adapter in front of the IR LED of the box she first sends the character 'U' (0x55) from her terminal. This allows the MCU of the IR adapter to calibrate its UART clock. The terminal will start receiving messages from the box, displaying

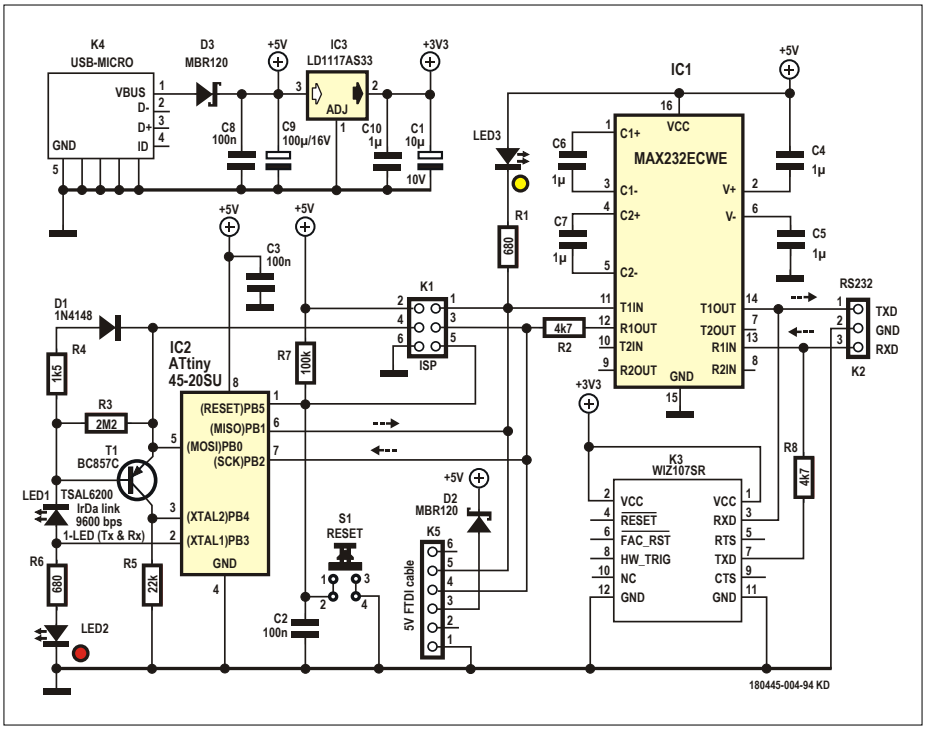


Figure 4. The IR communication adapter provides an interface between the hermetically-sealed tamper-evident box and a computer.

something like this:

```
Box unlocked. Status-1. Define
   challenge string.
Capacitor Cx1 voltage ratio: 0.710
Capacitor Cx2 voltage ratio: 0.704
Phototransistor voltage [V]: 0.181
Box temperature [degC]:      +25
Battery voltage [V]:         3.354
```

The box outputs these blocks of data every two seconds on its infrared port, except in states 3 and 5. In these states data transmissions stop after 60 seconds to save power.

Status-1

Status-1 is the initial state of the box after power-up, and the device waits for a challenge string to arrive on its infrared receiver. Alice enters the challenge string in the terminal (up to 65 characters, longer strings are truncated). The box will now enter Status-2.

Important note: all strings, including

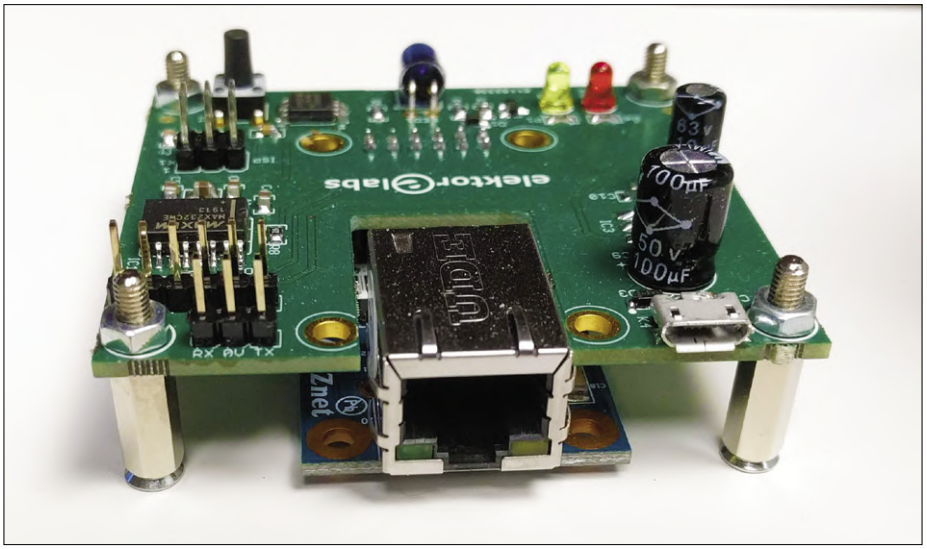


Figure 5. The IR communication adapter with the (optional) Ethernet module mounted.

multi-character commands, must be copy-pasted into the terminal and not typed character by character!

```
Box unlocked. Status-2. Define
   response string.
Capacitor Cx1 voltage ratio: 0.710
Capacitor Cx2 voltage ratio: 0.704
Phototransistor voltage [V]: 0.181
Box temperature [degC]:      +25
Battery voltage [V]:         3.354
```

Status-2

The box waits for a response string to be entered. The voltage ratios of Cx1 and Cx2 and the phototransistor voltage are stored in the MCU's SRAM. If these voltages change too much during the voyage then tamper alert is given. Alice enters the response string (up to 65 characters) to arm and lock the box and make it enter Status-3. The travel-time clock is started too. Alice writes down the date and exact time of day along with the challenge and response strings and hides it all securely (in her head, if possible). It will be used for verification later. She then mails the box to Bob.

```
Box locked. Status-3. Enter
   challenge string to unlock.
Capacitor Cx1 voltage ratio: 0.710
Capacitor Cx2 voltage ratio: 0.704
Phototransistor voltage [V]: 0.181
Box temperature [degC]:      +25
Battery voltage [V]:         3.354
```

Status-3

Both the challenge-and-response strings have been entered and the box is armed and locked. Any tampering with it or entering an incorrect challenge string five times will trigger the zeroization

COMPONENT LIST

Tamper-evident Box

Resistors

Default: SMD 0805
R18,R19 = 100Ω
R9 = 330Ω
R3 = 1kΩ
R30 = 1.5kΩ
R4,R6,R12,R16,R17,R20,R23,R25 = 10kΩ
R8,R10 = 22kΩ
R11,R22 = 33kΩ
R1,R2,R29,R28 = 100kΩ
R27 = 220kΩ
R26 = 330kΩ
R5,R13,R14,R15,R24 = 1MΩ
R7 = 2.2MΩ
R21 = 10MΩ
P3,P1 = 1kΩ trimmer
P2,P4 = 25kΩ trimmer

Capacitors

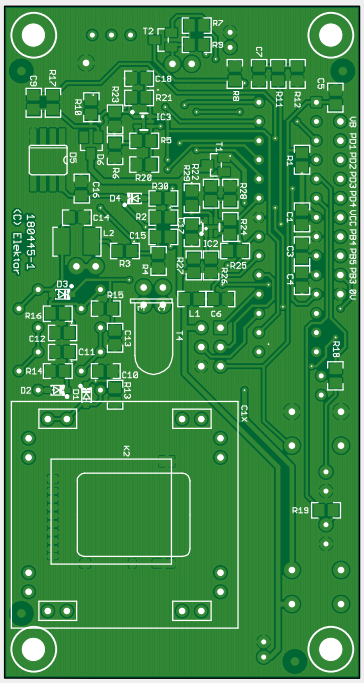
Default: SMD 0805
C3,C4 = 22pF
C14 = 120pF
C7,C16 = 1nF
C9,C10,C11,C12,C13,C15,C18 = 10nF
C1,C5,C6,C17 = 100nF
C2,C8 = 10µF 10V, tantalum, 2mm pitch

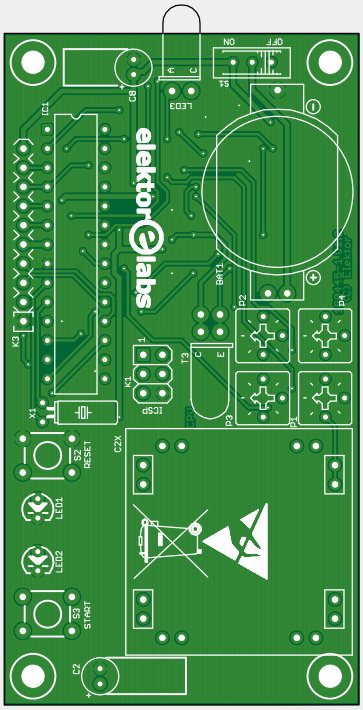
Inductors

L1 = 10µH (0805)
L2 = 220µH (1812)

Semiconductors

D1,D2,D3,D4 = 1N4148WS (SOD-323)
D5 = LM285D-1.2V
D6 = BAT54
IC1 = ATmega328P-PU, programmed
IC2 = MAX9915EXT+T
IC3 = MAX9914EXK+T
LED1 = 3mm, yellow
LED2 = 3mm, red
LED3 = TSAL6200
T1,T2 = BC857C

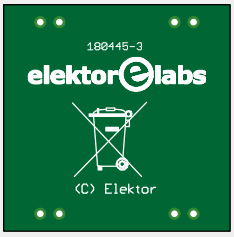


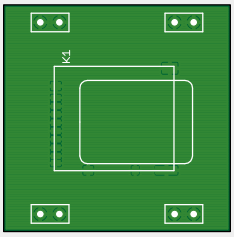


T3,T4 = SFH-300-3/4

Miscellaneous

BAT1 = battery holder, 2032, e.g. Renata HU2032
C1x,C2x = 2-pin pinheader + socket for C1x & C2x, 0.1" pitch
K1 = 6-pin (2x3) pinheader, 0.1" pitch
K2 = microSD card connector, push-push
K3 = 10-pin pinheader, 0.1" pitch
S1 = slide switch, SPDT
S2,S3 = tactile switch, 6x6 mm
X1 = 32.768kHz quartz crystal
Sealed LED holder for LED3
1 pc PCB #180445-1
2 pcs PCB #180445-3





sequence. The box remains in Status-3 during its travel with the papermail carrier.

The SRAM buffers containing the challenge and response strings are complemented (bitwise-inverted) every two seconds. This way, the parasitic burn-in of SRAM cells [4] is prevented, because each bit will spend and equal amount of time in both the '0' and '1' state.

At Bob's place

After a long and perilous journey, the box (hopefully) arrives at Bob's place. He connects his IR adapter to a computer, places it in front of the box's IR LED, and presses the Reset button on the adapter. This will produce a three-second pulse on the infrared port to tell the box to activate its infrared port and start communicating on it. After three seconds Bob proceeds by sending a single character 'U' to calibrate the IR adapter's UART, and the terminal starts displaying data:

```
Box locked. Status-3. Enter
   challenge string to unlock.
Capacitor Cx1 voltage ratio: 0.703
Capacitor Cx2 voltage ratio: 0.711
Phototransistor voltage [V]: 0.175
Box temperature [degC]:      +23
Battery voltage [V]:         2.752
```

As we can see, the battery voltage has decreased, but there are no visible signs of tampering as the Cx voltage ratios and the phototransistor voltage values have not changed much. Bob calls Alice and asks her for the challenge string. He enters it in his terminal, and, if everything is fine, he will see:

```
Box unlocked. Status-4. Challenge
   string correct!!
Capacitor Cx1 voltage
   ratio:      0.703
Capacitor Cx2 voltage ratio: 0.711
Phototransistor voltage [V]: 0.178
Box temperature [degC]:      +24
Battery voltage [V]:         2.752
Min temperature [degC]:      -2
Max temperature [degC]:      +31
Travel time [s]:           502,164
Response:
Trust no one 01a2g23w46e57f80g12r3
   e34fv245hasdvfr4.
```

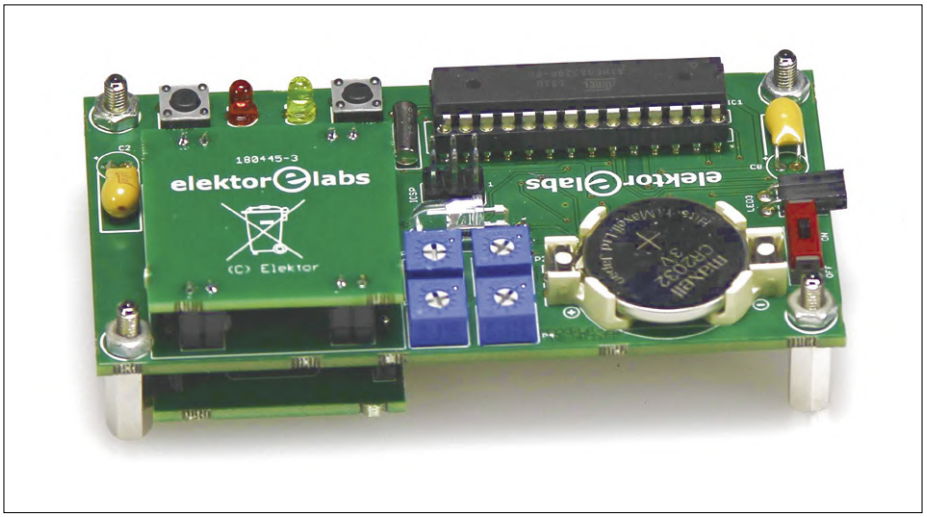


Figure 6. The fully assembled board with open-plate capacitors installed on both sides.

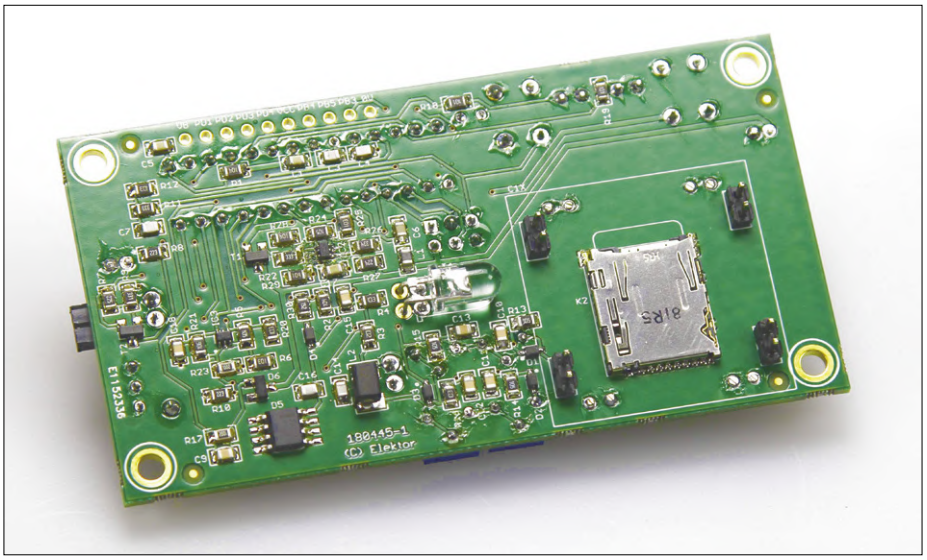


Figure 7. The bottom side of the board/ PCB is populated with SMD components and through-hole transistor T4 that looks like an LED. Capacitor C1x is mounted over the microSD card connector.

Status-4

Bob immediately writes down the current time and date and calls Alice again to verify the response string. If Alice accepts it, she will now give Bob the exact time and date that she wrote down when she locked the box. This allows Bob to calculate the theoretical travel time and compare it to the time measured by the box. If the two match, then Bob is allowed to use the OTP key stored on the microSD card to encrypt and decrypt his secure communications with Alice. In case of being tampered with along the way, the box will erase the part of the SRAM containing the codes and the travel timer. In this case the box will output something like this:

```
Box zeroized. Status-5. Tampering,
   wrong password, or low battery!
Capacitor Cx1 voltage ratio: 0.712
Capacitor Cx2 voltage ratio: 0.713
Phototransistor voltage [V]: 0.179
Box temperature [degC]:      +12
Battery voltage [V]:         2.793
```

Status-5

The zeroization sequence was executed due to the detection of a tamper event or after entering five invalid challenge strings. The tamper-check voltages are no longer updated, allowing the holder of the box to see the last recorded state before tampering was detected.
The zeroization sequence is executed in one of the following cases:

- After entering an incorrect challenge string five times;
- The actual capacitor voltage ratio differs too much from the recorded value;
- The actual phototransistor voltage differs too much from the recorded value;
- Battery voltage too low (below 2.1 V);
- Temperature out of bounds (below −20°C or above +60°C).

To avoid stray light from triggering tamper event 3, make sure the box is hermetically sealed. Apply some black sealant paste around LED3 on the inside of the box and seal the edges of the box with sticky black electrical isolation tape. Test it by exposing the closed box (in Status-3) to direct strong sunlight. Now Bob can enter the 'DE' command to see what has caused the zeroization:

```
Box zeroized. Status-6. Tampering
   status display!
Capacitor Cx1 voltage ratio: 0.683
Capacitor Cx2 voltage ratio: 0.656
Phototransistor voltage [V]: 0.188
Box temperature [degC]:      +10
Battery voltage [V]:         2.775
Min temperature [degC]:      -5
Max temperature [degC]:      +30
Cause of zeroization:
   Capacitor Cx2. kCx2= 0.656
```

Status-6

After receiving the 'DE' command the box will transmit details about the cause of the zeroization.

The data displayed in states 5 or 6 doesn't have to be valid as it depends on what Mallory has done to the box. She may have reset it and entered challenge and response strings before forwarding the box to Bob in Status-3. However, even if she somehow manages to enter the correct strings, she will have to spoof the travel timer as well, and this will be much more difficult.

Back to Status-1

The command 'RE' can be used to reset the box to Status-1 without opening it. Pressing the S2 reset button inside the box will do the same.

A word about hardware Trojans

Since most people usually only worry about software malware and Trojans attacking their computers, a few words about hardware Trojans is in place here. A simple system with a single MCU that is never connected to the Internet is 100% secure against typical malware that would target your PC or smartphone. Please watch [5] for a basic explanation of the problems related to solid-state memory devices at very high level of integration.

A hardware backdoor Trojan consists of extra circuitry added to the MCU's silicon die. This can do as much harm as a software Trojan or even more. It poses a great danger for today's MCUs with their CPU, SRAM, EEPROM and flash memory integrated on one silicon die, and that are programmable with a low voltage (3 to 5 V). Basically, there are two types:

1. General-purpose hardware Trojans

Mallory can plant a hardware Trojan in the MCU, e.g. to periodically copy the MCU's SRAM and registers to some secret part of flash memory. Regardless of the MCU's application, a 'frozen' snapshot of RAM will always be useful for her and Eve. With such a Trojan in place Mallory can easily read the challenge and response strings defined by Alice. However, spoofing the travel timer will be a little bit more difficult.

2. Application-specific hardware Trojans

To plant this kind of hardware Trojan, Mallory needs to have knowledge about the MCU's firmware first. Then she rigs the MCU with a circuit capable of recognizing the firmware that Alice flashed inside her MCU. Upon starting the MCU, the Trojan will act as a bootloader and change a few lines in the MCU's program (e.g. fill them with NOPs using the AVR's notorious SPM command) to disable for instance some crucial protective actions. The possibility of a successful attack

COMPONENT LIST

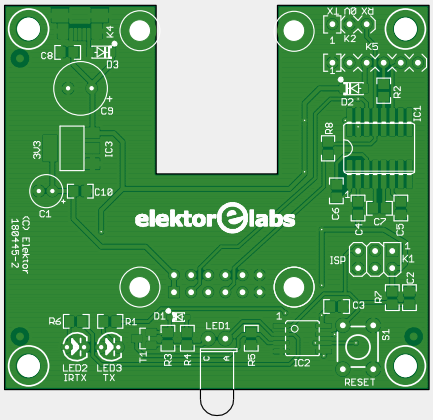
IR Adapter

Resistors
Default: SMD 0805
R1,R6 = 680Ω
R4 = 1.5kΩ
R2,R8 = 4.7kΩ
R5 = 22kΩ
R7 = 100kΩ
R3 = 2.2MΩ

Capacitors
Default: SMD 0805
C2,C3,C8 = 100nF
C4,C5,C6,C7,C10 = 1µF, X7R, 50V
C1 = 10µF 10V tantalum, 2mm pitch
C9 = 100µF 16V, 3.5mm pitch

Semiconductors
D1 = 1N4148WS (SOD-323)
D2,D3 = MBR120
IC1 = MAX232ECWE
IC2 = ATtiny45-20SU, programmed
IC3 = LD1117AS33
LED1 = TSAL6200
LED3 = 3mm, yellow
LED2 = 3mm, red
T1 = BC857C

Miscellaneous
K1 = 6-pin (2x3) pinheader, 0.1" pitch
K2 = 3-pin pinheader, 0.1" pitch
K3 = 12-way (2x6) pinheader socket, 0.1" pitch
K4 = Micro USB connector Type B
K5 = 6-pin pinheader, 0.1" pitch
S1 = tactile switch, 6x6mm



WIZ107 Ethernet module
PCB #180445-2

of this kind is much lower than with a general-purpose Trojan.

Is there any defense?

Luckily for Alice and Bob, there are many ways to counter this kind of hardware threats. The simplest is to return to 1980's technology and use a simple Z80 system like [6] with SRAM and EPROM as separate ICs. The EPROM is erased with UV light and needs a voltage of 13-14 V on its Vpp pin to be written, making it much harder to plant a hardware backdoor. Furthermore, the internal architectures of the Z80 and 27C-type EPROMs (produced in many millions for over 20 years) are very well known, and it is easy to decapsulate the IC and inspect it under a microscope if needed. With millions of pieces lying around the globe, Mallory would have a very hard time rigging them all.

Porting the MCU firmware of the TRNG or OTP Crypto Shield to a Z80 can be done at the expense of execution speed and physical size, but the systems would still work fine. Please note another misconception of today's design engineering here: small-, mini-, micro- and nano-sized devices are not always better!

Of course, a Z80-based system would be too unwieldy and power-hungry to be used for the paper-mail box, but this problem can be solved in another way, with a little extra circuitry, everything fitted inside the existing box.

If you studied the schematics carefully, you may be wondering about the function of K3. It is intended for an add-on circuit to protect the box against hardware Trojans. More about this in a future article in Elektor, so stay tuned!

Sounding off

The prototype of the box as shown on the photos has made several trips with different land mail carriers and was found to function well. However, some carriers are rougher than others and it turned out to be necessary to block C1x and C2x to avoid them to come loose. Sticking non-conductive foam or bits of plastic on top of them is one method. If the MCU is mounted in a socket, you might want to block it too. The battery sits quite tight in its holder, but you never know.

If you have other suggestions for improving the device described in this article, especially regarding the hardware-backdoor defense part, please contact me through the project page at Elektor Labs [7]. I have a few ideas but you may have better ones, and other methods can always be proposed. To confirm its security, any crypto device must be reviewed by a number of independent experts, so please feel free to participate in any way!

As usual, the design files and source code for this project can be downloaded from the project's webpage [7]. ⏮

180445-01



Bob and Alice finally meet in person.

4 SALE @ WWW.ELEKTOR.COM

→ Tamper-Evident Box - panel of all bare PCBs
www.elektor.com/180445-1

→ Tamper-Evident Box - preprogrammed microcontroller ATmega328P-PU
www.elektor.com/180445-41

→ IR Adapter - programmed microcontroller ATtiny45-20SU
www.elektor.com/180445-42

Web Links

[1] Random Number Generator: www.elektormagazine.com/labs/random-number-generator-150116

[2] One-Time Pad (OTP) Crypto System: www.elektormagazine.com/labs/one-time-pad-otp-crypto-system

[3] Erasing data from flash drives: www.schneier.com/blog/archives/2011/03/erasing_data_fr.html

[4] Data Remanence in Semiconductor Devices: www.usenix.org/legacy/events/sec01/full_papers/gutmann/gutmann.pdf

[5] Elektor video interview on security: www.elektormagazine.com/news/secure-communications-an-interview-with-luka-matic

[6] A small and simple Z80 computer: www.sunrise-ev.com/z80.htm

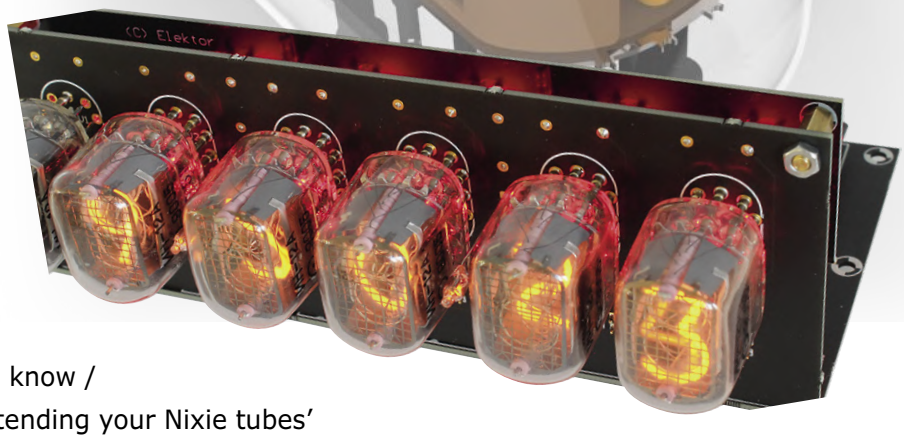
[7] Tamper-evident paper-mail box for secure distribution of one-time pads: www.elektormagazine.com/labs/tamper-evident-paper-mail-box-for-secure-distribution-of-one-time-pads

eLABS PROJECT

Elektor New Precise Nixie Clock Revisited

An engineering bulletin for enthusiastic users new and old

By **Andy Leitch** (UK) and **Luc Lemmens** (Elektor Labs)



Elektor's New Precise Nixie Clock a.k.a. 6-Digit Nixie Clock was among the top ranking projects published in the past five years, both in terms of readership resonance and kit sales. Not resting on our laurels, we compiled and now publish a set of tweaks and enhancements for the clock. The modifications are of different levels and intents, and range from 'nice to know / optional' right up to serious advice like extending your Nixie tubes' lifetime. Although the mods have resulted in an updated clock board and a wholly new backlight board, it's very well possible to adapt your existing board and run the new software too.

PROJECT DECODER

- Nixie
- PIC
- clock
- update

- entry level
- ➡ intermediate level
- expert level

- 1 hour approx. (upgrade/mods only);
- 3 hours (semi-kit assembly)

- lab tools,
- PIC programmer

- €10 approx. (upgrade/mods only);
- €140 approx. (semi-kit excl. case)

Back in May 2016 Elektor Labs magazine published the New Precise Nixie Clock [1] and since then many readers either purchased the complete kit or just the bare PCBs, and bought their own components to build the project. In fact, judging from the number of views as well as tech-interested followers on the Elektor Labs website [2] it's one of the most popular designs of the past years.

In the wake of the May 2016 publication, we received requests and suggestions to enhance and extend the clock, especially the software. That job was finally seen to by one of the staunchest followers on Labs, Andy Leitch. Andy completely rewrote the software originally produced in assembler, to 'C' and adapted the hardware enabling the clock to also display the date and act as an alarm clock, among other things. To fully exploit the functionality of the new firmware now available [3], the circuit boards require minor adaptations. However, if you are satisfied with adding date display and preventing/limiting tube cathode poisoning (more about that later), then replacing or reprogramming the microcontroller will suffice.

The most eye-catching enhancement Andy made is the multi-colour backlighting for the Nixie tubes to replace the six blue LEDs from the semi-kit. Besides being decorative, the new backlighting also indicates what information is being shown on the clock display. Andy used a strip of perfboard and through-hole WS2812 RGB LEDs for this, but in order to make it completely tidy and easy to assemble, Labs designed a circuit board with the same LEDs in SMA housings. This PCB is available from the Elektor Store in the same black colour as the other PCBs from this project. More about it below.

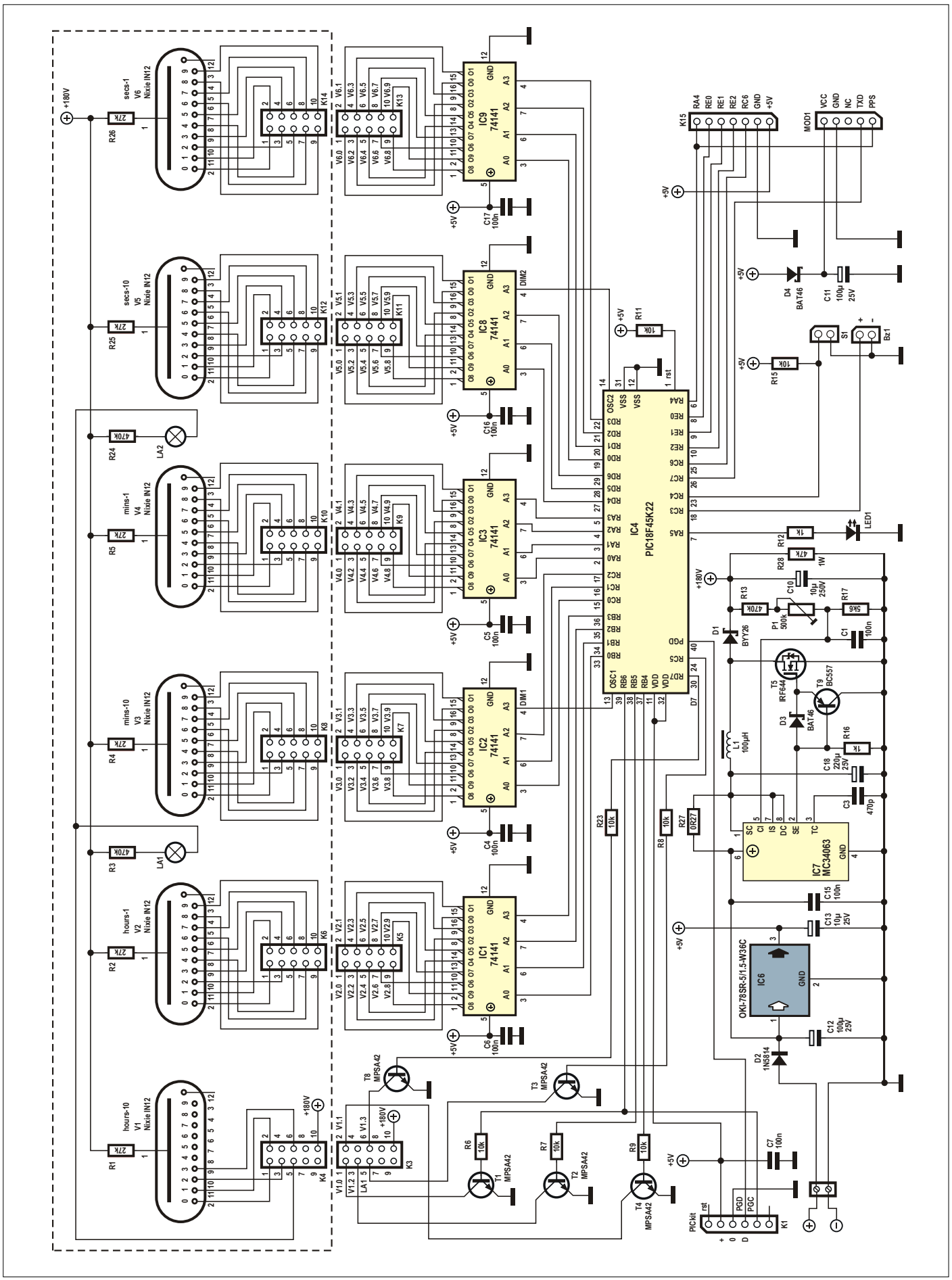


Figure 1: Circuit digram of the the revised New Precise Nixie Clock. This version incorporates all changes to the original design of 2016.

New features, new circuit diagram

The hardware and some of the software modifications to the New Precise Nixie Clock have resulted in a new schematic printed in **Figure 1**. This schematic matches the new printed circuit board now included in semi-kits of the New Precise Nixie Clock sold through the Elektor Store. The instructions are for the old PCB; the new version V2.1 version does not need any hardware mods.

Button modification

Status: mandatory.

The existing toggle switch is best replaced with a momentary-action type. This very simple modification will open most features of the new software. Simply unsolder the existing switch and replace with a pushbutton. You may need to drill out the mounting hole to suit the replacement switch, but if you use one with a 4.95-mm shaft, it will fit without drilling.

Preventing cathode poisoning

Status: highly recommended.

If a Nixie tube is operated without regularly exciting all the numbers, the unused wires tend to get coated by a highly resistive sputtered material. The effect is commonly referred to as *cathode poisoning*. It will result in dark spots or segments of the numbers when they should completely be lit. If Nixies are used in a clock, the seconds digit is unlikely to be affected by cathode poisoning as it cycles through all numbers every ten seconds, but other digits are in danger.

Cathode poisoning can largely be prevented (or at least limited) by temporarily cycling every tube through all its numbers every now and then, and that is exactly what the new software does. The first 'cleaning' will be done at power-up and is repeated every 30 minutes (at 15 and 45 minutes past the hour). If installed, the WS2812 backlight will blink and cycle through colours when the 'preventive cleaning' is performed. This option can be switched off in the clock's settings.

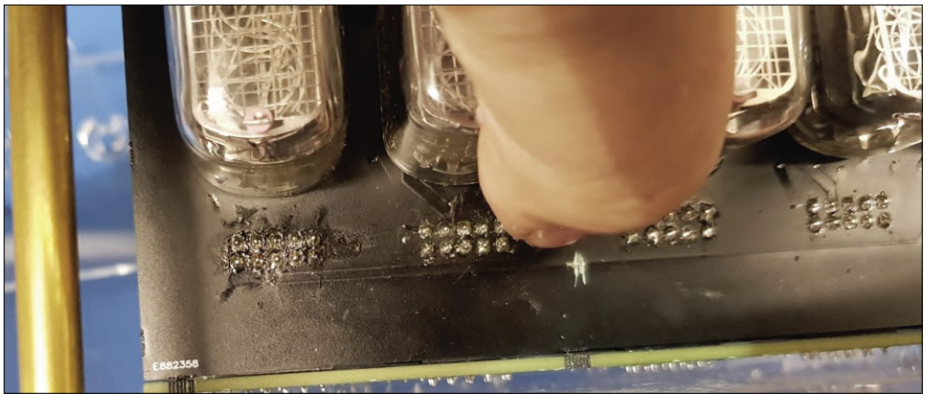


Figure 2: Cutting a PCB track to disconnect one neon from its driver. (change applies to existing PCBs only)

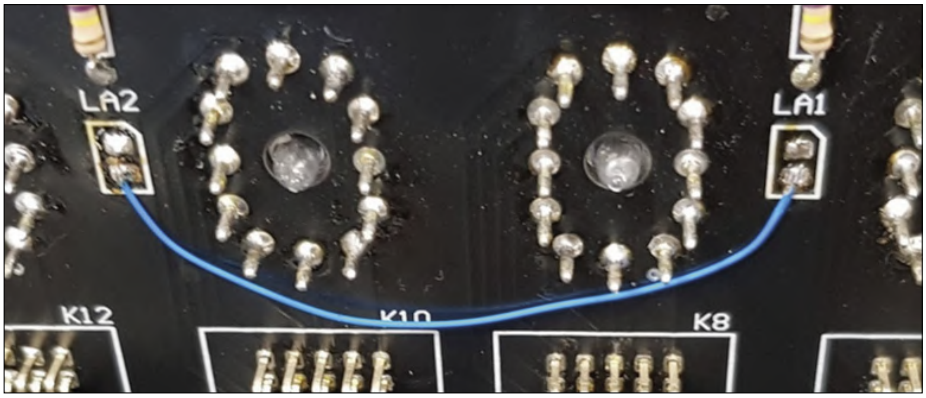


Figure 3: Interconnecting the neon lamps with a wire. (change applies to existing PCBs only)



Figure 4: Connecting the '3' element of Nixie tube V1 to the middle pin on top of K4. (update applies to existing PCBs only)

Alarm sounder modification

Status: highly recommended.

Although in principle this is only needed if alarm clock function is used, the beep when setting parameters is in practice essential. A low-current device is needed and these are easily obtained from Amazon, eBay, RS components, etc. A search for 3–24 V piezo buzzers should find a suitable candidate. Check the supply current is 10 mA or less, as the PIC can only supply 25 mA out of the port pin.

The sounder is polarity-sensitive and is connected black wire to pin 1 of JP1 (at the edge of the PCB) and red wire to the pin on the PIC side. R14 can be desoldered or cut from the board. A small hole in the back of the case allows sound out, and the sounder can be glued in place.

Date display modification

Status: optional.

This tweak is needed to show the date in EU format (dd/mm/yy) as a 3 is needed for displaying of the days on the leftmost digits. The PCB is laid out to drive the two neons separately, and the '3' digit of the left Nixie tube is not connected. It is modified to join both neon lamps to one driver, and then connect the '3' to the now unused driver.

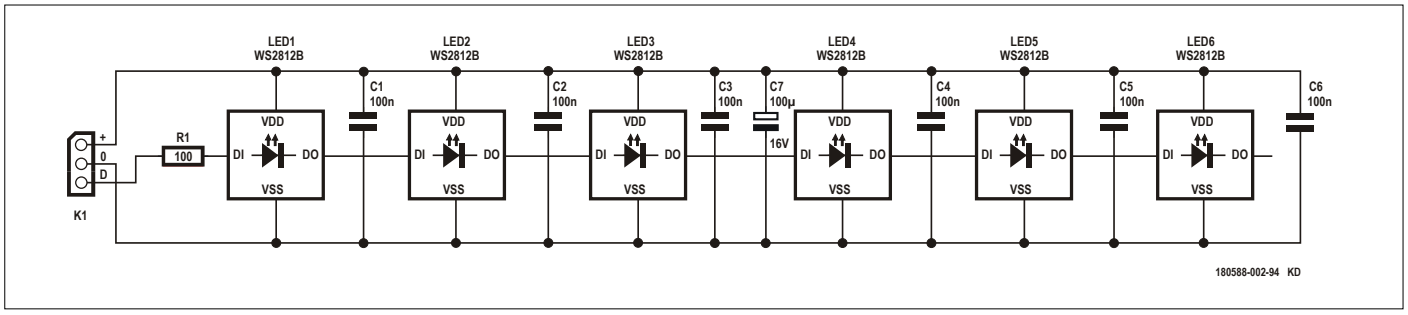


Figure 5: Circuit diagram of the decorative backlight board implemented with SMD LEDs. Note the decoupling capacitors which are needed to suppress noise caused by the relatively high current.

- **Step 1:** disconnect one neon from its driver. The lower track on the tube side of the PCB is cut with a craft knife. It is cut again approximately 1 mm along so a small section of track can be peeled away, see **Figure 2**.
- **Step 2:** Connect the neons together, see **Figure 3**.
- **Step 3:** Connect the '3'. It connects to the middle pin, top row, of K4, see **Figure 4**.

WS2812 'Neopixel' LED Backlight

Status: optional.

Caution: The LEDs need 5 V but quite a lot of current. You can use a second 5 V regulator powered from the inlet 9 V supply, or get power, ground and data from the programming header. If you take power from the programming header, you need a heatsink on IC6, or better change it to a switching type (e.g. Murata OKI-78SR-5/1.5-W36-C).

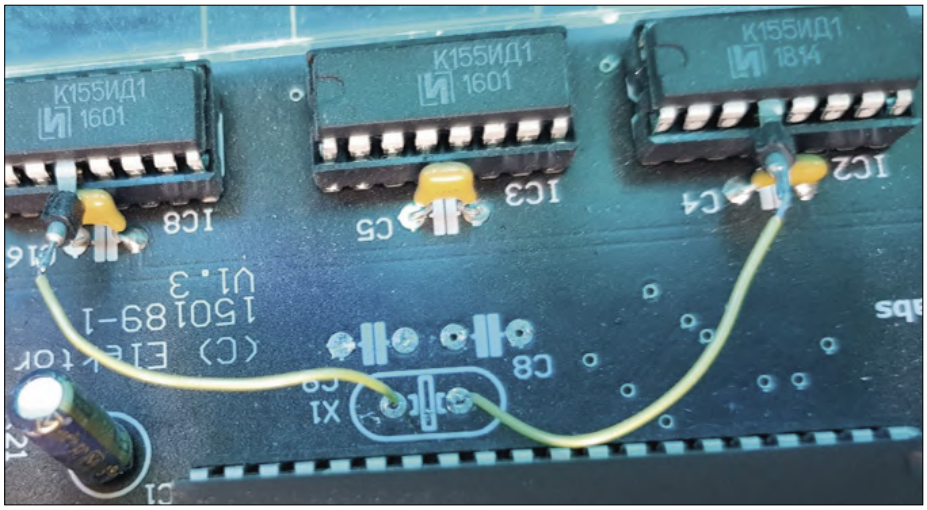


Figure 6: The vacated crystal connections on the PCB are connected to driver IC pins bent aside. (update applies to existing PCBs only)

The schematic of the lively backlight is shown in **Figure 5**. Pin RB7 is not used in the project other than for programming, so a little code was added to enable it to drive the backlight LEDs — the driving signal comes out on header K1, pin 4.

COMPONENT LIST

Backlight PCB 180588-1

Resistor

R1 = 100Ω, thick film, 5%, 0.1W, 150V 0805

Capacitors

C1,C2,C3,C4,C5,C6 = 100nF, 50V, X7R, 0805

C7 = 100µF, 16V tantalum, 10%, case 2312

Semiconductors

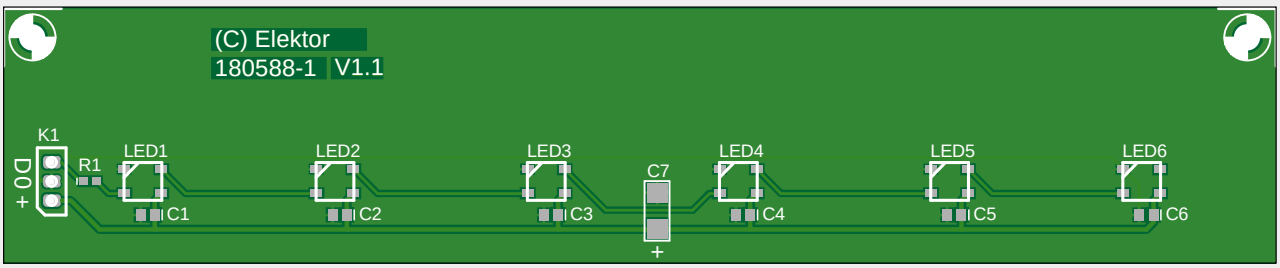
LED1 - LED6 = WS2812B RGB LED (SMD type)

Miscellaneous

3-way cable, approx. 10cm length

3-way SIL connector

PCB 180588-1 V1.1



COMPONENT LIST

Revised Clock PCB no. 150189-1 V2.1

Resistors

R1,R2,R4,R5,R25,R26 = 27kΩ 5% 250mW
R6,R7,R8,R9,R11,R15,R23 = 10kΩ 5% 250mW
R3,R13,R24 = 470kΩ 5% 250mW
R12,R16 = 1kΩ 5% 250mW
R17 = 5kΩ 6 5% 250mW
R27 = 0.27 Ohm, 5%, 2W
R28 = 47kΩ 5% 1W
P1 = 500kΩ trimpot

Inductor

L1 = 100 µH, 1.5A, radial, DxL 8x8mm

Capacitors

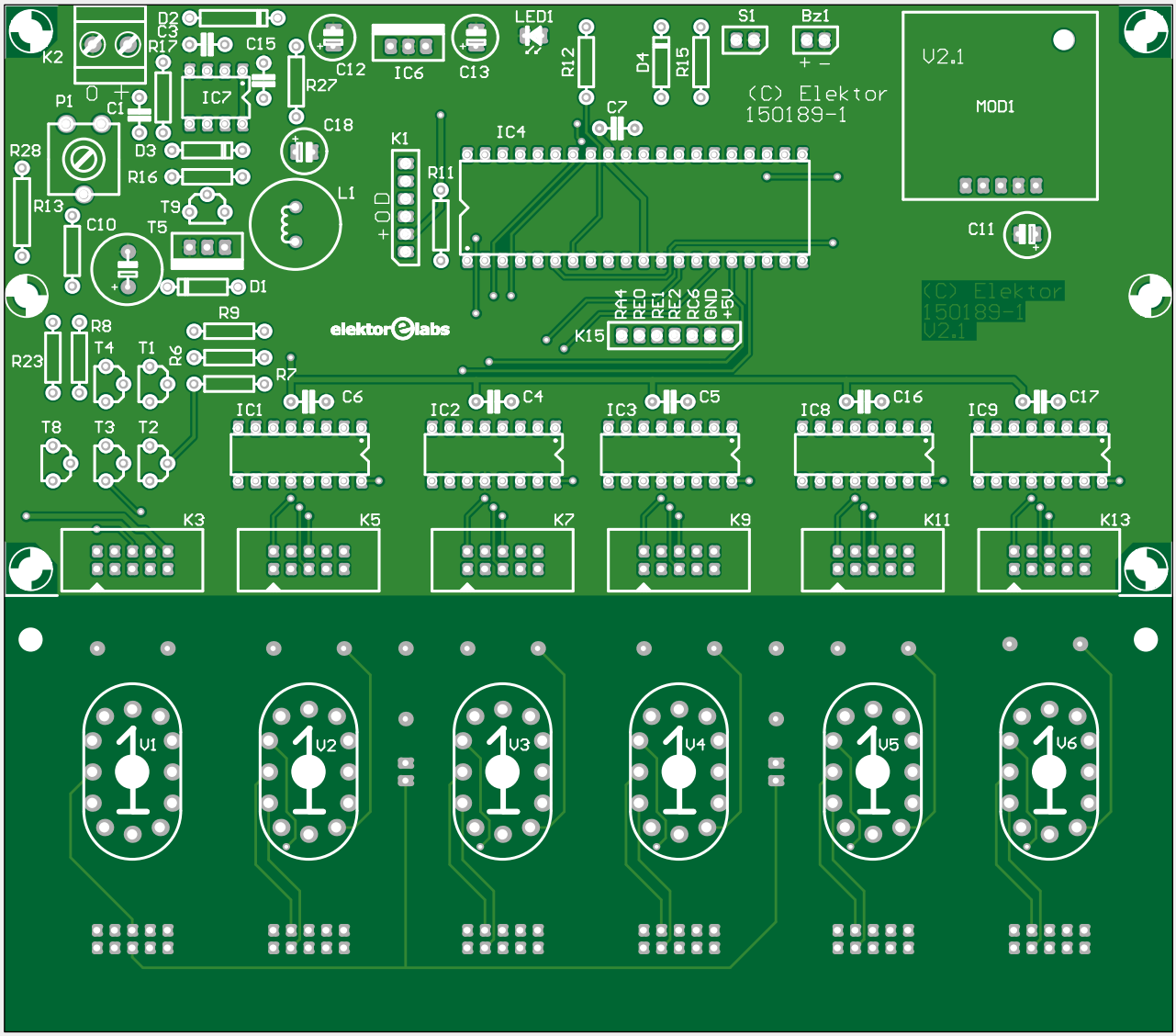
C3 = 470pF, 50V, Y5U
C1,C4,C5,C6,C7,C15,C16, C17 = 100nF 50V 20%
C10 = 10µF, 250V, 20%, radial, 5mm pitch
C11,C12 = 100µF, 25V, 3.5mm pitch, 8x11mm
C13 = 10µF, 25V, 2mm pitch, 5x1 mm
C18 = 220µF, 25V, 5mm pitch, 10x16mm

Semiconductors

D1 = BYV26, ultrafast diode, 600V, 1A
D2 = 1N5814, Schottky diode, 40V, 1A
D3,D4 = BAT46, Schottky diode 100V, 150mA
LED1 = red, 3mm
T1,T2,T3,T4,T8 = MPSA42, NPN, 300V, 500mA
T5 = IRF644PBF, NMOS, 250V, 14A
T9 = BC557C, -45V, -100mA, 500mW, h_{fe}=400
IC1,IC2,IC3,IC8,IC9 = K155ID1 (74141)
IC4 = PIC18F45K22, programmed, Elektor Store # 180588-41
IC6 = OKI-78SR-5/1.5-W36-C, DC/DC converter 5V, 1.5A, Murata
IC7 = MC34063, DC/DC controller

Miscellaneous

S1 = SPST pushbutton, 20V (shaft 4.95mm)
Bz1 = 1.5–20Vdc buzzer, e.g. Loudity LD-BZPG-2312
K1 = 6-way SIL pinheader, 0.1" pitch
K2 = 2-way PCB screw terminal block, 0.2" pitch, 630V
K3,K5,K7,K9,K11,K13 = 10-way (2x5) pin socket, snapoff, vertical
K4,K6,K8,K10,K12,K14 = 10-pin (2x5) pinheader, snapoff, right angled
K1 = 6-pin SIL pinheader, 0.1" pitch
LA1,LA2 = T1.1/4, neon lamp, wire ended
MOD1 = GY-NEO6MV2 GPS module, w. antenna
V1,V2,V3,V4,V5,V6 = IN-12 Nixie tube
PCB 150189-1 V2.1, from Elektor Store



The WS2812 SMD type LEDs are mounted on the Labs PCB with a surface mount 10 µF, 16 V capacitor in the middle of the board and 100 nF capacitors at the supply pins of each LED. The first LED has a 100-Ω resistor in series with the data wire from the RB7. Alternatively, you can build your own backlight board using stripboard (perfboard/Veroboard), and through-hole versions of the WS2812. Labs went for SMD components not only because this type of LEDs is easier to align, but also because this flat design fits into the acrylic case without any mechanical issues.

Dimming

Status: recommended.

The 74141-compatible decoders show blank for input values upwards of 9. To flash, and therefore dim, all the digits, two chips need to have their 'bit 3' pin (i.e. pin 4) connected to the processor instead of ground.

If dimming is enabled in the settings of the clock without the modification, then the digits will show '7' instead of turning off. The quartz crystal has to be de-soldered. Repeatedly heat one end at a time in turn and gently lift X1 out, the two capacitors C8 and C9 are also removed. Two of the decoder chips, IC2 and IC8, are lifted out. Pin 4 is straightened carefully, and then the ICs are re-inserted. Watch out for legs getting bent under when re-inserting! Use an old turned-pin IC socket or a female jumper cable to connect to the chip — **do not solder** as soldering causes leakage which is visible during dimming.

Link the pins to the now empty crystal connections with single core wire, as shown in **Figure 6**.

GPS module

Status: recommended.

Over the past months Labs received a few questions from readers wanting to build the Nixie clock and buy the components separately but having failed to find a source for the A2035H GPS module from Maestro Wireless Solutions. It turned out this part is no longer in production. Fortunately, Maestro now has the A2235H, a newer GPS module that does fit on the old PCB (although the module is smaller than its predecessor). However, it needs two additional 2.2-kΩ pullup resistors from the I²C pins to the 1.8 V output of the module. You may also need some (Kapton®) isolation between the PCB and the module to avoid short circuiting the shielding of the module to a via underneath [2].

Both Maestro GPS modules have a default serial output of 4800 baud. Many other modules are available, but usually run at 9600 baud.

Fortunately, as part of the software rewrite, the clock now features auto baudrate detection on the GPS module.

Operating instructions

With the new software running there is a single button to control the clock. Short presses change the display or parameter; long presses change the mode, in accordance with the current display. All options, settings and parameters available to configure and adjust the revised New Precise Nixie Clock may be found in the user manual produced by Andy and stored at [2].

Conclusion

Although the original New Precise Nixie Clock runs beyond reproach with the equally original software, the new version presented here offers several new features that are worth the update. If you should do all the hardware changes, well… that's up to you, of course. If you pick just a few, fine! The authors believe the upped GPS reception, the anti-cathode-poisoning feature and the multi-colour backlight are absolute enhancements for this beautiful and nostalgic clock!

180588-01



4 SALE @ WWW.ELEKTOR.COM

→ New Precise Nixie Clock, (revised) PCB, bare, 150189-1 V2.1
www.elektor.com/180589-1

→ New Precise Nixie Clock, Backlight PCB, bare, 180588-1 V1.1
www.elektor.com/180588-1

→ Programmed PIC18F45K22 (DIP-40), 180588-41
www.elektor.com/180588-41

→ GPS module
www.elektor.com/open-smart-gps

Web Links

[1] New Precise Nixie Clock, Elektor Labs magazine May & June 2016: www.elektormagazine.com/magazine/elektor-201605/28960/

[2] Project/Engineering web page: www.elektormagazine.com/labs/150189-6-digit-nixie-clock

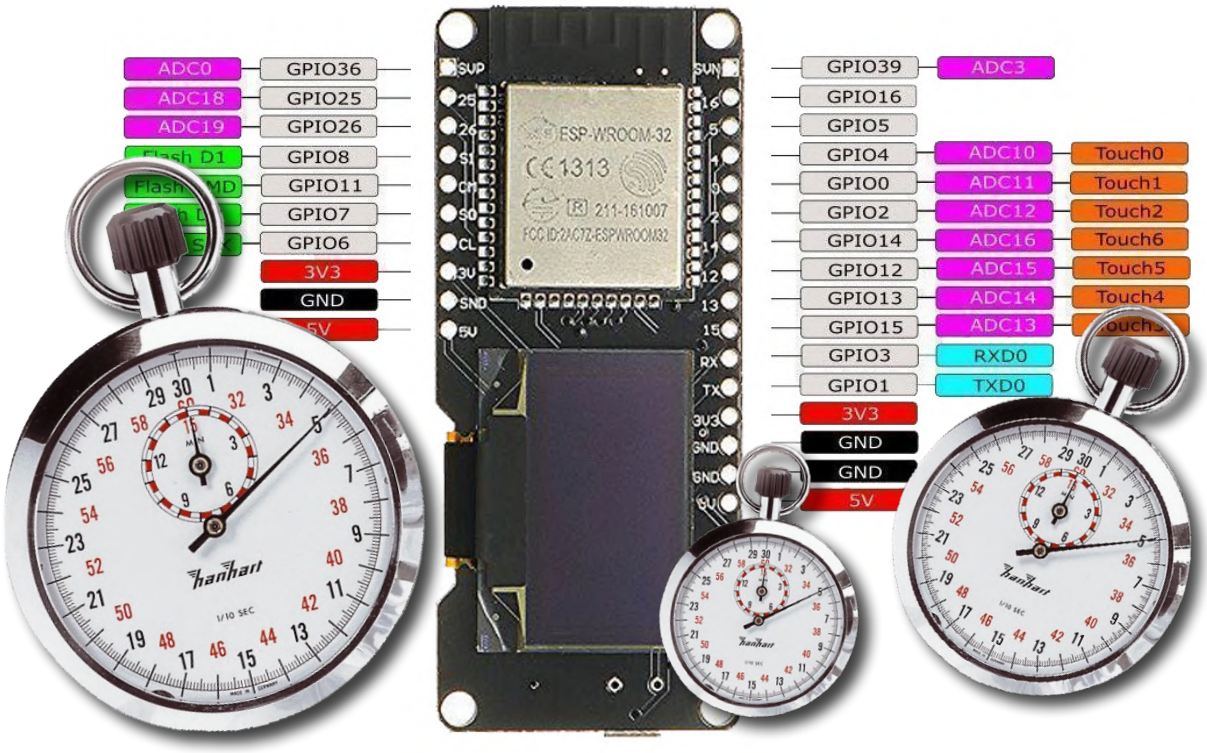
[3] Project/Publication web page: www.elektormagazine.com/180588-01

Practical ESP32 Multitasking (3)

Software timers

By **Warren Gay** (Canada)

Microcontroller developers often need timers, to perform retransmissions and other error recovery procedures, for debouncing of button presses, or just for a blinking LED indicator. The ESP32 device and the Arduino IDE provide software timers using the included FreeRTOS library. In this third part of the series we focus on the timer facility.



In order to illustrate how FreeRTOS timers work, we'll work with a demo assignment to construct an alert LED class. This is not your average blinking LED however. When activated, it quickly blinks five times and then is off for the rest of the period, and repeats. It will visually appear as an attention getting alert or as a visual phone ring alert.

The `AlertLED` class will use one FreeRTOS software timer internally for each LED driver. As a class, you will be able to add as many as you need, without burdening the memory utilization of the ESP32. **Listing 1** illustrates the full program [2]. Line 100 shows the creation of an alert LED instance with a simple C++ declaration statement:

```
static AlertLED alert1(GPIO_LED,1000);
```

The class instance name here is `alert1`, using the specified LED (GPIO 12 from line 5), and the flash period of 1000 milliseconds. We could easily add another one for GPIO 13, by simply adding the line:

```
static AlertLED alert2(13,1000);
```

Our demonstration will focus on one indicator but the C++ class packaging makes for convenient use.

The AlertLED class

The class definition is copied below for convenience. The class data members include the member `thandle` (line 11), which is a handle to the FreeRTOS timer (type `TimerHandle_t`). Initially it is assigned the value of `nullptr` (same as the C language NULL). The data member named `state` will track the LED state (line 12), which is active high (`true` = on). The data member `count` (line 13) will be used in the callback as a substate, which is explained later. Finally members `period_ms` and `gpio` configure the blink period in milliseconds and the GPIO to drive for the LED (lines 14 and 15).

Listing 1: The alertled.ino program [2].

```
0001: // alertled.ino
0002: // MIT License (see file LICENSE)
0003:
0004: // LED is active high
0005: #define GPIO_LED      12
0006:
0007: //
0008: // AlertLED class to drive LED
0009: //
0010: class AlertLED {
0011:   TimerHandle_t     thandle = nullptr;
0012:   volatile bool     state;
0013:   volatile unsigned count;
0014:   unsigned          period_ms;
0015:   int               gpio;
0016:
0017:   void reset(bool s);
0018:
0019: public:
0020:   AlertLED(int gpio,unsigned period_ms=1000);
0021:   void alert();
0022:   void cancel();
0023:
0024:   static void callback(TimerHandle_t th);
0025: };
0026:
0027: //
0028: // Constructor:
0029: //  gpio        GPIO pin to drive LED on
0030: //  period_ms   Overall period in ms
0031: //
0032: AlertLED::AlertLED(int gpio,unsigned period_
      ms) {
0033:   this->gpio = gpio;
0034:   this->period_ms = period_ms;
0035:   pinMode(this->gpio,OUTPUT);
0036:   digitalWrite(this->gpio,LOW);
0037: }
0038:
0039: //
0040: // Internal method to reset values
0041: //
0042: void AlertLED::reset(bool s) {
0043:   state = s;
0044:   count = 0;
0045:   digitalWrite(this->gpio,s?HIGH:LOW);
0046: }
0047:
0048: //
0049: // Method to start the alert:
0050: //
0051: void AlertLED::alert() {
0052:
0053:   if ( !thandle ) {
0054:     thandle = xTimerCreate(
0055:       "alert_tmr",
0056:       pdMS_TO_TICKS(period_ms/20),
0057:       pdTRUE,
0058:       this,
0059:       AlertLED::callback);
0060:     assert(thandle);
0061:   }
0062:   reset(true);
0063:   xTimerStart(thandle,portMAX_DELAY);
0064: }
0065:
0066: //
0067: // Method to stop an alert:
0068: //
0069: void AlertLED::cancel() {
0070:   if ( thandle ) {
0071:     xTimerStop(thandle,portMAX_DELAY);
0072:     digitalWrite(gpio,LOW);
0073:   }
0074: }
0075:
0076: // static method, acting as the
0077: // timer callback:
0078: //
0079: void AlertLED::callback(TimerHandle_t th) {
0080:   AlertLED *obj = (AlertLED*)
       pvTimerGetTimerID(th);
0081:
0082:   assert(obj->thandle == th);
0083:   obj->state ^= true;
0084:
      digitalWrite(obj->gpio,obj->state?HIGH:LOW);
0085:
0086:   if ( ++obj->count >= 5 * 2 ) {
0087:     obj->reset(true);
0088:     xTimerChangePeriod(th,pdMS_TO_TICKS
       (obj->period_
         ms/20),portMAX_DELAY);
0089:   } else if ( obj->count == 5 * 2 - 1 ) {
0090:     xTimerChangePeriod(th,
0091:       pdMS_TO_TICKS(obj->period_ms/20+
          obj->period_
           ms/2),
0092:       portMAX_DELAY);
0093:     assert(!obj->state);
0094:   }
0095: }
0096:
0097: //
0098: // Global objects
0099: //
0100: static AlertLED alert1(GPIO_LED,1000);
0101: static unsigned loop_count = 0;
0102:
0103: //
0104: // Initialization:
0105: //
0106: void setup() {
0107:   // delay(2000); // Allow USB to connect
0108:   alert1.alert();
0109: }
0110:
0111: void loop() {
0112:   if ( loop_count >= 70 ) {
0113:     alert1.alert();
0114:     loop_count = 0;
0115:   }
0116:
0117:   delay(100);
0118:
0119:   if ( ++loop_count >= 50 )
0120:     alert1.cancel();
0121: }
```

```
0010: class AlertLED {
0011:   TimerHandle_t     thandle = nullptr;
0012:   volatile bool     state;
0013:   volatile unsigned count;
0014:   unsigned          period_ms;
0015:   int               gpio;
0016:
0017:   void reset(bool s);
0018:
```

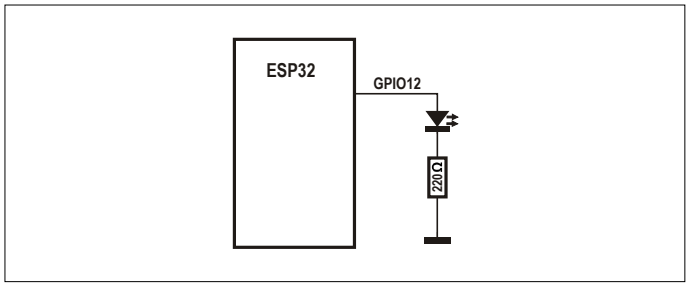


Figure 1. The wiring of the demonstration program *alertled.ino*.

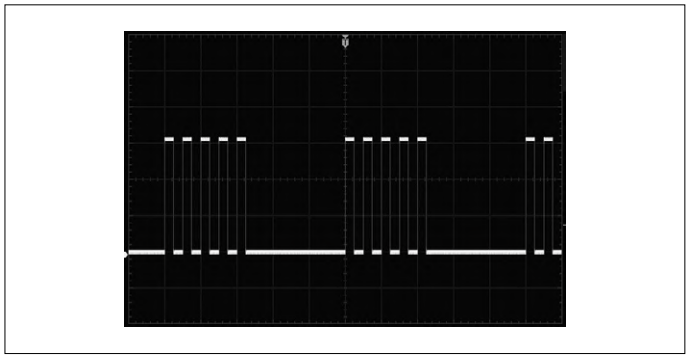


Figure 2. The LED drive signal for *alertled.ino*, horizontal is 200 ms / div.

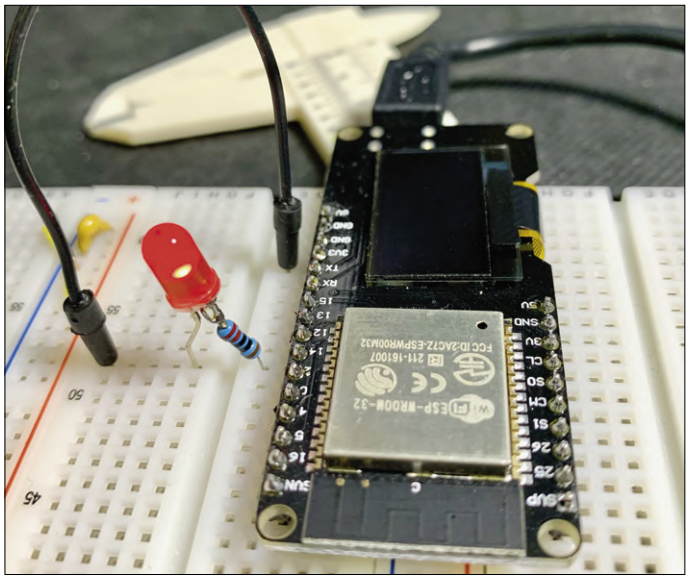


Figure 3. ESP32, LED and 220 ohm resistor wired for the *alertled.ino* demonstration.

```
0019: public:
0020:   AlertLED(int gpio,unsigned period_ms=1000);
0021:   void alert();
0022:   void cancel();
0023:
0024:   static void callback(TimerHandle_t th);
0025: };
```

The class also includes some public method calls including:

- `alert()` - activate the LED indicator (line 21)
- `cancel()` - disable the LED indicator (line 22)

There is also a C++ `static` method named `callback()` in line 24. More will be said about this later.

Method AlertLED::alert()

The method `alert()` and `callback()` are the most interesting parts of the class. We noted that the handle was initialized as `nullptr`. When the `alert()` method is called for the first time, it will find the value to be null in line 53, causing the timer to be created.
The software timer is created by the use of `xTimerCreate()` in lines 54 to 59, returning the handle of the newly created timer. The `assert()` macro in line 60 tests for successful creation. The arguments to `xTimerCreate()` are:

1. A user friendly string name of our timer (here it is simply "`alert_tmr`"). This is not used by FreeRTOS other than for some statistic gathering functions. It normally should be unique but does not have to be if you don't perform lookups by name.
2. A timer period in *system ticks* (converted from milliseconds in line 56 using the `pdMS_TO_TICKS()` macro).
3. This indicates whether the timer is "one shot" (value `pdFALSE`) or an "auto-reload" timer (value `pdTRUE`). Our example creates an auto-reload timer (line 57).
4. This is a "Timer ID" in FreeRTOS terminology. This should be thought of as a "user data parameter". It is a void pointer, which in this example, is the address of the class object using the C++ reserved keyword `this` (line 58).
5. The address of the callback function. In line 59, we supply the name of the static method `AlertLED::callback()`. This is the function that will be called when the timer is triggered.

Some data members are reset by calling internal method `reset()` in line 62, and followed by starting the timer in line 63 using `xTimerStart()`. All timers are created "dormant" until they are started/restarted. Once started, they enter the "running" state.

Static method AlertLED::callback()

The AlertLED class defines a *static* method call named `callback()`. For those unfamiliar with C++, this means that the method is just a function. As such, it has no implied object pointer (no "this" pointer). It is the same as a C function except for the funky C++ name and the fact that it enjoys special access to the internals of the class object.
The software timer callback requires one argument of type `TimerHandle_t` (line 79). This gives us access to the timer handle that is being triggered but we need more informa-

tion - in that case the LED instance which should be driven. Line 80 obtains this information using the "Timer ID" value - the address of the (LED) class instance (`alert1`) using the `pvTimerGetTimerID()` call, which needs the timer's handle. This retrieves the value we supplied in line 58, when the timer was created. We can now get access to the LED object and its members by the variable `obj`.
Lines 83 and 84 toggles the configured GPIO for the LED. Then the value of count is incremented (line 86), and until the count reaches 9, we simply return from the callback. This performs the simple on/off fast blinkling for the first half of the alert period. When the count reaches 9, lines 90 to 93 are executed. The function `xTimerChangePeriod()` alters the time period used by the timer we created. Notice that period_ms/20 is added to the period_ms/2, to compensate for the last dark half of the fast blink time.
Finally, on the 10th time into the callback, lines 87 to 88 are executed. The data members are reset by calling internal method `reset()` (line 87), and the timer period re-established for a fast blink in line 88. Because the data member `count` is cleared to zero, the whole cycle repeats. The auto-reload nature of the timer will keep the pattern repeating until `AlertLED::cancel()` is called.

Method AlertLED::cancel()

To stop the alert from blinking, `AlertLED::cancel()` method is used. To accomplish this, `xTimerStop()` is called (line 71), and the LED is forced off by writing LOW to the LED's GPIO in line 72.

The need for volatile

The alert reader will have noticed that we used the `volatile` C keyword in lines 12 and 13 of the class definition. Why is this necessary? This informs the compiler that whenever it reaches for the data member values `state` or `count`, that it should go directly to the memory of the class instance (`alert1` in our demo) rather than depend upon a value that might still exist in a register. There are two tasks interacting with your class:

- The FreeRTOS daemon class (via the callback)
- Your task that invokes the `alert()` or `cancel()` methods of the class.

If these values were not marked as volatile, either or both tasks might use values that are still in a register, since this is normally the optimal thing to do. However, the values in memory may have changed by the *other* task. For this reason the keyword volatile causes the compiler to disregard this possible optimization and generates code that fetches the values directly from memory instead.

The demonstration

For a simple demo *alertled.ino*, we can again use the Lolin ESP32 OLED Display Module. The wiring is shown in **Figure 1**. The LED is wired according to the active-high configuration.

To prove that the `AlertLED` class can activate and cancel, the `loop()` function maintains a count variable named `loop_count` (line 101). Function `setup()` initially activates the LED, but when the loop count reaches 50, the alert is canceled in line 120. When the loop count later reaches 70, the alert is reactivated in line 113, and the counter reset.
When the program is flashed to the ESP32 and running, you should see the alert LED flashing in an attention getting manner. A little while later it will turn off, only to resume again. **Figure 2** illustrates the LED drive signal.
Figure 3 shows the setup, using a resistor LED combination and a Dupont wire to GND.

FreeRTOS limitations

So far, we have applied the software timer API, without saying too much about some important limitations of the callback. These limitations include:

- Never perform a blocking call within the callback (like `delay()` or a blocking queue push for example).
- Avoid long execution times (this will disrupt the API).
- Avoid using too much stack space (ESP32 Arduino is probably limited to about 1500 bytes). Functions like `printf()` and `snprintf()` often require considerable stack space and thus should probably be avoided in the callback.
- Some of the FreeRTOS API functions operate through an internal queue. This is why our demo program used `portMAX_DELAY` in lines 88 and 92. The timer queue depth for the ESP32 Arduino is 10, which is unlikely to get full, unless many timers are involved at once.

Summary

You might wonder why not simply use a task to manage each alert LED? The main reason for this is that each task requires a stack and a task control block (TCB). If you needed to allocate for 32 LED indicators, this would require 32 x (356 + 1500) bytes, totalling 59,392 bytes! Contrast that to the FreeRTOS timer approach requiring 40 bytes for each timer, for a total of 1200 bytes (object type `StaticTimer_t` is 40 bytes). The stack is shared with the FreeRTOS daemon task for all timers (previously known as the *timer service* task).
Applications might mix and match approaches for special needs but many times a FreeRTOS timer satisfies the requirements. ⏮

200071-01

Web Links

[1] Practical ESP32 Multitasking, Elektor Magazine 1/2020: www.elektormagazine.com/190182-01

[2] Project source code: https://github.com/ve3wwg/esp32_freertos/blob/master/alertled/alertled.ino



SigFox and the IoT (4)

Setting up a dashboard

By **Frank Schleking** and **Bernd vom Berg** (Gernmany)

Our short series of articles closes with a look at the development of a custom dashboard to help visualize measurements sent over the Sigfox network, both graphically and numerically.

In the previous instalments of this series we examined the basic Sigfox hardware for acquiring and transmitting measured data and the concepts underlying data transmission in the Sigfox network. Our measurements are now sitting in the Sigfox backend waiting to be picked up and processed further.

A free web-based dashboard tool is available that lets the user create and operate a custom visualization and monitoring interface on any PC, laptop, tablet or smartphone. And, if suitably enabled, this dashboard can be called up and viewed by anyone (or just by any authorized user) over the Internet from anywhere in the world.

The starting point for the developments described here is the Arduino sketch Sigfox-3_0.ino running on the MKR FOX 1200 board. If you select the fourth menu item 'Send three readings' then the three measured quantities (atmospheric pressure, air temperature, and light intensity as recorded by the LDR) will be continuously acquired, the results being shown on the ePaper display on the motherboard and output over the serial monitor to the Arduino IDE. When the button is pressed these readings will be automatically sent over the Sigfox network, received by the Sigfox backend, and stored. And from that point we can proceed to visualize the data on a dashboard.

The thinger.io dashboard platform

We will be using thinger.io, an open source IoT visualization platform which enables the rapid and straightforward construction of clear and attractive data visualizations. It can even be used free of charge for simpler projects.

The system at the moment does not support mathematical operations on the received data, and that means that the readings must be delivered to it in a form already processed and ready to display. However, the required processing can easily be handled by our Arduino software running on the MKR FOX 1200 board. The model for visualizing the data values is based on a dashboard or instrument panel which can be made publicly available to anyone with a web browser.

Step by step

A few simple steps are required to develop a custom dashboard as follows.

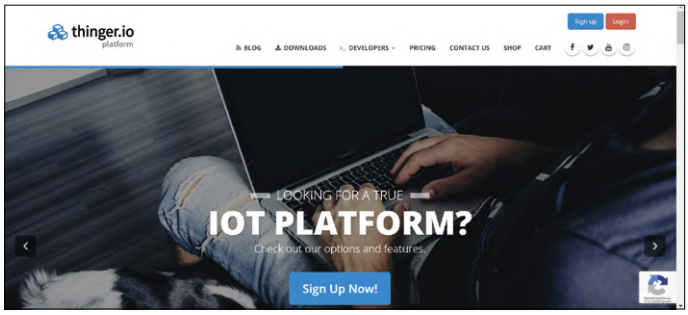


Figure 1. Home page of thinger.io.

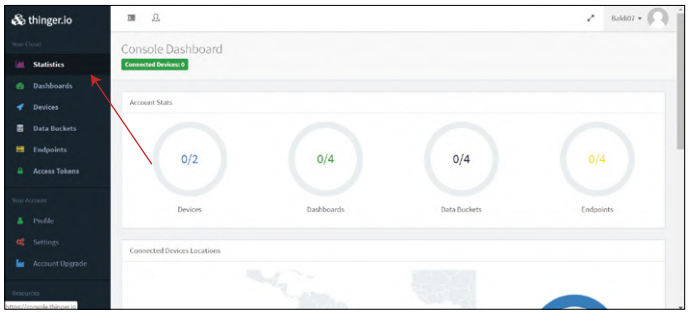


Figure 3. Main account screen showing statistics.

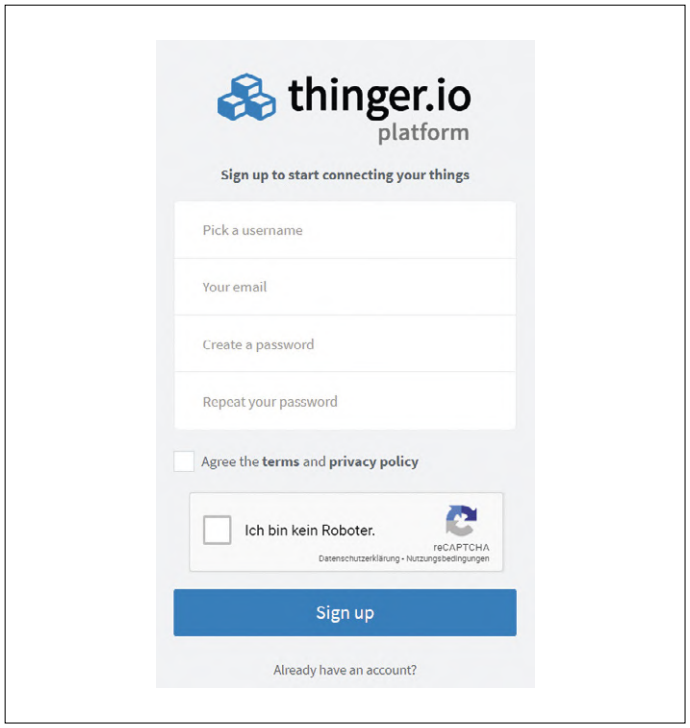


Figure 2. Setting up a new account.

- Create a free user account at thinger.io.
- Set up a 'data bucket' at thinger.io to accept the readings sent from the Sigfox backend.
- Set up an access point and token at thinger.io, and with it configure the Sigfox backend so that the Sigfox backend has the necessary credentials to transfer data to thinger.io.
- Set up a callback on the Sigfox backend to send data over the Internet from the backend to the data bucket at thinger.io.
- Design a suitable graphical dashboard on the thinger.io site to visualize the data.

Creating a user account at thinger.io

A (free of charge) user account has to be set up at thinger.io in order to use its services. To do that, go to the thinger.io homepage [1] (**Figure 1**) and click on 'Sign up' at the top right. This will take you to the form for registering a new user (**Figure 2**). Fill in the fields of the form and click on 'Sign up'.

The services available to users with free accounts are somewhat restricted, but nevertheless are more than enough for most small experimental projects. Full details on what is available can be found under the 'Pricing' tab.

Once you have confirmed your e-mail address you can go to the login page and log in. The main screen of your account is called 'Statistics' and this is the start point for all further actions (see **Figure 3**). The main menu is on the left-hand side, containing all the functions needed to create a visualization dashboard for your IoT project.

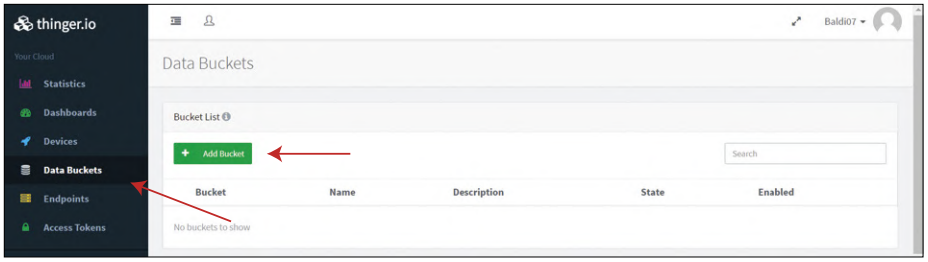


Figure 4. Setting up a new data bucket.

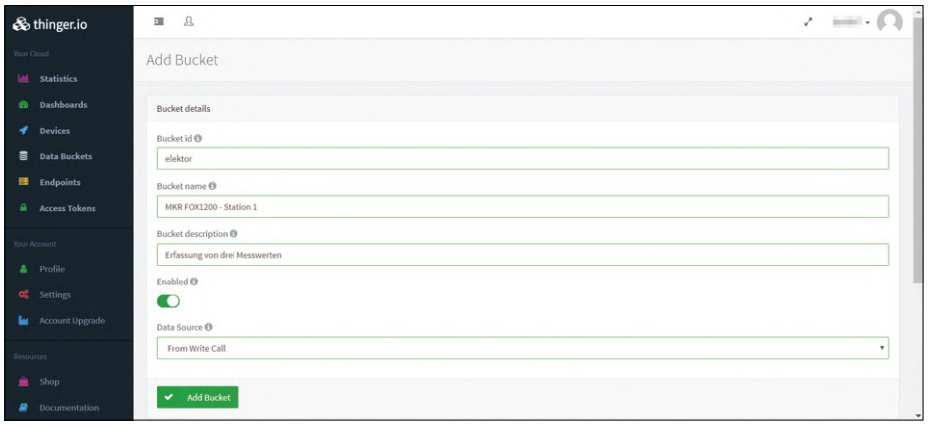


Figure 5. Specifying the properties of the data bucket.

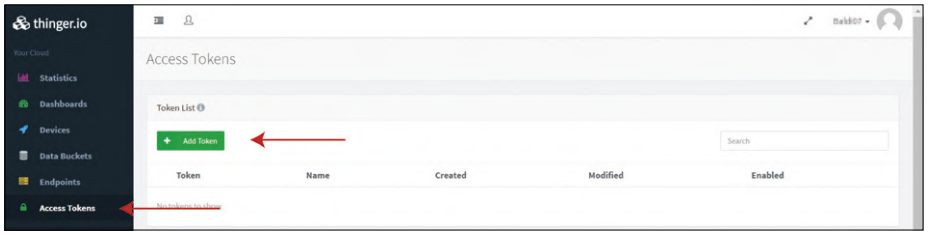


Figure 6. Setting up a new token (access point).

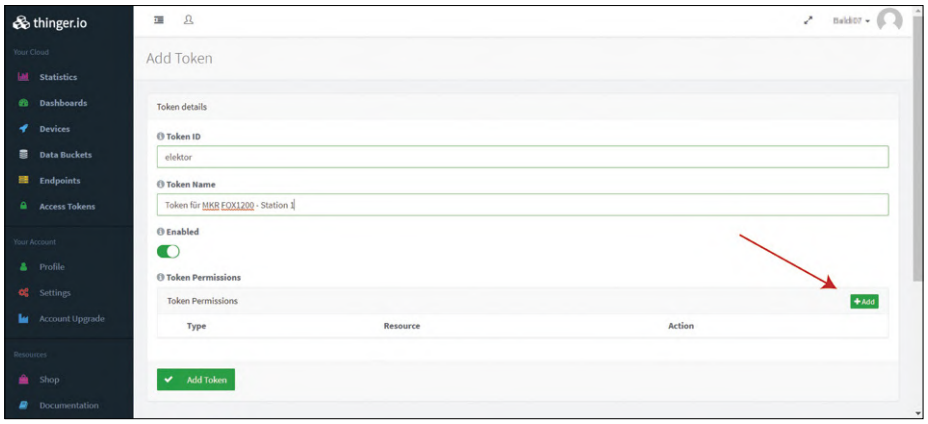


Figure 7. Specifying the properties of the token.

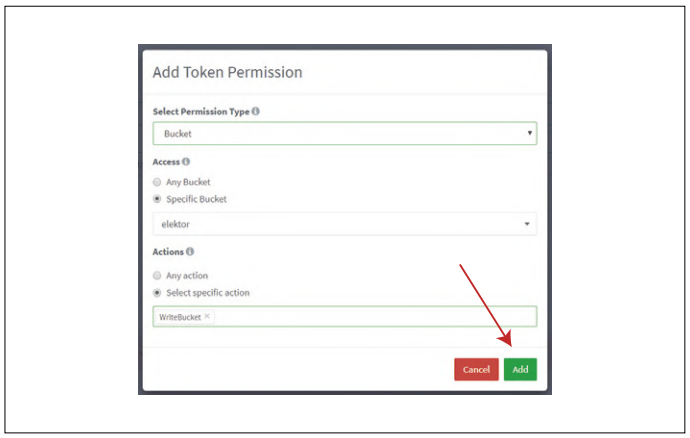


Figure 8. Configuring token permissions.

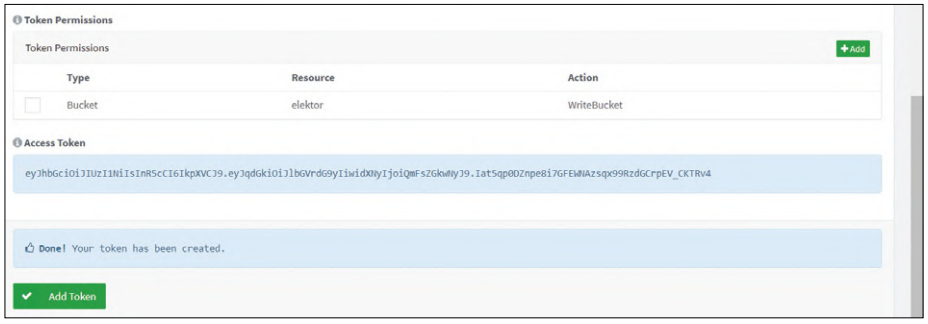


Figure 9. The token is created and activated with the appropriate permissions.

Setting up a data bucket at thinger.io

In the next step we set up a 'data bucket', which is a database where readings can be stored and from which they can later be read back. The Sigfox backend will automatically write readings to this database using the Sigfox callback.

From the main screen click on 'Data Buckets' on the left-hand side and then on 'Add Bucket' (**Figure 4**). A window will appear to let you configure the characteristics of the data bucket (**Figure 5**). The 'Bucket id' is a identifier that must be unique within the whole account, and that will later be used when configuring the bucket to work with Sigfox. For this example we enter 'elektor'. The fields 'Bucket name' and 'Bucket description' can contain any text that might be of help to the user. Use the pull-down menu in the 'Data source' selection field to set its configuration to 'From Write Call'. That means that the data will be sent to thinger.io over the Internet from the Sigfox backend using the Sigfox callback. As soon as the callback sends the data values, they will be stored here in the bucket. Finally, click on 'Add bucket' and the bucket will be created.

Setting up an access point at thinger.io to receive data from the Sigfox backend

The Sigfox backend must of course know where the readings that it has acquired should be sent using its callback: in other words, we have to tell the Sigfox backend the address of the receiver at thinger.io. This step is done by creating a token with the appropriate credentials to allow access to the data bucket. In the main window click on 'Access Tokens' and then on 'Add Token' (see **Figure 6**). A window will appear in which you can enter the properties of the token (see **Figure 7**). First we set the 'Token ID', which can be any value we like: for this example we choose 'elektor'.

Under 'Token Name' you can add a free-text description of the token or just a name.

Then click on 'Add' (the small green button at the bottom right) and then a 'token permission' (i.e., credentials to access the data bucket) will be added. In the next step you will be able to configure this in detail (see **Figure 8**). In the 'Select Permission Type' field select 'Bucket'. In the 'Access' field click on 'Specific Bucket' and then choose 'elektor' (the identifier of the bucket that we set up) from the drop-down menu. Under 'Actions' choose 'Select specific action' and in the field below that choose

'WriteBucket' from the drop-down menu. Finally click on the 'Add' button: this will generate a token for the target bucket that you selected (with identifier 'elektor') and add it to the 'Token Permissions' list (as shown in Figure 7). Now click on 'Add Token' (Figure 7, bottom left) and this access point, which will be the reception address for messages from the Sigfox backend, will be created in thinger.io (**Figure 9**).

The important thing to note is the blue 'Access Token' field, which contains the authentication credentials generated by thinger.io that the Sigfox backend will require.

The Sigfox backend will need to be configured so that it sends this access token code to the thinger.io destination address along with the payload data: this will ensure that the payload it sends will be accepted by thinger.io and correctly directed to the 'elektor' data bucket. It is therefore necessary to copy-and-paste this access code and store it safely somewhere.

With that we complete the configuration of the parameters at thinger.io and we can return to the Sigfox backend.

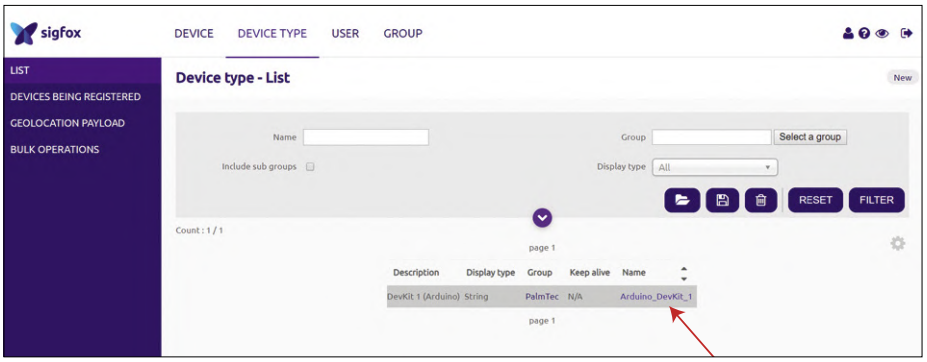


Figure 10. Choosing the desired device type.



Figure 11. The callback window.

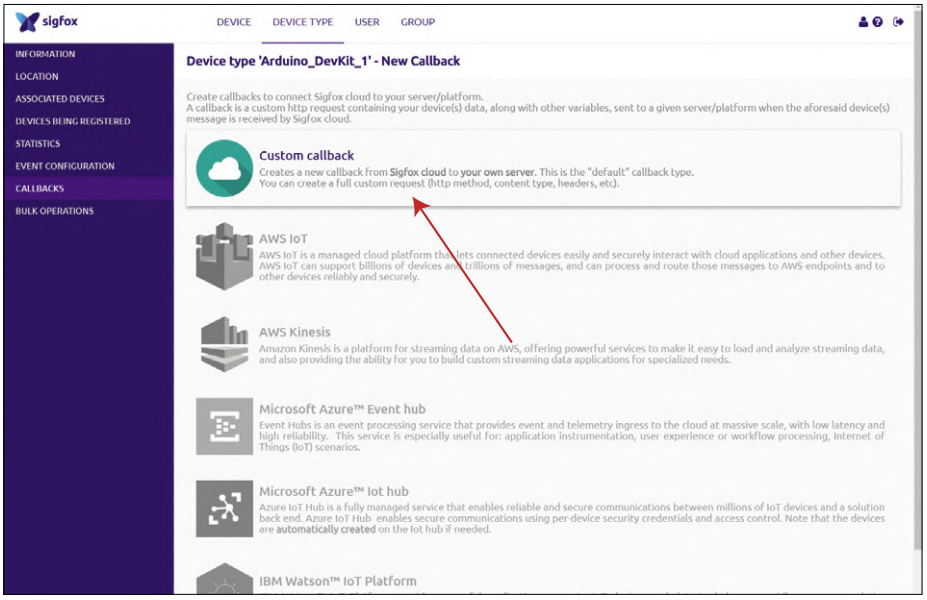


Figure 12. There is a wide selection of possible types of callback.

Configuring the callback in the Sigfox backend

We want to arrange things so that when the MKR FOX 1200 sends data to the Sigfox backend, the information is automatically forwarded to thinger.io for further processing and visualization. To do this we use a feature of the Sigfox backend that we have mentioned a few times before known as a 'callback'. A callback is nothing more than an automatic data forwarder that is executed whenever the Sigfox backend receives data from a Sigfox station such as our MKR FOX 1200. The backend must of course be correctly configured to enable this forwarding, ensuring that the data are sent to the desired recipient.

In order to transfer the data payload to our thinger.io 'elektor' data bucket we need to make use of our token and the access token code that thinger.io has generated for us in the previous step. In other words, the backend will make an HTTP connection over the Internet to access the token, and then uses the access token code to authenticate itself. Once successfully authenticated it can write data into the data bucket.

In order to set up the callback first log in to the Sigfox backend [2]. On the main page click on the 'Device Type' tab and in the list that then appears (the list will only have one entry in this case) click on the device type name 'Arduino_DevKit_1': see **Figure 10**. The overview page for this device will then appear. In the list on the left click on 'Callbacks' (see **Figure 11**). If you now click on the small button marked 'New' at the top right of this window a list will appear giving all the different types of callback available that work with the most widely used dashboard and cloud programs (**Figure 12**). From these select 'Custom callback'. In the window that now appears you will be able to configure the callback and how data are sent from the Sigfox backend to thinger.io (see **Figure 13**). Note that at this point it is essential to take care that all the entries you make on this page appear **exactly** as described



Figure 13. Configuring the callback.



Figure 14. Starting to create a dashboard.

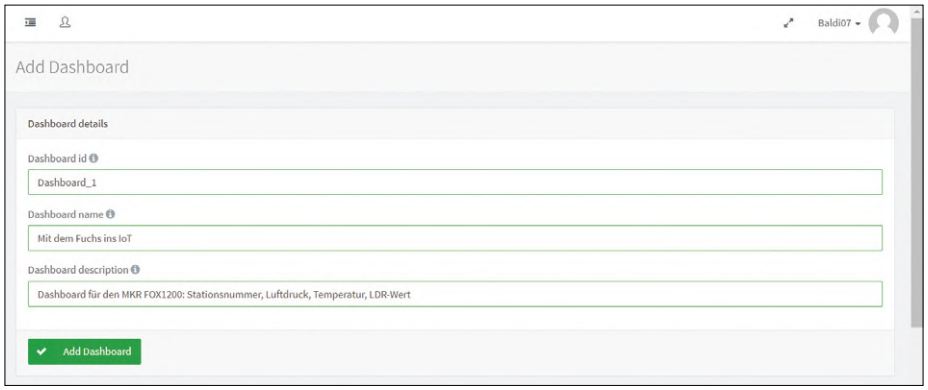


Figure 15. Basic parameters of the dashboard.

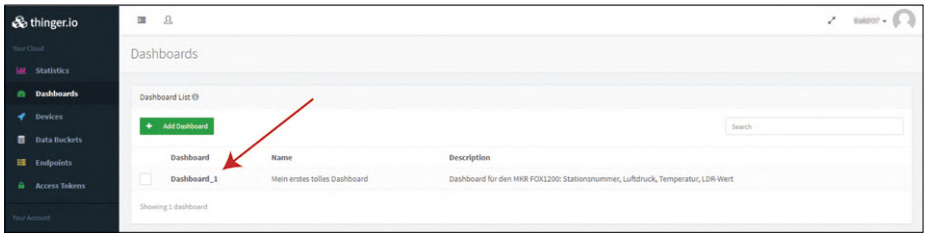


Figure 16. List of available dashboards.

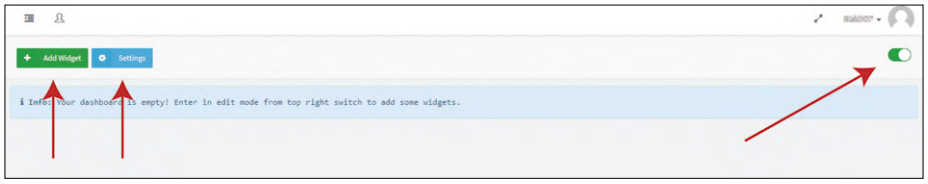


Figure 17. The dashboard in edit mode.

below: the tiniest mistake will result in error messages and indeed the whole system not functioning correctly.

A detailed description of how the individual settings work is beyond the scope of this article: very comprehensive information can, however, be found at [3], [4] and [5]. Let us begin at the top of Figure 13. Under 'Type' the correct settings are 'DATA' and 'UPLINK', and for 'Channel' select 'URL', as it is over an HTTP connection that we will be talking to thinger.io.

Now things get a bit more tricky. The data values sent from the MKR FOX 1200 in the payload to the Sigfox backend need to be unpacked back into individual values within the Sigfox backend. Internally to the backend each value is assigned a freely-chosen variable name. These names are used in specifying the further processing that the backend does on the variables' values. The way the Sigfox payload is unpacked is specified in the 'Custom payload' field, and of course must mirror the way the data values are packed into the payload on the MKR FOX 1200. We need to specify precisely the sequence of data in the Sigfox payload, the data types involved, and the storage format, in order for the payload to be unpacked correctly into the individual variables. For our example the 'Custom payload config' should contain three elements, each appearing **precisely** as follows.

```
temp::float:32:little-endian
pressure::float:32:little-endian
ldr::uint:16:little-endian
```

The first data value packed into the payload sent by the MKR FOX 1200 is therefore the temperature reading. In the Sigfox backend this is given the name 'temp' and has data type 'float', represented using four bytes (32 bits).

The second data value in the payload is the atmospheric pressure, also a 'float' variable represented using 32 bits.

The third and final value in the payload is the LDR voltage. In the backend this is given the name 'ldr', and is a variable of type 'uint' stored in two bytes (16 bits).

The term 'little-endian' specifies the order of the bytes used to store a given value: little-endian is the order

used by our Atmel microcontroller. Further information on this can be found at [6]. You should enter the above three elements, concatenated into a single line and separated by spaces, into the custom payload configuration field. You can now configure the Sigfox backend to process these three variables, or, alternatively, you can send them directly to thinger.io in this form. Under 'URL pattern' you specify the destination URL at thinger.io. The URL is constructed in the following way.

```
https://api.thinger.io/v1/users/{user_id}/buckets/
   {bucket_id}/data
```

In the address replace `user_id` with the user name from your thinger.io account (from Figure 2) and `bucket_id` with the bucket identifier, here 'elektor' (from Figure 5), in both cases without curly brackets or quotation marks.
Under 'Use http Method' select 'POST'.
The next step is to set up authentication for the data transfer to thinger.io. This is done in the two fields next to 'Headers'. In the 'header' field enter the word 'Authorization' and next to that, in the 'value' field, enter 'Bearer', followed by a space and then the access token that we created as shown in Figure 9.

```
Bearer eyJhbGci7j76j67jzI1NiIsIn ... etc.
```

In the field 'Content type' enter 'application/json', which indicates that the body of the message is in JSON format.
In the final step we specify exactly what will be sent to thinger.io. The entry (in JSON format) in the 'Body' field should be as follows. Again, it is essential that the text appears **precisely** as shown.

```
{
    "Device-ID" : "{device}",
    "AtmPressure" : {customData#pressure},
    "Temperature" : {customData#temp},
    "Light-Sensor" : {customData#ldr}
}
```

The 'Body' is enclosed in curly brackets ('{', '}'). But what does the text in the body mean?
Each line between the curly brackets begins with a double quotation mark; after that appears an freely-chosen (but unique) variable name, followed by another double quotation mark. There follows a space, a colon and another space. It is at this point that the value of the variable will be inserted by the Sigfox backend. The variable name and its value will subsequently be sent in this format to thinger.io, where it will be stored in the appropriate data bucket.
So we have

```
    "Device-ID" : "{device}",
```

which will give the unique device number of the Sigfox station (i.e., our MKR FOX 1200), which is known to the backend. This information is sent along with the measured data so that the originating station can later be identified uniquely.
The line

```
    "AtmPressure" : {customData#pressure},
```

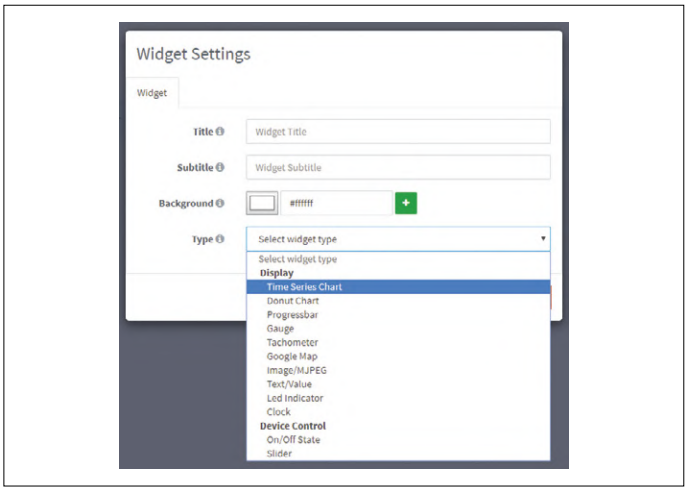


Figure 18. First window for creating a widget.

sends a variable with name `AtmPressure`. The value of this variable corresponds to the value of the backend variable called `pressure`, which we previously specified in the 'Custom payload config' field. The keyword `customData#` indicates that the value of a variable unpacked using the 'Custom payload config' field is to be inserted. This is how the variable names and values are reformatted for transmission to thinger.io.
The last two lines are processed in the analogous way.
Each line, apart from the last, must be terminated with a comma.

That completes the configuration of the callback in the Sigfox backend to enable it to send the desired data to the 'elektor' data bucket at thinger.io. When you have completed all the above steps correctly, you can click on 'OK'.
Now, when the MKR FOX 1200 sends a payload to the Sigfox backend, and assuming that no errors are encountered in reception, the Sigfox backend callback will immediately forward the data in the format specified above to thinger.io for further processing.

The dashboard at thinger.io

Now we come to the last piece of the puzzle, adding some polish to our project. We want to present the data in graphical form on a dashboard. In order to create a dashboard, go to the main thinger.io console (Figure 3) and on the left click on 'Dashboards' and in the next screen on the 'Add Dashboard' button (see **Figure 14**).
You now need to fill in the fields in the 'Dashboard details' window (**Figure 15**). The 'Dashboard id' is an identifier unique across the entire account. The fields 'Dashboard name' and 'Dashboard description' can be filled in with any information: they are intended to be easily user-readable. Confirm the entries by clicking on 'Add Dashboard'. The new dashboard will be added to the 'Dashboard list' which you will see whenever you click on the 'Dashboards' menu item in the main console window (as shown in **Figure 16**).
If you click on the name of the dashboard, a design window will appear displaying an empty dashboard (see **Figure 17**). At the top right there is a small switch: slide this to green to enter edit mode. There are also two buttons labelled 'Add Widget'

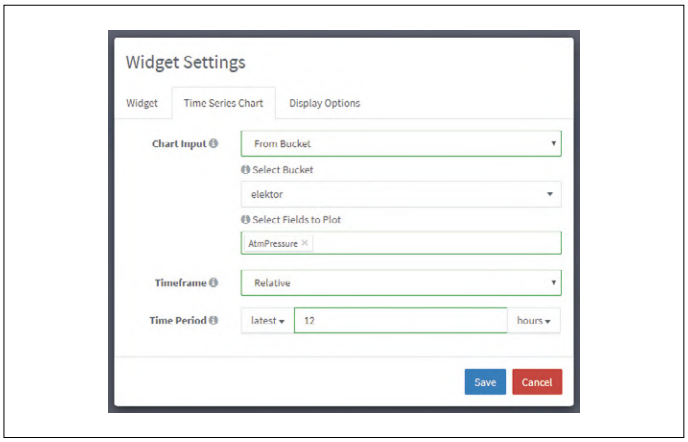


Figure 19. Configuring a time-series chart.

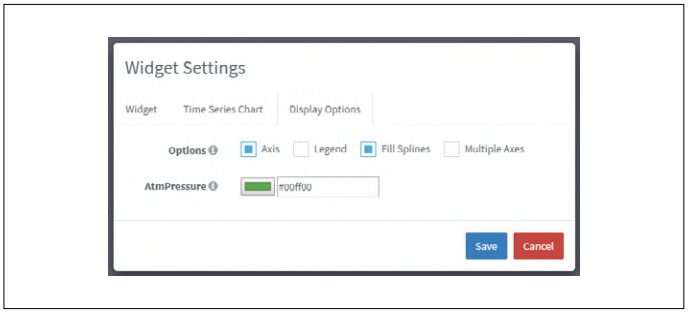


Figure 20. Setting the basic characteristics of the display.

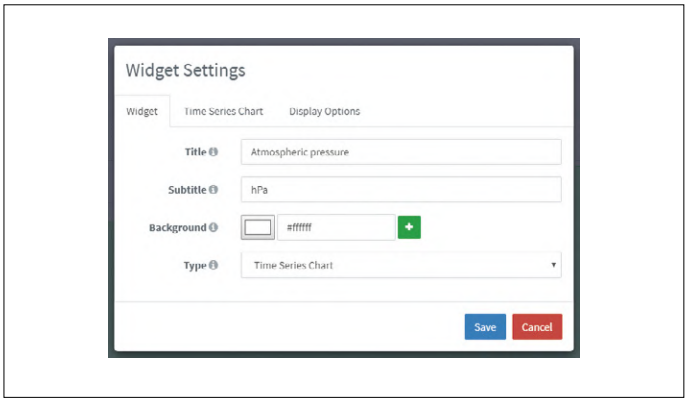


Figure 21. Settings in the 'Widget' tab.

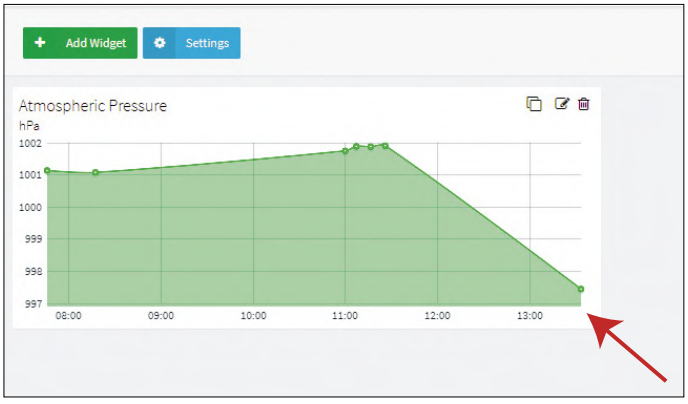


Figure 22. The time-series chart added to the dashboard: a few readings are already plotted.

and 'Settings'. In thinger.io the dashboard display elements that form part of the user interface are termed 'widgets'. A wide range of widgets is available, including time-series charts, doughnut charts, progress bars, analogue displays, Google maps with position information, images, text displays, LEDs and a clock. In our application we would like to show a time-series chart for each of the three measured values and a digital display for each showing the most recently received value.

Click on 'Add Widget' and a window entitled 'Widget Settings' will pop up (see **Figure 18**). Under 'Type' you can choose the type of widget you would like to add. In this case we choose 'Time Series Chart'. Two further tabs will appear, as shown in **Figure 19**. In the 'Time Series Chart' tab, under 'Chart Input', select the 'From Bucket' option. As you proceed more and more fields will appear that need to be filled in. In the 'Select Bucket' field choose the bucket 'elektor'. In the next field, 'Select Fields to Plot', choose the data value that is to be displayed.

We begin with 'AtmPressure'. Under 'Timeframe' choose 'Relative' and set the time window to be displayed to '12 hours'. This will cause the graph to display the most recent 12 hours' worth of data; of course this can be adjusted if desired. Click now on the 'Display Options' tab (**Figure 20**) and turn off the 'Multiple Axes' option: we are going to set up a separate graph for each of our variables, and so we do not require multiple Y-axes on the same graph. The 'Axis' option (already enabled) causes the X- and Y-axes of the graph to be scaled automatically. In the field next to the 'AtmPressure' label we can choose the colour in which the graph will be plotted: here we have gone for a soothing shade of green. If you enable the 'Fill Splines' option the region between the plot and the X-axis will be filled with the colour selected next to the 'AtmPressure' label. The 'Legend' option allows the plots to be labelled separately. Of course this is really only useful when more than one time series is plotted on the same chart.

Finally we return to the 'Widget' tab (see **Figure 21**), where we can now set the title and subtitle for the chart. We will set the 'Title' field to 'Atmospheric pressure' and the 'Subtitle' field to the unit of measurement, 'hPa'. If desired the 'Background' setting lets you adjust the background colour of the whole chart; however, we do not want to over-egg this particular pudding and will stick with the standard white. We finish setting up the chart by clicking on 'Save': the chart will be added to the dashboard (see **Figure 22**). You can make the chart larger or smaller by clicking in the bottom right-hand corner, and the mouse can also be used to drag the chart around the dashboard area. Resizing the chart is not perfectly smooth, and so it will tend to jump around a little.

We also want to display the current atmospheric pressure reading in digital form (**Figure 23**). The 'Add Widget' button again opens the menu for adding a new element to the display; in this case in the 'Type' field we select 'Text/Value'.

In the 'Widget Value' field we again select 'From Bucket', choose the 'elektor' bucket and again specify which value we want to display (here 'AtmPressure'). For the 'Bucket Value' we set 'Latest Value'. The 'Display Options' tab allows us to make various configuration settings. For example, we can set the precision to which the value is displayed ('Decimal Places') and we can set the 'Units' field to 'hPa'. This string will be displayed in the widget next to the measured value. There are further configuration options that can be set if required, but we will

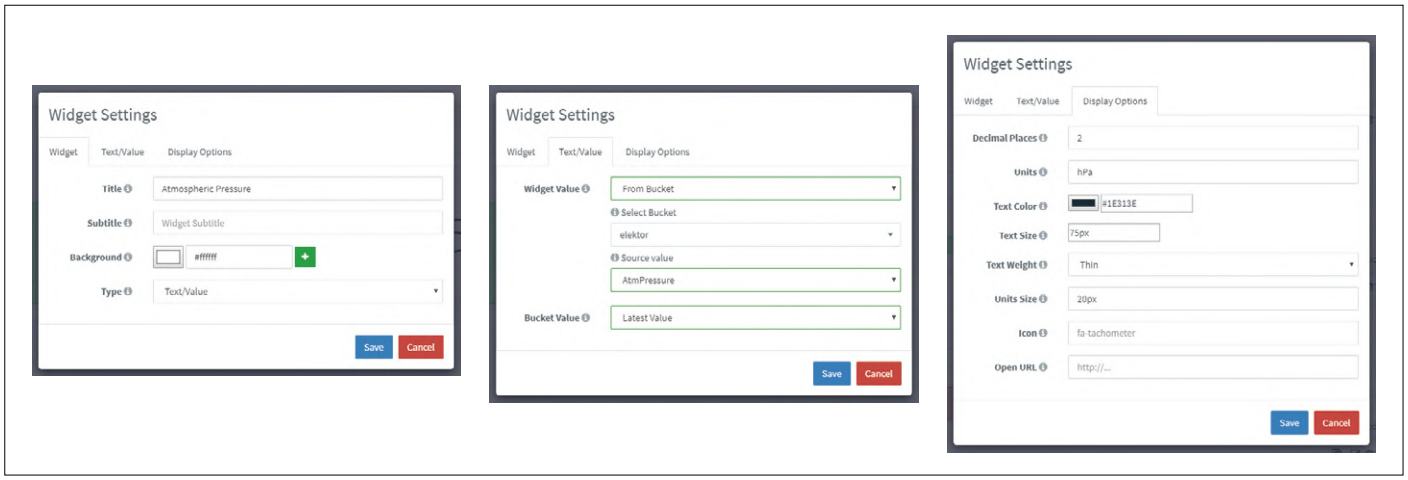


Figure 23. Configuring the digital display for the atmospheric pressure reading.



Figure 24. The completed dashboard.

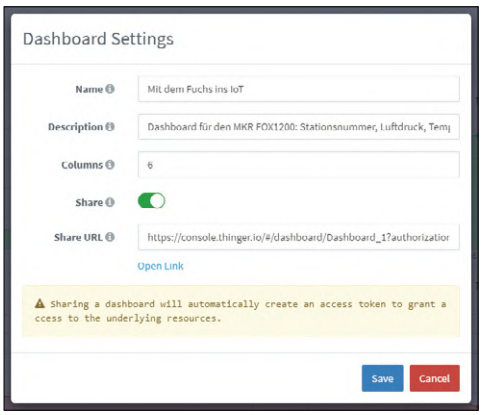


Figure 25. General dashboard settings.

not explore them further here. In the 'Widget' tab we again set the 'Title' field to 'Atmospheric pressure'. We can leave the 'Subtitle' field empty, as we have already arrange for the unit to be displayed next to the value.

That completes the configuration of the dashboard display elements for atmospheric pressure. The display for the other two readings are configured in a similar way: the final result is shown in **Figure 24**. Note that in order to ensure that all entries and modifications are stored in thinger.io, it is necessary to exit editing mode and then re-enter it. More information on designing dashboards on thinger.io can be found at [7]. Now we just need to enable access to the dashboard 'from outside'. Back in the Dashboard menu click on the blue button labelled 'Settings', which will take you to the dashboard settings

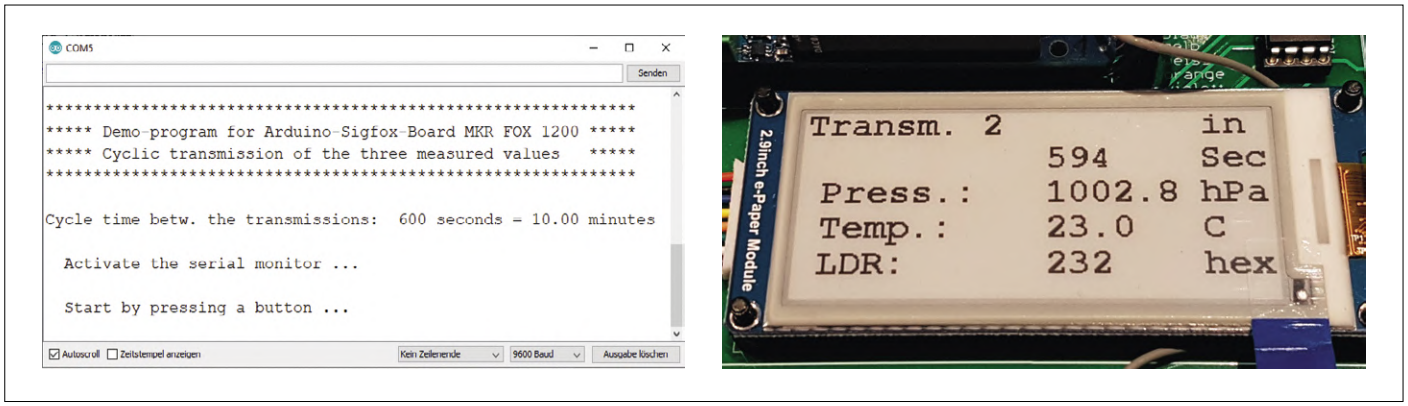


Figure 26. Output from Sigfox-Demo.ino on the serial monitor and on the ePaper display.



Figure 27. The dashboard looks good on a smartphone screen!

page (see **Figure 25**). Here you can give your dashboard a name and a description (although in this case it is not necessary as we have already set this up). The switch labelled 'Share' enables the dashboard for external access: the (rather lengthy) link required to access it will appear in the field labelled 'Share URL'. Using this link it is now possible to access the dashboard from anywhere in the world and examine the data it displays. Because of the length of the URL it is recommended to use a URL shortener such as bitly.com to abbreviate it. The shortened URL can then be converted into a QR barcode that can be scanned by a smartphone: this avoids the bother of having to tap in the long URL.

It is worth noting that when you enable public access to the dashboard at thinger.io it is automatically write protected: that is, it can only be read. Third parties cannot alter settings or make other adjustments: this is only possible after using the appropriate credentials to log in to the thinger.io portal.

We would now like to test the system as a whole. However, it is somewhat annoying to use the Sigfox-3_0.ino Arduino sketch to do this, as it is necessary to press a button each time we want to send a message. To help out there is another sketch, called Sigfox-Demo.ino, included in the software download [8]. At ten-minute intervals this program automatically acquires the three readings and transmits them (see **Figure 26**). The interval between transmissions is determined by the value of the global variable w_zeit and can be adjusted if desired, so you can arrange to acquire and display readings over a long period of time. However, always keep in mind that you are only allowed to send a maximum of 140 messages per day on the Sigfox network, which corresponds to about one message every ten minutes. In view of that it is less critical that a free account at thinger.io only entitles you to write data to a bucket at most once a minute. **Figure 27** shows how natty the dashboard we have created looks on the screen of a smartphone.

In conclusion

With that we come to the end of our short series of articles introducing the world of the Sigfox 0G network. The highly flexible and versatile programming of the Arduino MKR FOX 1200 enables the acquisition of a broad spectrum of types of data and their transmission over the Sigfox network all over the world. Meanwhile the simple yet powerful visualization possibilities offered by thinger.io allow the rapid and efficient development of custom dashboards for communicating information to end users.

We wish you every success on your forays into the exciting world of the Internet of Things! ⏮

190281-D-02

4 SALE @ WWW.ELEKTOR.COM
→ Arduino MKR FOX 1200
www.elektor.com/19096
→ Arduino Antenna 868 MHz
www.elektor.com/19095

Web Links

[1] Thinger: http://www.thinger.io/
[2] Sigfox backend: https://backend.sigfox.com/
[3] Information on configuring Sigfox callbacks: https://support.sigfox.com/docs/callbacks-documentation
[4] Information on creating custom Sigfox callbacks: https://support.sigfox.com/docs/custom-callback-creation
[5] Information on the Sigfox callback API: https://backend.sigfox.com/apidocs/callback
[6] Byte ordering when storing numeric values: https://en.wikipedia.org/wiki/Endianness
[7] Information on designing a dashboard: https://github.com/thinger-io/Docs/blob/gh-pages/console/dashboards.md
[8] Project web page: www.elektormagazine.com/190281-D-02

Start-Up Update: The Elektor Investment Program

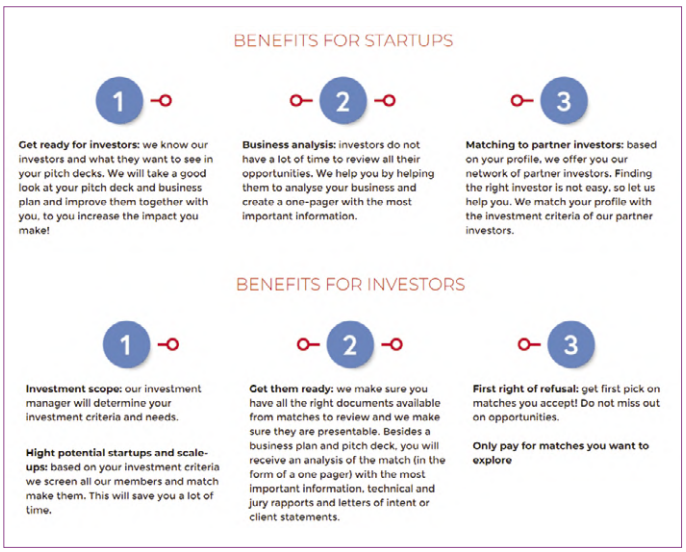
By the Elektor Start-Up Team



Elektor nurtures and empowers electronics-focused start-ups (e.g., hardware, software, and technical services) through events, competitions, and business solutions. The Elektor Investment Program [1] is a new service for both young companies and investors looking to connect and excel in the electronics industry.

What is the Elektor Investment Program?

As a start-up or scale-up, there comes a time when bootstrapping is no longer an option. If you are lucky, you already have some paying customers, but you will need money to take your business to the next level. Maybe you need to hire a team, enter new markets, or extend your development. The most likely step is to talk with investors. As a founder, you might ask yourself: Am I ready to pitch to them? Is my business plan and pitch deck ready? Will investors take me seriously? Members of the Elektor Investment Program team will guide you through this venture. We can help with your business plan, assist you in preparing your pitch deck, and match you with investors from our network.



Why did Elektor start the program?

If you simply send your pitch deck to investors and hope for the best, you could get declined for any number of reasons. Perhaps you contact the wrong investor. Or maybe your pitch deck lacks important information. One thing is for sure: you need to make the most of your pitch opportunities.

So, why did we set up the Elektor Investment Program? We've been running start-up competitions for a few years [2], and we've been interacting with many promising start-ups all along the way. Developing a program to further help start-ups find funding and partners was a logical next step.

We also aim to assist investors. We know that investors get dozens of pitch decks each year. And it takes a lot of time to read them all, and most of them are not good enough. We realized that we could help investors save time by connecting them with promising tech start-ups in our global network that match their investment scope.

Start-Up benefits

Are you a member of a start-up or even scale-up in need of funding? By participating in the Elektor Investment Program, you can take smart actions to improve your chance of getting funded. Sounds like a good deal, right? Here's what we offer:

- Personal help to improve your pitch deck and business plan.
- Business analysis to make a profile of your company.
- The opportunity to be connected to matching investors (based on your profile) from our network.

Investor benefits

The Elektor Investment Program can help investors find innovative companies. In addition to running successful start-up competitions, our professional start-up team scouts for promising companies around the world. Companies that join the program are screened by our experts, so we assist investors with promising leads. We provide:

- Access to high-potential technical start-ups and scale-ups that match your investment scope.
- Proper documentation, professional business connections, and a business analysis of your matches.

Join the Elektor Investment Program!

Want to join the Elektor Investment Program or learn more about it? Visit the program site [1] or email Arthur Lampe (Elektor's Investment Program Manager) today on **arthur.lampe@elektor.com** to get started. Good luck!

200112-01

Web Links

[1] Elektor Investment Program: www.elektormagazine.com/widgets/investment-program

[2] Elektor Start-up Competitions: www.elektormagazine.com/select/fast-forward-award

REVIEW

Joy-IT HD35 Electronic USB Load

Test & verify USB power performance

By **Dr Thomas Scherer**

Since its introduction over 25 years ago the USB port has become the *de-facto* standard for communications between a PC and its peripheral equipment. In the meantime, in an effort to reduce the plethora of non-standard wall-wart mains adapters, most other devices with built-in rechargeable batteries like mobile phones, tablets etc. use the 5 V supply pins of this port for charging. It's even possible to fit a mains outlet wall plate with a built-in 5 V supply and USB socket. With the introduction of Quick Charge the power supply communicates with the device being charged to negotiate an output voltage. A compliant charger can output a higher voltage and current beyond the typical 5 V / 2 A rating of the standard USB charger. With so many essential devices relying on a fully-functioning charger and so many different fast-charge standards emerging it is convenient to have a single unit that can quickly check that the charger is doing everything it should...



When I saw this little device called the JT-HD35 from Joy-IT on the Internet, I was curious. An electronic load for the USB port? Another crazy idea dreamt up in the Far East! But as I read in more detail what this test unit can do, it all became a bit clearer. It can be used to test the performance of all types of USB ports. This is not only useful to test a port designed into a product for production but can also be used to ensure that an off-the-shelf charger really can provide the power quoted in its spec sheet. I have



Figure 1. This small plastic box contains the JT-HD35 USB load tester.



Figure 2. The plastic box from below showing technical data and the manufacturer's URL.

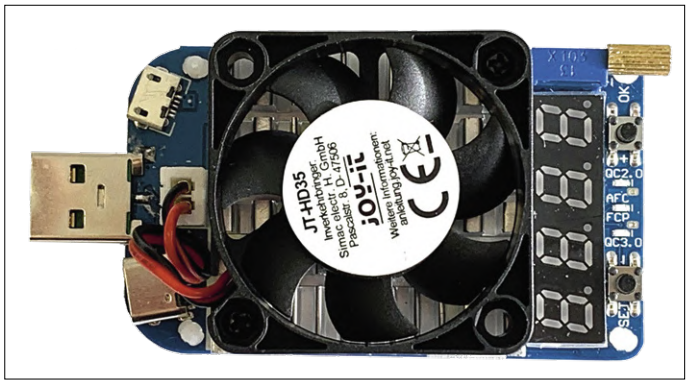


Figure 3. The USB load module in all its glory. It's all there: three USB inputs, fan, display, three LEDs, three pushbuttons (the third is under the board) and a potentiometer.

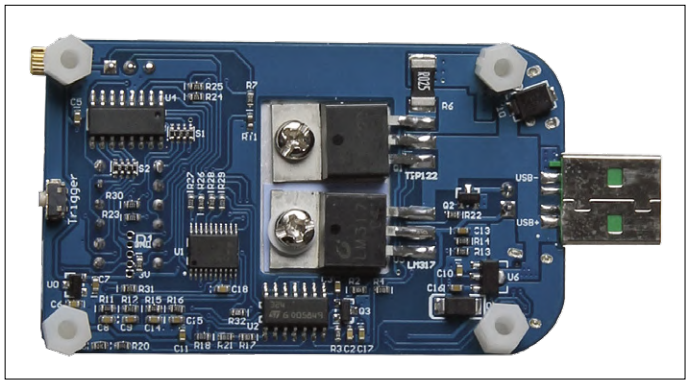


Figure 4. Underside of the USB load module (Image. Joy-IT).

already reviewed another USB test device from Joy-IT[1] called the UM25C — but this one here is a different kettle of fish with different capabilities.

Features

The Joy-IT JT-HD35 is a compact, electronic USB load resistor able to measure and display the output power up to 35 W from a USB port or charger. It can test the output of a standard 5 V USB charger as well as those compliant with some of the more recent fast-charge standards.
The key features of this unit are:

- Operating voltage: 4 to 25 V (DC)
- Load current: 0.25 to 5 A
- Power: max. 35 W
- Indicators: 3 status LEDs (green, blue, red)
- Display: 4-digit 7-segment display LED, red
- Operation: 3 buttons, 1 multiturn potentiometer (load current adjust)
- Connections: USB 2.0, Micro-USB, USB-C
- Fan: 40 mm, 0 to 8000 rpm, temperature controlled
- Dimensions: 84 x 41 x 28 mm
- Weight: 52 g

Unboxing

The load tester consists of a single board and is supplied in a translucent plastic box (**Figure 1**) with the picture on the lid showing what is inside. Apart from the USB load tester board there is no documentation or links to any sites where the documentation can be downloaded. The underside of the box is printed with the main features of the load tester but again no links (**Figure 2**). This is no great loss because it's no problem to Google the company name and find the user manual for this device in PDF format [2].
I prefer to store this type of document in an 'Instructions' folder on my PC; it saves tedious searches by hand — and a PDF is ideal as it can be searched for keywords in most cases. First of all, I was keen to see how far I could use the device without resorting to the manual.
Figure 3 shows what's in the small plastic box. The unit consists of an assembled PCB with a heatsink and fan fixed in place using only two screws. On the underside shown in **Figure 4** you can see the TIP122 power Darlington and an LM317 regulator which are mounted on the underside of the heatsink. The NPN Darlington transistor is rated at 65 W max. and is probably used as the variable load to dissipate the power. Also on this side is a microcontroller labelled U1 = N76E003AT20, which is

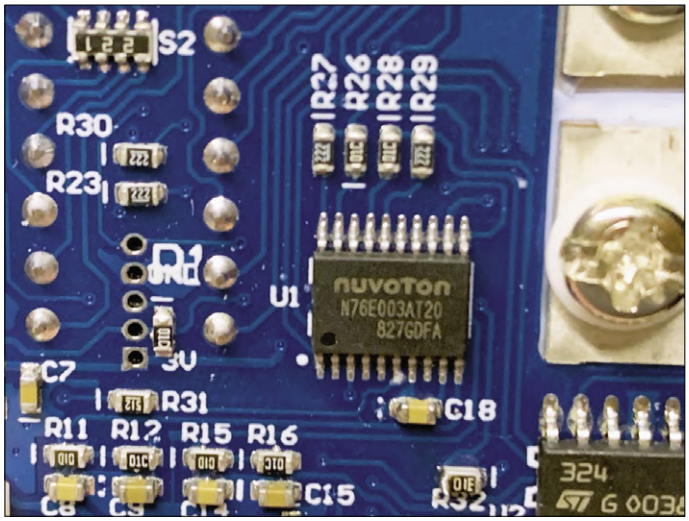


Figure 5. This close-up clearly shows the chip's identification marks.

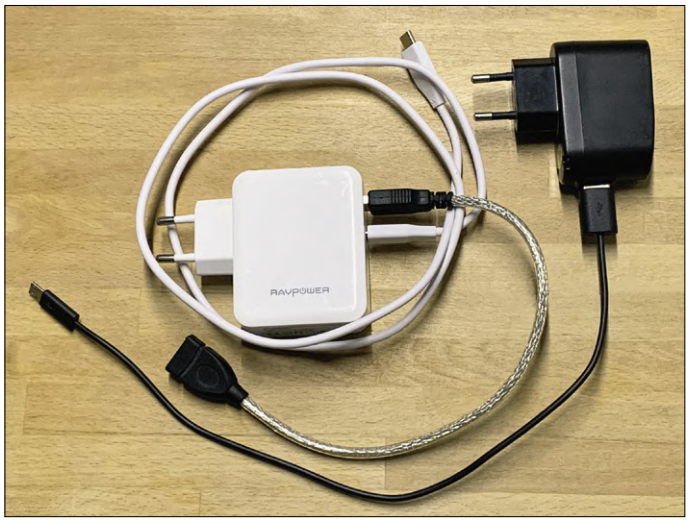


Figure 6. These two power supplies are about to be put through their paces.

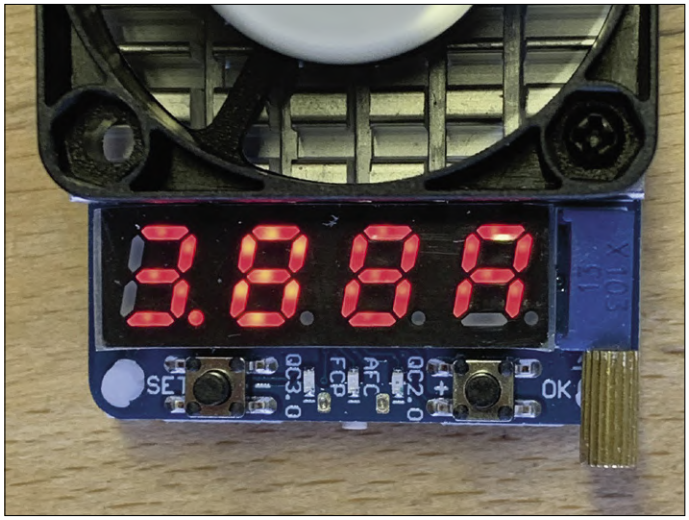


Figure 7. Proof: The RAVPower QC 3.0 compliant charger can supply 3.88 A via USB-C at 5 V.

a small extremely low-cost (€0.28) 8051 derivative. Spaced out across the board are a sprinkling of peripheral chips and passive components. Because the writing on these tiny SMD components is not easy to read, I usually take photos of them and enlarge the result (see **Figure 5**).

Testing, testing...

I selected two power supplies/chargers with three cables as guinea pigs (**Figure 6**) for the USB load tester. The little black one on the right with the micro-USB to USB-A cable is a cheap standard charger with 5.1 V and 1 A load capacity, as it used to come with many (old) smartphones. It cannot deliver anything much more than five watts. The white 'smart' power charger from RAVPower is surrounded by a USB-A extension cable and a USB-C cable. It supplies 5 V at 2.4 A from its USB-A socket and not only 5 V at 3 A from the USB-C socket, but also 9 or 15 V at 2 A or 20 V at 1.5 A thanks to its compliance to QC 3.0. The total power output amounts to 36 watts, which ties in nicely with the rated power handling of the JT-HD35.

After switching on, when connecting to the black charger, the voltage is shown in two digits on the display, i.e. about '5.1 V'. After a little erratic operation of all the pushbuttons, I think I got some idea how this thing works: pressing the 'SET' button cycles the display to show voltage, power (initially '0.0P') and current (here flashing '1.00A'). Now if you press the OK button once, the current setting value stops flashing and remains constant. Then after about 20 s the cooling fan kicks in. This indicates that when the displayed current is flashing it shows the required value of load current but without the load connected. When the displayed current is continuous it indicates that the load is connected. Another press of the 'OK' button removes the load and sets the display value flashing again.

By adjusting the multiturn pot setting while testing the small black charger under load I increased the load current to 1.1 A and noticed that the output voltage dropped to 3.5 V. It was reassuring to find that the charger has some built-in over-current protection. It was also interesting to see that micro-USB connector is unsuitable to deliver high levels of current. Even with a current limited to 1 A the voltage drop across the connector resulted in a measured supply at the load of 4.9 V, whereas the USB-A extension cable used with the same power supply delivered 5.0 V.

Now with the USB-A socket of the RAVPower unit connected to the load tester via a USB-A lead its current limit starts to activate at around 3 A. Connected via the USB-C cable, it was possible to crank up the load current to 3.88 A before the power supply shut down (see **Figure 7**). It looks like the technical spec for the RAVPower charger is quite modest when it comes to its maximum output current rating.

And now?

After testing it with 5 V USB supplies I wanted to move on and try various other voltages according to QC 2.0 or 3.0. At first I wasn't sure how I could change the mode but after searching through the manual it became obvious I had overlooked a pushbutton on the underside of the board. This is the 'trigger' button — used to activate the curiously named 'trigger/decoy' feature. I guess the JT-HD35 could be thought of as a kind of decoy device which sends appropriate acknowledgements to the charger, to make it switch its output (surely 'Mode' would

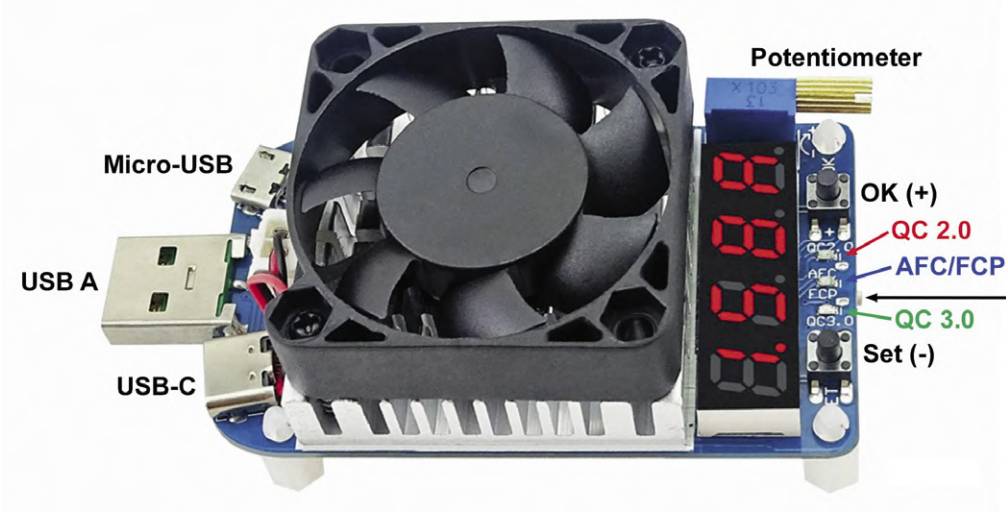


Figure 8. Connections, displays and controls of the JT-HD35 (image modified from Joy-IT original).

be a better label). I added clearer labels to identify the button in the photo shown in **Figure 8**.

A long (2 seconds) press of this pushbutton switches the unit out of its default (5 V) to the other modes. The display should first show '-5.0' to indicate it is in QC 2.0 mode. The red LED flashes and you can use you the '+' and '-' buttons to adjust the charger voltage output in steps through 5/9/12/15/20 V. Further presses of the trigger push button get into the other modes including 'AUTO' which can identify if the charger has a quick charge function — in this mode message exchange can take up to 25 s to complete. Although USB was originally conceived as a universal interface to reduce e-waste, various companies such as Huawei, Samsung and Apple now offer fast charge schemes which are not interoperable. A recent vote in the European parliament however returned a massive majority in favour of making iPhones and Androids use the same charger — this could mark the end for Apple's Lightning Cable in Europe when the law comes into force in the summer.

According to its specification the JT-HD35 can provide load tests for chargers designed for standard QC2.0, QC3.0, Huawei's FCP fast charge system, Samsung's AFC9 and 'other quick charge modes' although it is not specific what these other standards are. Without any of these quick chargers to hand I was unable to test this feature of the device.

Without reference to the manual, I certainly struggled to change the operating mode. Operation of this electronic USB load was (for me) not entirely intuitive. For my needs I am only seldom likely to make use of a test device like this. Chances are when I come to use it again in three months time I will have forgotten how it works.

Conclusion

All in all, this inexpensive electronic USB load is functional and does exactly what it's supposed to do. The decision not to include a printed copy of the operator's manual is a big thumbs-up to help reduce unnecessary waste. The standard of documentation is quite poor and just about covers the basics of operation. This is only to be expected for such a low-cost piece of test kit that only gets occasional use. When it comes to displaying the word 'Auto' or 'OTP' I think even I would be able to work out a better representation of the letter 't' on a 7-segment display. There is always room for improvement… I would like to see better signage for the trigger pushbutton and a better red filter over the 7-segment display to improve readability. All in all, this USB load does what it says on the tin. It is inexpensive and, thanks to the fan, can dissipate high power levels for extended periods. The maximum power handling is 35 watts and it did not get over-warm when I ran it at 30 watts so it should be able to take whatever the current USB charging specification can throw at it. If you think you can make use of this electronic load to test a USB power supply this is a good, convenient low-cost solution. You could hardly build one more cheaply yourself from scratch. ⏮

200049-02

4 SALE @ WWW.ELEKTOR.COM
→ Joy-It JT-HD35 USB Load Tester
www.elektor.com/joy-it-jt-hd35-usb-load-resistor-35-w

Web Links

[1] Review: The Joy-It UM25C USB Tester: www.elektormagazine.com/news/review-the-um25c-usb-tester-with-color-lcd-and-bluetooth

[2] JT-HD35 operating manual: https://www.joy-it.net/files/files/Produkte/JT-HD35/JT-HD35-Manual.pdf

CAN Bus + Arduino for Solar PV Cell Monitoring

Detect and locate serviceable panels in large arrays

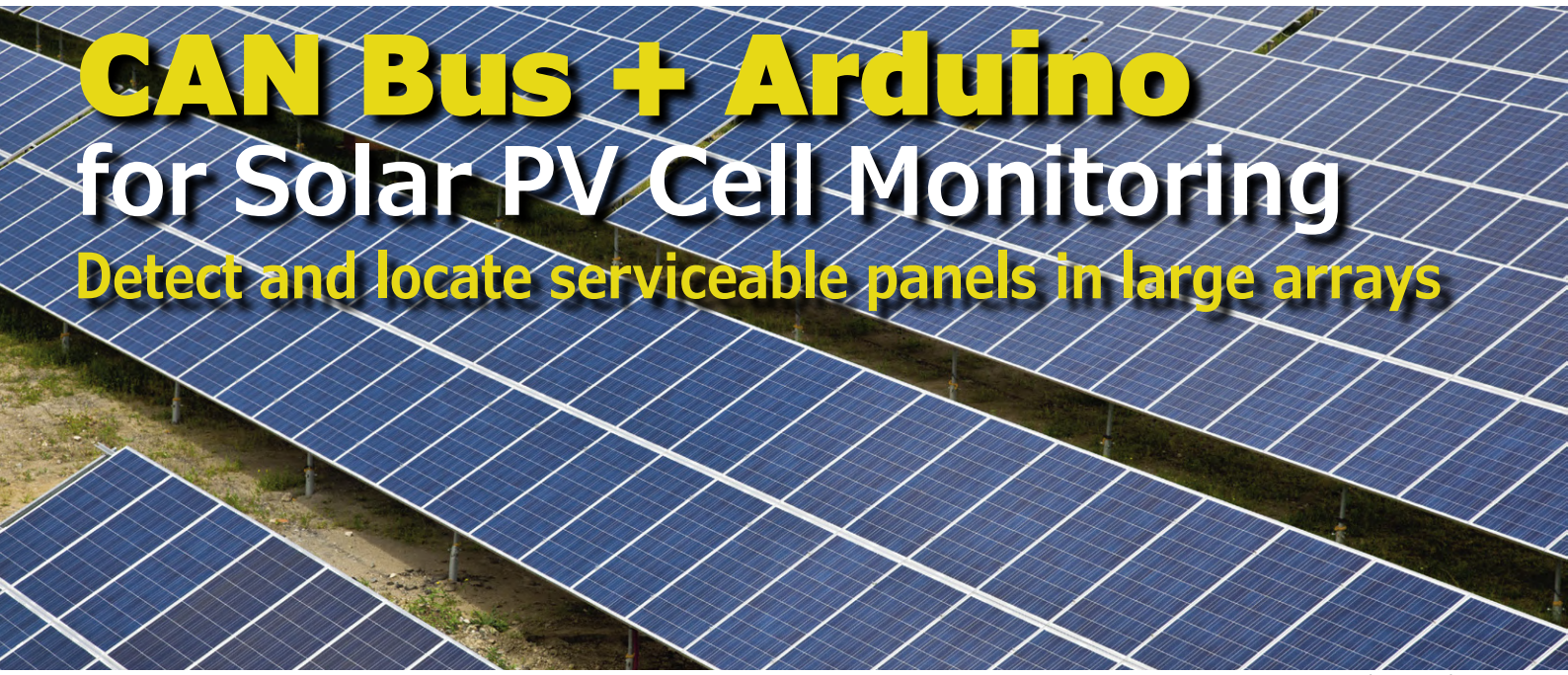


image: Shutterstock

By **Somnath Bera** (India)

Why limit the use of the exquisite CAN bus to in-vehicle sensorics? That's what the author thought when customers, installers and authorities bickered about disappointing results from a 25 kW solar cell array on commercial exploitation. Here's how an electronics system got designed to read out the power yield of each PV panel individually along with its number so the serviceman can pinpoint it and clear the source of the problem (literally — read on).

The situation: a roof-mounted solar PV cell array comprising 55 panels, each generating 24 volts and up to 25 kilowatts in total supplied to a large single-phase, 220-V out, inverter, charging a large battery array well out of sight. The consumer: a conglomerate of shops requiring electricity for evening sales and other activities. The shops were supposed to get six hours of lighting from 5:30 pm to 11:30 pm. The setup was inaugurated with a great fanfare in the presence of a big shot who came down from headquarters.

Wash & wipe

The solar PV array, inverter, and the battery system worked fine for some time but slowly, over a couple of months, the performance deteriorated such that even with charging periods of 10 to 12 hours, the 'evening' power did not last for more than 3 to 4 hours. The brick batting started. The local Township Administration Department (TAD) started blaming the installers for poor quality installation, cheap PV products etc. But a quick glance at the rooftop mounted PV panels revealed the mystery in no time. The 55 panels worth 500 watts each are spread over three rooftops, and huge numbers of pigeons and crows were flying right above them. Most of the PV panels were dirty and covered in at least two major servings of bird dropping. The shopping complex is situated adjacent to the local marketplace which is thick with birds like crows and pigeons. The droppings are so frequent that almost all PV panels get the complete share in about a fortnight's time.

The remedy I suggested was to wash each panel with water every alternate day, and as a long-term measure, to move the panels away to a place with fewer birds in the air above. The regular cleaning restored the performance to the original level and after a while it turned out that only selective cleaning was sufficient to maintain the performance, rather than the alternate-day scheme. Still the washer person had to visually check all the panels first and then proceed to the selective cleaning. The checking takes substantial time as the worker has to go round all the panels to see which ones need cleaning. For the 55 panels involved this was rather easy but for TAD's next setup, a 100-kW PV panel array, a nightmare loomed for the cleaning person or staff. In short, TAD wanted to know instantly which panel in the array needs cleaning so that the "bucket & mop" person can go straight to that panel and do the surface cleaning in no time.

Enter the CAN bus

Control Area Network (CAN) using the cheap Arduino is a great solution for measuring the performance of the individual PV panels. The CAN bus was invented by Bosch within the automo-

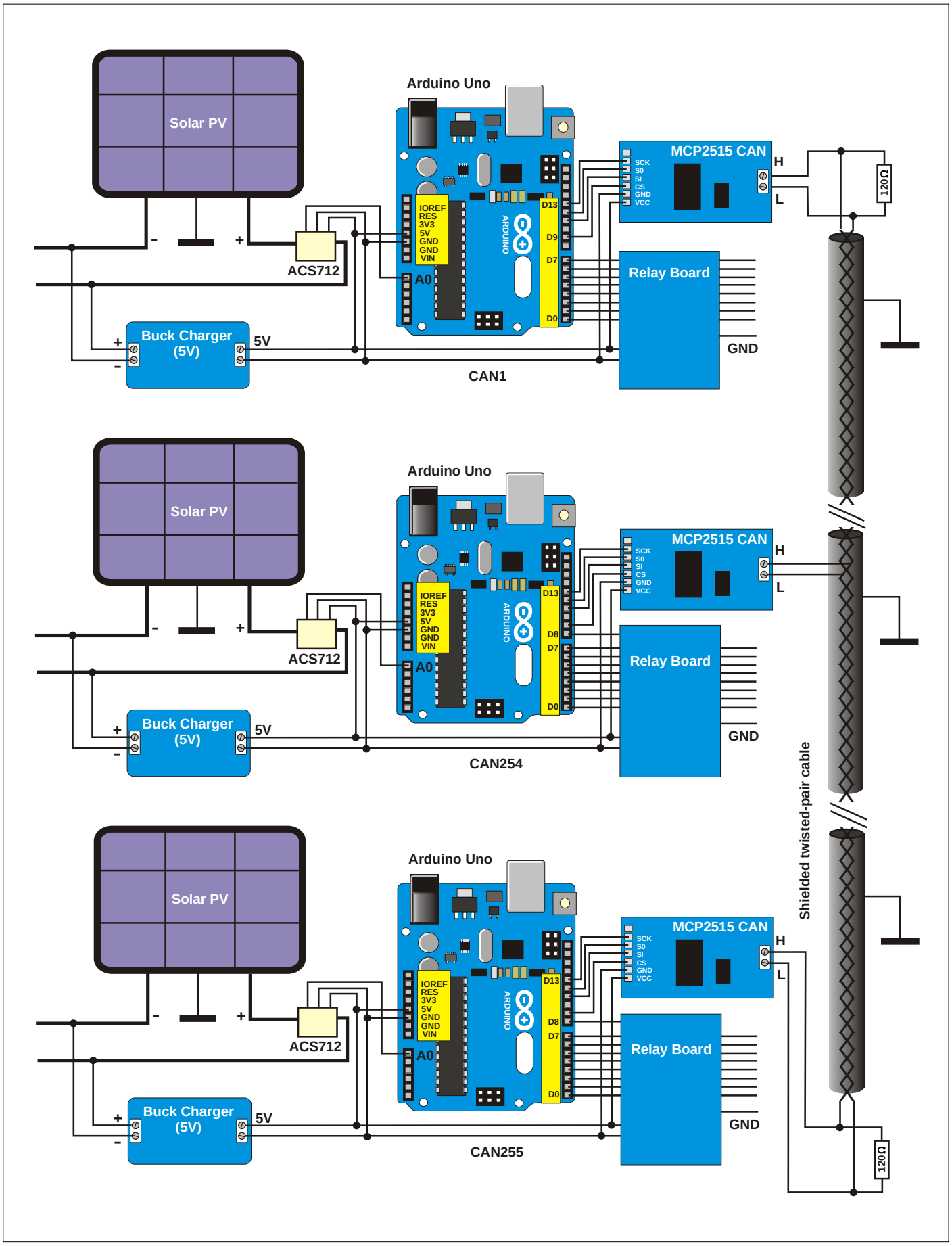


Figure 1: Block diagram showing the modules, devices and cables involved in making a CAN bus based monitoring system for a solar panel array with up to 120 (!) PV panels (just 3 shown here). Each panel is individually monitored for failing output current due to whatever cause. One Arduino UNO is assigned the CAN Bus Master level, the others are slaves.

tive industry and permits the collecting of signals from hundreds of sensors fitted and connected in a vehicle. Parameters sensed for a modern car include ambient temperature, tyre pressure, brake fluid level, engine temperature, speed, AC temperature and many more. All sensors are connected on just two wires in a bus structure and connected to the central computer which monitors the sensors' output signals and if necessary responds by issuing control commands to relevant sensors, again over the two-wire bus.

The CAN bus runs from node to node using a shielded twisted-pair cable having an impedance of 120 ohms. For proper matching and to reduce signal reflection, both end nodes on the CAN bus have a 120 Ω termination resistor.

For a cheap CAN bus transceiver we use the MCP2515 chip, which normally comes from China at under $1.50. At the microcontroller side it's SPI ready and at the CAN bus side it has a barrier strip for connecting the CAN H (high) and CAN L (low) bus wires. The bus is implemented with good quality, durable, shielded twisted-pair copper cable. For the microprocessor we propose the ATMEGA328 sitting on an Arduino UNO, which receives the sensor signals on its digital and analogue I/O pins. In order to control actuators you can connect relays, servos and other output devices on the remaining I/O pins.

For simplicity we will use only CAN 2.0 which can carry 0 to 8 byte data at a time from one node to another. All the nodes including the master get an identifier (0 to 255) through which the communications get sent & received. The lower the identifier the higher up in the order of devices sending & receiving. Usually the bus master gets the lowest identifier. The CAN bus (protocol) has excellent error checking capability.

The number of nodes can vary from 0 to 130 on a single CAN bus and the length also varies from a few meters to few kilometres. Reduced data speed means larger distances between the end nodes. However, for anything beyond a few hundred meters a CAN bus repeater should be used to ensure noise immunity and signal integrity The maximum number of nodes permissible on a typical CAN 2.0 bus around 120, which is still way more than the 32 offered by an alternative like the RS485 MODBUS.

BUS speed	Bus Length	Cable Stub Length
1000 kbit/sec	40 m	0.3 m
500 kbit/sec	110 m	0.3 m
250 Kbit/sec	240 m	0.3 m
125 Kbit/sec	500 m	0.3 m
100 Kbit/sec	500 m	0.3 m
50 Kbit/sec	1300 m	0.3 m
20 Kbit/sec	3300 m	0.3 m
10 Kbit/sec	6600 m	0.3 m
5 Kbit/sec	13 km	0.3 m

A solar PV panel health monitor

Using the CAN bus we can add up to 120 solar PV panels and read their individual performances, i.e. the voltage and the current contributed by each one to the total of DC fed to the inverter. A suggested block schematic for a PV cell performance surveillance system is given in **Figure 1**.

Being connected in parallel, all PV cells in the array produce the same voltage but different output currents. Therefore, a simple current sensor is connected at the load terminal of each PV panel. The ACS712 is a Hall-effect "probe" for measuring current. The -30 version can measure current up to 30 amps. The barrier strip is for connecting the device in series with the PV power circuit. Assuming Chinese sourcing, the ACS712-30 costs around $2.00 apiece. For DC readings from an ACS712-30 current sensor with its 'out' pin connected to the Arduino UNO A0 pin, you simply program

```
load_current = 0.075714*(analogRead(A0) - 512)
```

The UNO's 5 volt supply comes from a small Buck charger connected to the 24 volt PV cell bus. The buck charger should cost around $2.00 apiece. Its main advantage is adding extremely little power dissipation to the monitoring system.

Besides accumulation of dirt, grime and pollen, other factors that degrade PV cell performance include humidity resistance (small bubble formation on the surface), snail trails (discolouration of the surface), micro cracks and hot spots. Any and all of these may cause the PV cell output to go down by as much as 35% which is measurable in terms of the output current. Since we are not overly interested in measuring the *absolute* current contribution of the panel, we can make do with the value returned by

```
load_current = analogRead(A0) - 512
```

This 'value' gets compared with a set reference as well as with values from other panels with a view to deciding to raise an alarm or not. Having seen the alarm and the location (number), the cleaning person will go to the offending panel and check for dust, dirt and pollen before washing it once. If that fails to solve the problem, the next level is a check for hot spots, excessive humidity resistance etc. In the worst case the panel needs to be replaced.

Can Bus 4*4

The project allows for a 4*4 keyboard to be incorporated on the CAN bus, for control purposes and local entry of numbers. **Figure 2** shows the schematic and **Figure 3**, the experimental setup on the author's workbench. You will see the keyboard being used as an option in the project software.

The project software

The software deveped for the project takes the form of a bunch of Arduino sketches and is hosted on the Elektor Labs website [1] along with some updates and notes from the author. The software should be taken as a starting point, not as a finished, slick application, and is intended to spur your ingenuity and creativity once "the numbers are in reach". It is broadly classified in two parts: master or slave. A slave picks up the DC reading from the A0 port and relays it to the master. The master has two modes of operation: Master in display mode, or Master in control mode. The CAN master is given the ID '1' while the other units are assigned '254' and '255' like in Figure 1, or any other number between 2 and 253. The maximum number of nodes is 120.

Master in display mode: This is achieved by using digital I/O pin D10. When D10 is high the master goes into display mode, collecting data from all the slaves and displaying them in a grid structure. It then finds out if any signal (i.e. power

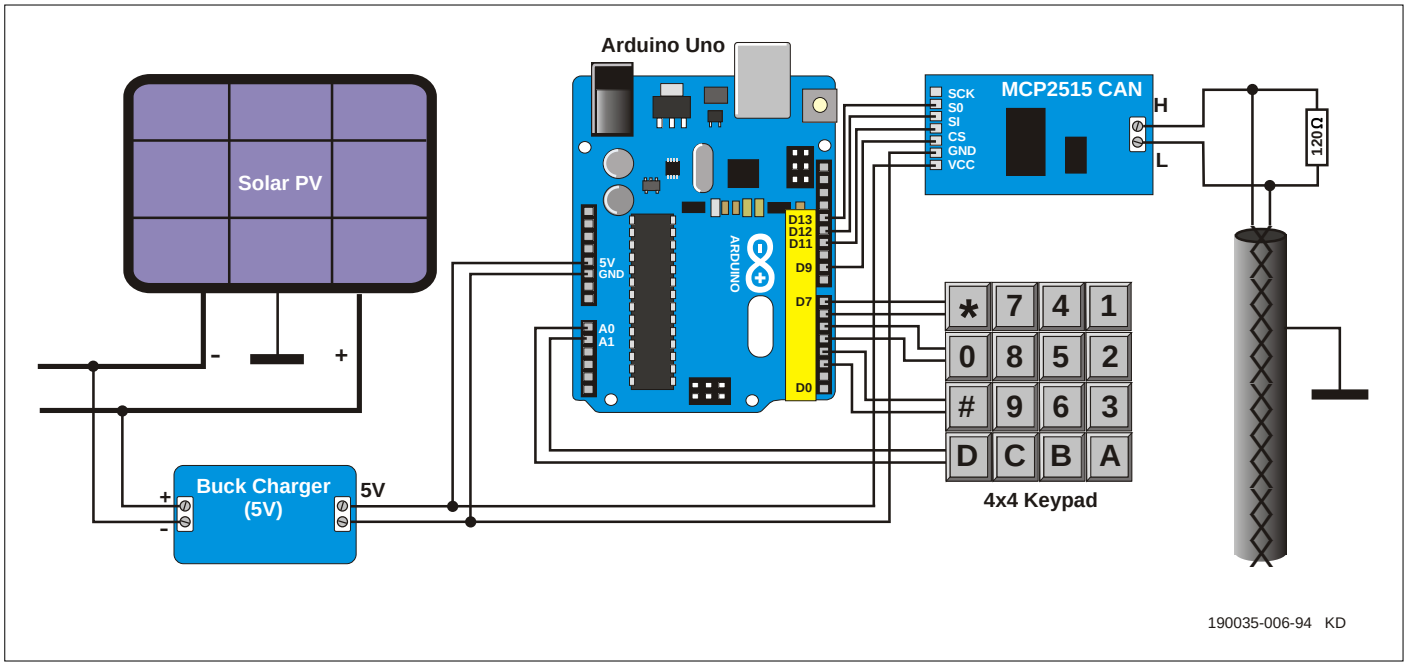


Figure 2: Schematic of the CAN Bus Master with a 4x4 keypad for use in control mode.

level) is under the desired level and goes on to issue an alarm for the cleaning of that particular panel. This mode is optimum for PV array monitoring.

Master in control mode: When D10 is open or low, the master goes into control mode. In that case the nodes are connected to relays driven through digital pins D0 to 8. There is a 4*4 CAN keypad connected on the master CAN device (ID = 1). The master issues a command string like

```
25581#,25580#,13011#
```

etc. and the respective node relays are switched on or off to control, say, a shower and/or a wiper connected to the PV cells. Examples:

`25580#` : 255 is the node number; 8 is the digital pin number; 0 is off; 1 is on.
`25471#` : 254th node, relay connected to the digital pin 7 and a 1 to energize it.
`25130#` : 251st node, Digital-3 relay, off.

The hash character (#) is used to terminate the command. The CAN bus master can have its own PV current sensor, connected to its A0 pin or it may be devoid of any PV panel connected to it.

Your application?

One alternative use of this type of configuration is to find out in a small city or park how many light poles are actually on or off.

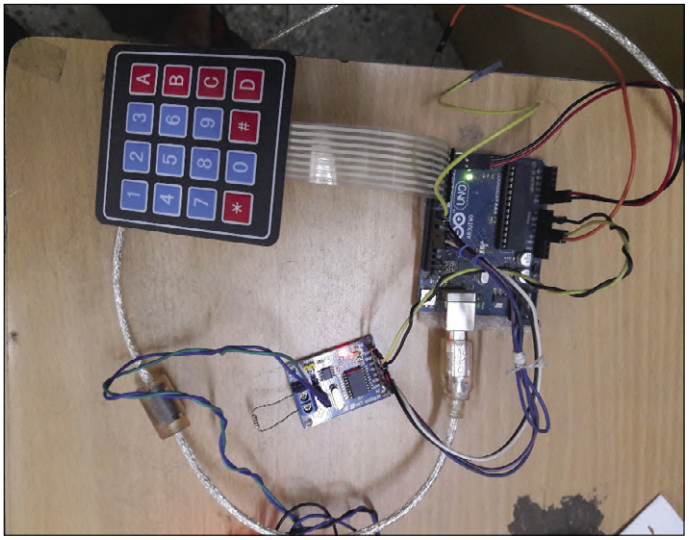


Figure 3: Experimental version of the CAN Bus master controller extended with a 4x4 keypad.

A system like the one described here also enables many small units connected on the CAN bus to be controlled from a central desk. Other uses include hotel room alarm buttons for guests, and hotel lights. Instead of running all kinds of wiring inside the building, a single twisted-pair CAN bus may be enough for extensive, reliable control and measurement.

190035-01

Web Link

Project on Elektor Labs: www.elektormagazine.com/labs/can-bus-for-solar-pv-cell-monitoring

Analogue Electronics Design (3)

Case Study #1 — Section 3: Preamplifier response optimization and trade-offs



By **Ton Giesberts** (Elektor Labs)

In this article series, experts in the field explore aspects of analogue electronics design that should benefit an increasingly 'digital-only' audience, and underpin that analogue is not black magic. In this third and closing instalment of the first Case Study in the series we revert to the microphone preamplifier used in the Bat Detector^{PLUS} project [1], concentrating on those features that set it apart from a routine audio preamp implemented with opamps.

If you are new to this series, reference should be made to the previous installments [1],[2]. For all readers to benefit from, the circuit of the microphone preamplifier we're about to discuss was extracted from the Bat Detector^{PLUS} full schematic and is shown separately in **Figure 1**. Flashlights on and let's do some more design analysis on it.

Dynamic range

Because of the very high fixed gain and the low supply voltage of the preamplifier, there is a chance that the output will saturate and clip, and when signals are too small, and/or frequency dependent, the gain may be insufficient. The dynamic range is determined by the maximum undistorted output signal and the smallest usable signal well above the noise. Unlike ADCs and DACs, *S/N* is not the same as dynamic range. The *S/N* of microphones is usually specified at 94 dB sound pressure (1 pascal), but for most microphones there is also a linear relationship between sound pressure and output voltage, usually up to 120 dB. Then there is still minimal distortion and therefore the dynamic range is considerably greater than the specified *S/N*. For the bat detector, it is obvious that adding a fast, automatic gain control adapted (perhaps even limited) to the source signal would be a good addition to further increase that range. This could be something quite simple, like the classic circuit with a JFET as variable resistor. This will also improve the detection of the modulation. There is hardly any danger of feedback (howl) if the high frequencies are filtered out properly at the power amplifier, preventing the loudspeaker from reproducing them, if that should be a problem at all. Unfortunately, despite the frequency separation, both harmonics in the output signal of the power amplifier and the reproduction by the loudspeaker can result in feedback from the loudspeaker to the microphone. This is a mechanical problem, because even the best loudspeaker is not ideal and will distort the sound. In the case of the bat detector, there are of course no requirements for really small signals as with an audio amplifier. A signal-to-noise ratio of 20 dB in the signal going to the comparator is likely to suffice.

Bias currents

Seeing a 10 kΩ resistor fitted to each non-inverting input may raise the odd eyebrow. This is normally done to compensate an extra offset caused by the bias currents of the opamp. But even at the maximum temperature range of a TL06x, those currents through the JFET inputs are smaller than 5 nA. That 5 nA across a 100 kΩ resistor drops a minuscule 0.5 mV, i.e. almost negligible, also because there is no DC coupling between the various amplifier stages. Unless bipolar opamps are applied, the bias currents can amount to a few 100 nA or even µA. Only then, this needs to be considered, depending on the resistance values of course. Also, the 10 kΩ resistors render the amplifier stages more sensitive to disturbances and unwanted parasitic feedback. Consider replacing them with wire bridges.

Opamp as comparator

The circuit also uses half a TL062 as a comparator. This is often tempting to do when there is still an opamp left, or to use a dual version instead of a single opamp. Using a real comparator implies an extra IC. Comparators generally have an open-collector output and are not always advisable. The first thing to pay attention to is whether the maximum differential input voltage range is enough, and fortunately that is the case here. The same goes for opamps sporting CMOS and JFET inputs. With

bipolar opamps like audio opamps, that range is often limited by an extra network to protect the input transistors, e.g. a pair of diodes in anti-parallel or Zener diodes in anti-series. Also, if the specifications are exceeded, the output voltage can flip over. A problem with the TL062 is its maximum input offset, which is up to 15 mV, even up to 20 mV at the full temperature range. An applied signal will have to compensate that offset before the output will toggle. In the case of the detector, the sensitivity is dependent on the exact IC used. It's a dependency we don't want.

Another problem is the differential voltage gain which is 3000 minimal according to the datasheet and 6000 times typically, which is actually too little to work properly as a comparator. To work accurately, at least 10, possibly even 30 times more gain is needed. Take a real comparator like a LM311 or LM339, then that gain is typically 200,000.

The effect of the supply voltage on the differential gain is not specified, but it would be interesting to know if it affects the rather low ±4.5 V from the 9 V battery. To make proper rectangular waves at frequencies as high as 120 kHz, more bandwidth and certainly a higher slew rate is needed. With a comparator like the LM311 the slew rate depends on the differential signal, and factually on overdriving. The larger this input signal, the faster the output voltage changes, but even at 2 mV, around 30 V/µs is available mainly on account of the output transistor, and not the pull-down or pull-up resistor. By contrast, the 3.5 V/µs slew rate offered by the TL062 in our design is somewhat low.

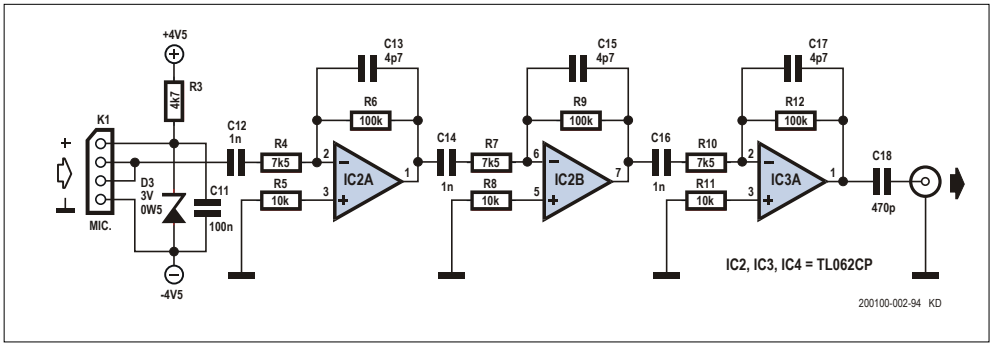


Figure 1: The 3-stage microphone preamplifier designed into in the Elektor Bat Detector^{PLUS} has some remarkable features.

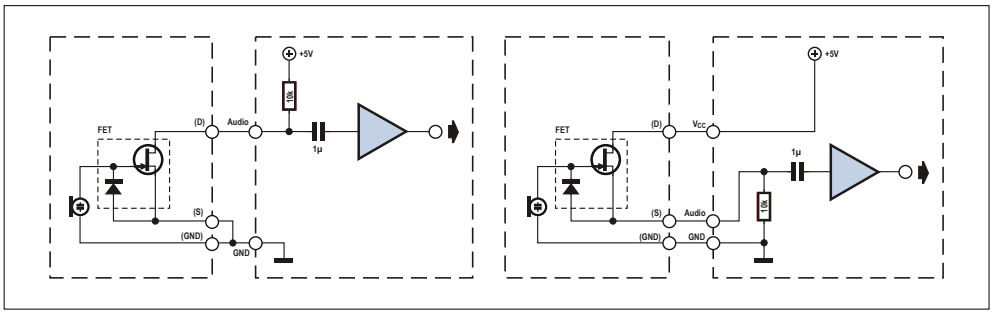


Figure 2: Schematic diagram for connecting a 2-pin electret microphone (left) and a 3-pin version (right). In a 2-pin electret, the FET source and ground are already connected internally.

2- or 3-pin electret?

For many, the electret microphone capsule will be familiar. Most types have just two connections. One is the ground connection (GND) and attached to the metal housing (screen). The other is the output signal taken from the drain (d) of an internal JFET. The source (s) of the JFET is connected to ground. A polarized diaphragm converts the sound into a voltage for the JFET's gate (g).

By inserting a pull-up resistor between the input of the microphone amplifier and the supply voltage (which is usually quite low between 1.5 and 5 V), the FET is biased, and some amplification of the order of 10 dB already occurs inside the electret microphone. The disadvantage is the non-linear steepness of a FET. The steepness or transconductance g_m is the relationship between variation of the drain current caused by variation of the gate voltage. Because this relationship is nonlinear, the output signal will deviate from the diaphragm voltage, causing a slight distortion.

With small signals, this effect is not that noticeable, but as the sound pressure increases, the distortion rises and even becomes visible on an oscilloscope. Also, the phase of the output signal can be inverted, but that is usually not a problem and can be solved by reversing the polarization of the diaphragm.

With 3-pin electret microphones, the source of the FET is also available on the third connection, see **Figure 2**. We are not talking about types with a built-in amplifier that need a real supply voltage rather than biasing only. Using that third connection, the FET can be configured as a source follower, reducing the distortion. Unfortunately, the output signal is lower in that configuration. The drain connection is connected directly to the positive power supply and the source is pulled down to ground. The value of the pull-up resistor in the 2-pin, and the pull-down resistor in the 3-pin configuration, depends of course on the type of electret microphone and the supply voltage. Always check the datasheet. MEMS microphones always have an IC built in, so a real supply voltage is needed. There are tricks to establish a 2-wire connection to a MEMS microphone by placing a load at the output to vary the supply current, but what happens to the specified properties is anyone's guess.

Calculation of the input noise

For the noise we have to calculate using a bandwidth of 120 kHz, rather than 20 kHz, the default range for audio. Worse, it is

often only mentioned A-weighted. The well-known formula that represents the noise as a function of temperature, resistance and bandwidth says:

$$U_{noise} = \sqrt{(4\ k\ T\ B\ R)} \qquad \text{(Eq. 1)}$$

with
k = Boltzmann constant (1.38×10^{-23} J K^{-1})
T = absolute temperature (in K, 270 K is assumed)
B = applicable bandwidth (in Hz)
R = resistance (in Ω)

The total noise at the input is comprised of the noise already present in the source signal (U_{noise1}) and the noise generated by the resistors (U_{noise2}) and the opamp used (U_{noise3}).

$$U_{noise(total)} = \sqrt{((U_{noise1})^2 + (U_{noise2})^2 + (U_{noise3})^2)} \qquad \text{(Eq. 2)}$$

According to the datasheet, the noise of the microphone we used was measured with an A-weighted filter. That filter sadly is not specified for frequencies above 20 kHz, just the ultrasonic area we need. It is even recommended to use a steep 20 kHz filter in the sound pressure meters in combination with the A-weighted filter. This makes the statement in the datasheet unusable. The only thing that is certain is that it will be considerably larger for our calculation. So we make an assumption. If we compare the A-weighted filter with a first-order 10 kHz low-pass filter, then the noise at a bandwidth of 120 kHz is up by at least $\sqrt{(120 / 10)} = 3.5$ times, so roughly estimated (or even guesstimated) 11 dB higher for our calculation than stated in the datasheet. Hence, a signal-to-noise ratio of 59 dB(A) at a sound pressure level of 94 dB (1 pascal, Pa) and a bandwidth of 120 kHz is then estimated to be 48 dB if the measurement is unfiltered. At that sound pressure, an output level of –42 dBV is given for 1 kHz, i.e. 8 mV. At 1845 times gain it yields 14.76 V and means an excessively overdriven amplifier. But ultrasonic, above 40 kHz, the output level of our MEMS microphone decreases and between 10 and 20 kHz there is even an increase of more than 10 dB, so the degree of overdriving is highly frequency dependent and almost unpredictable. A calculation based on a linear noise behaviour is then not exactly what we need for the operation of the detector, but we make do with it.
Of course, it would be better to adjust the frequency response of the amplifier to the microphone. What should the calculation for the noise look like? If it should be possible to exchange microphones, then only a digital solution is considered. But back to calculus, now for the noise per source to the output. The noise of the microphone at a sound pressure is 94 dB then:

$$8 / (10^{(48 / 20)}) \times (100 / 7.5) = 0.425 \text{ mV}$$

For the resistor noise contributions we need to calculate with the parallel circuit of the two that determine the gain, i.e. the 100 kΩ and the 7.5 kΩ ones. We leave out the resistor in the non-inverting input path, assuming it is replaced by a wire bridge:

$$\sqrt{(4 \times 1.38 \times 10^{-23} \times 270 \times 120000 \times 6977)} \times (100 / 7.5) = 0.047 \text{ mV}$$

The opamp noise is amplified as with a non-inverting amplifier. The noise of a TL06x is typically 42 nV/√Hz, so:

$$(42 \text{ nV} \times \sqrt{120000}) \times (1 + 100 / 7.5) = 0.21 \text{ mV}$$

The total noise level at the output of the first amplifier stage in the Bat Detector works out at:

$$\sqrt{(0.425^2 + 0.047^2 + 0.21^2)} = 0.476 \text{ mV}$$

giving a signal-to-noise ratio (S/N) of

$$(100/7.5) \times 8 \text{ mV} / 0.476 \text{ mV} = 224.1$$

or a solid 47 dB when expressed in decibels.
In other words, the noise at the microphone amplifier output is almost exclusively determined by the MEMS microphone. The contribution of the following stages is negligible with enough amplification. If the gain per section drops, the next amplifier stages will also have an impact. This is another reason why eight sections with little gain is ill advised. And again, this calculation is only valid assuming the noise is linearly distributed and not approximately like the frequency characteristic.

MEMS microphone straight on the PCB

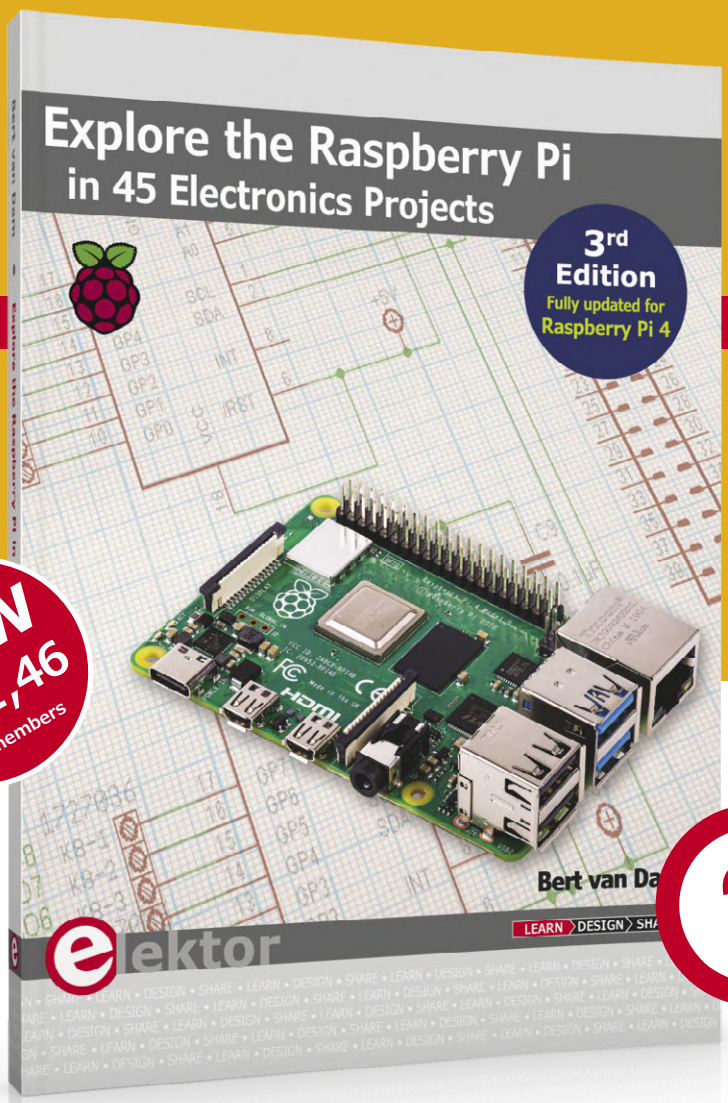
Whether it makes sense or not we'll leave aside for a moment, but what would the practical application of a MEMS microphone look like if it were mounted directly on the PCB? The following proposals and queries can be taken as a starting point.

1. Mounting a MEMS microphone on the PCB should be done in such a way that the acoustic port has a good duct to the ambient air.
2. Is a rubber seal or anything else necessary? Is it possible to make a sealed connection to the housing of the detector so that moisture and dust etc. cannot penetrate the MEMS microphone?
3. Is it also possible to over reduce the frequency response and sensitivity? The use of a bottom-port MEMS microphone will be better for the shortest possible acoustic path to the outside. It is then located at the bottom of the PCB raising the use of SMD components to: recommended.
4. If no option is used for other microphones, the selected frequency characteristic of the MEMS microphone is subject to correcting to obtain a straight frequency characteristic so that no other microphones are needed. Applying a digital version will be a more sensible option than implementing the whole system in analogue fashion. By modifying the firmware "in a flash" it is easy to make improvements, obtain different filter characteristics (e.g. a specific frequency band), or compensate for tolerance.

But even for the current version of the amplifier in the Bat Detector^{PLUS}, the question remains: how would you make your own test setup to determine the actual frequency characteristic with any accuracy?
With this challenge we conclude Case Study 1 in the series Analogue Electronics Design. In the next Elektor edition we will be filleting another subject 'in a completely analogue way'. ⏮

(200100-01)

Explore the Raspberry Pi in 45 Electronics Projects



NOW € 31,46 for members

Description:

Getting started with Raspberry Pi has never been easier. Read the Third Edition of Bert van Dam's Explore the Raspberry Pi in 45 Electronics Projects for a thorough introduction to the Raspberry Pi, handy programming tips (e.g., Bash, Python 3, and Javascript), and 45 compelling design projects.

Fully updated for the Raspberry Pi 4, the book details a wide variety of design projects:

- Drive an electromotor
- Temperature control
- Motor speed controller
- Web server with CGI
- And more!

Take your Raspberry Pi-based projects to the next level.

elektor

Read more on:

www.elektor.com/explore-the-raspberry-pi-in-45-electronics-projects-3rd-edition

Inexpensive E-Scooter

How does Lidl's "road approved" 300 euro E-scooter perform?

In many cities you can see small electric scooters all over the place. You can rent as well as buy them, although at fairly high prices. Can much cheaper scooters also be used? An illustrated answer is given in this article, using the example of a dirt cheap e-scooter spotted among the 'weekly deals' promoted by a large supermarket chain in Germany.



Figure 1: The loot — after hunting the e-scooter I still had time for some food supplies.

By **Dr Thomas Scherer** (Germany)

Last year I was on the road for the first time with a rental car in downtown Tel-Aviv. As such I am not squeamish and adapt to the regional driving habits quite quickly. Different countries, different driving style. But I hadn't experienced this yet: in this bustling city in the Middle East there are not only narrow streets and hardly any parking spaces, but in the densest traffic you are overtaken and harassed by various electric scooters flashing past on the left and right at the same time. Also at the boardwalk, there are always the carelessly left rental e-scooters.

Ergo: I also wanted to have one...

Panning to Europe

All European countries have different rules regarding these small scooters. While in many European cities the whirring of the little speedsters has long been part of everyday life (and pedestrians have become accustomed to this new "danger"), this is new territory in Germany, as these vehicles were not registered until 2019. Of course, not without corresponding bureaucratic regulations (see box on **small electric vehicles**).

And, as befits the country of the "soon" to open capital airport BER, these special regulations make the scooters more expensive, because first of all the Chinese production has to adjust to the German extra sausage. This is now the case, and so the country's largest food discounters, Aldi and Lidl, offered corresponding models at a low price shortly before Christmas 2019. Lidl kicked things off and had the model "Doc Green

Small Electric Vehicles
(German regulations)

Since 2019, small electric scooters have been legal and allowed for real use in Germany too. But naturally not without conditions:

- 20 km/h maximum
- not on sidewalks
- 14 minimum driver age
- no helmet requirement
- detachable 'lighting devices'.
- compulsory insurance
- General Operating License
- two independent brakes
- no indicators required
- a "bright bell" is needed



Figure 2: The contents of the box — the folded e-scooter is waiting for the assembly of the handlebars.

E-Scooter ESA 5000" in their program at the end of November for an incredible €299. Aldi followed shortly after with a model priced at €279.

Thanks to the Internet, interested parties were already informed in advance. That's why I knew that the Aldi model with one charge would only get half as far as its Lidl competitor, which is supposed to cover 22 kms after all. I also knew that with the model M365, there is a practically identical but more expensive and non-DE approved device from Xiaomi [1], and this Chinese company stands for high performance and quality at low prices, at least for smartphones. Even though Lidl denied this connection, it was worth the extra cost to me, and therefore I was already standing in front of the local Lidl store in the dark at ten to seven on the day of sale. And I was not the first! In front of me there were already two gentlemen of similar age, each armed with a large shopping trolley and ready to hunt through the corridors and capture the coveted piece. By 7:00 am another eight people were behind me. All of them waited in the queue almost in a British polite calm.

And the door opened and the race began. In fact, the hunters in starting positions 1 and 2 had also loaded the first two scooters. And I got the third - of six. When I was back at the cash desk, a look back showed that everybody was gone. Whew — lucky! In **Figure 1** you can admire my hunting trophy.



Figure 3: The "accessories" include the charger, a few screws including an Allen wrench, and papers.



Figure 4: Fully assembled e-scooter. That was easy.



Figure 5: This is how the handlebars and "dashboard" look when switched on. The battery is flat.



Figure 6: Stable welding seams and folding mechanism.

First impressions

Once at home I transported the package to my electronics lab in the basement and of course I couldn't resist: it had to be unpacked. After it was lifted out and freed from the foam pieces, the picture in **Figure 2** unfolded. The steering column could be folded away and the handlebars still had to be fixed with four Allen screws. Also a charger and paperwork — especially the "data confirmation" — was there, as **Figure 3** shows. The latter is important because it corresponds to a "General Operating License" and the vehicle can only be insured on its presentation. Completely assembled, the scooter was standing in front of me in all its glory just five minutes later (**Figure 4**).

But before starting the scooter, I had to find out where to switch it on and what the LEDs etc. mean. According to the manual you press the black button on the handlebars to switch the scooter on. Then the very bright, two-digit display lights up with the correct speed of 00 km/h and the LEDs indicate the charge level of the battery. **Figure 5** shows the handlebars in riding position. At this point my zest for action was abruptly slowed down because there wasn't much power in the battery, and the manual demands a first full charge. The red paddle on the left is the recuperation brake and the blue one on the right is a kind of thumb operated throttle. The handlebar grips are not rotatable. In the upper right you can see the brake lever for the rear disc brake.

Five hours later the battery was full. The first test drive could have started now. Tried it out in the cellar: yes, it drives. Even then I noticed: there's not that much power, even if it was still in eco mode to be on the safe side. But before setting off, I wanted to know how well this electric scooter is designed.

A look at the frame shows that the

Features

- Road approval
- Range: approx. 22 km
- Speed: max. 20 km/h
- Motor: rated power 350 W
- Li-Ion battery rated at 36 V / 7.2 Ah (259 Wh)
- Charging time: up to 7 h
- Charger: 42 V / 1.5 A
- Puncture-proof 8.5'' air chamber tyres (Honeycomb)
- Non-slip rubber pad on step board
- 2 independent brakes
- Front light: 20 lux, white reflectors
- Rear light: 10 lux (brake light), red reflectors
- Lateral: reflectors in yellow
- Bell (preassembled on handlebars)
- Side stand
- LED display for speed and charge status



Figure 7: Front wheel with hub motor and solid rubber tyres.



Figure 8: Rear wheel with disc brake and locking device for the folded down handlebars.

aluminium tubes are neatly welded (**Figure 6**). The mechanism for locking the handlebars also looks very stable. **Figure 7** shows the front wheel. Lidl talks about "8.5" air chamber tyres (Honeycomb)" but I am not convinced. It reminds me more of normal solid tyres. In any case you don't have to fear any loss of pressure. **Figure 8** shows the rear wheel with the oversized, perforated brake disc, rear lighting and license plate holder. The grey rubber nipple at the top is used to lock the folded-down handlebars. So you can carry the scooter comfortably despite its weight of around 15 kg. You can find more details in the **Features** box.

Inner life

As always, the innards are particularly interesting for electronics engineers. So the scooter had to be screwed open right away. That's easy, because there is only a thicker plastic cover at the bottom (**Figure 9**). Thirteen screws later you have it in your hand. **Figure 10** shows that it is simply sealed with a foam tape. No big problem, because eBay offers such aluminium covers for little money in different colours for the Xiaomi M365 (search for "Bottom Battery Cover Xiaomi M365") and they should fit. In general, there are a lot of matching accessories under the keyword "M365".

Figure 11 shows the most important thing: the battery. It is built from round cells (type 10s3p) and looks quite decent. The brochure information is also correct. The battery is screwed to the frame from below and is easy to replace if faulty. The second most important thing is the electronics: **Figure 12** shows that it is extra protected from moisture by a transparent plastic case. The six MOSFETs for a three-phase bridge circuit are clearly visible. From this it is clear that it is an electronically controlled BLDC motor, which makes it efficient and durable. **Figure 13** shows the charging socket, which is protected with a rubber plug. Again the great welding seams impress.

Test drive

After everything was reassembled, I went out into the street. In the YouTube video [2] you can admire my first, still cautious driving attempts. At first I only dared to drive in eco mode, which goes up to 15 km/h. And rightly so!

To make the scooter accelerate, you must first push it to a speed of over 2 km/h. This is no problem, one push is enough. This little inconvenience has the advantage that the high power consumption when moving off is reduced and the batteries last longer. Here we go and the 15 km/h did not feel unsafe despite the small wheels. But then came the reversing, and as we all know, you have to brake for that. First of all: I didn't break anything...

Of course I tested the left red lever with the recuperation brake first. Unfortunately it has a rather digital behaviour and the scooter brakes abruptly and strongly. Since I was not prepared for this, I almost flew across the handlebars. So I accelerated again and then decelerated with the rear brake. The too you have hardly any control in the lever. The disc brake is much too violent for my taste. If there would be foliage, the rear wheel would be gone. So far, so hard.

There's hardly any feeling of acceleration in eco mode. I would have imagined 350 watts differently. But maybe the full power only comes in normal mode, I thought. So I stopped, pressed the



Figure 9: The underbody is made of plastic and can be dismantled.

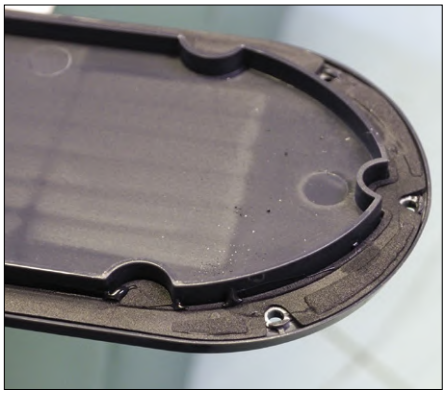


Figure 10: Unscrewed cover with foam seal.

black button twice to switch over and off I went. It may be that there is some measurable difference between the modes except the final speed, but I could hardly feel it. After all, the Lidl scooter runs its 20 km/h as promised. That should be enough.

Conclusion

In general I was positively surprised by the quality of the e-scooter. It is really neat and stable. The battery doesn't disappoint either and you can fully trust it to have the 22 km range - I haven't tried it though as I don't like pushing. The electronics also look good and the lighting is bright and sufficient. That a scooter like this can be made for this price amazes me.

I didn't like the acceleration so much though. That should be 350 W, really? There is no real electric feeling - except when recuperating. So to the biggest shortcoming: the brakes are just too violent. While the disc brake certainly gets a bit smoother over time, recuperation is simply too aggressive. Whoever designed this has only used a part of his mental potential.

All in all, I would come to a positive overall assessment despite the brakes taking some getting used to. The price/performance ratio is already very good. Nevertheless, I would strongly advise you to take a test drive if you want to buy such an e-scooter for yourself or your offspring.

A good friend of mine tried out if this scooter fits into the frunk (coinage from front and trunk) of his brand new Tesla Model X. **Figure 14** proves that it does. Martin would like to drive around town on the scooter in case his Tesla takes a break at a charging station. A good idea and the scooter is certainly cheaper than a wing mirror of his new electric vehicle. But the Tesla is 12.4 times faster and offers 885 times more power. ⏮

191251-03



Figure 11: The battery. Looks pretty decent.



Figure 12: The six MOSFETs are clearly visible in the encapsulated electronics.



Figure 13: Charging socket on the left side of the scooter.



Figure 14: Scooter in the frunk of a Tesla Model X (Photo: Martin Jepkens [3]).

Web Links

[1] Xiaomi Scooter: www.mi.com/global/mi-electric-scooter

[2] Test drive (YouTube-Video): www.youtube.com/watch?v=9UNWDpAYAoU

[3] Martin Jepkens: www.me-marl.de

Steeped in Electronics

Lead-free soldering and EU regulatory zeal

By **Ilse Joostens** (Belgium)



When I was very young, in the early 1970s, we lived in a row house with lead water pipes. When I was five years old my family moved into a brand-new house without lead pipes, but with flooring that contained asbestos and had been treated with pentachlorophenol. I must say that the latter sounds really ghastly, and with all the bad news now about things we used to like so much, it seems like a miracle that I am still alive and able to write this column.
The world has clearly become a lot more dangerous since I was young, and with the introduction of the RoHS regulations in 2006 the European Union in particular apparently found it necessary to launch a major crusade against the use of a number of hazardous substances in consumer electronics. For some mysterious reason, lead is regarded as the main culprit, and now it is even supposed to be banned from ammunition and fishing weights.

Lead-free baking

If you do soldering, make repairs or build prototypes as a hobby, you need not read any further, and you should get down on your knees and thank the gods that you are allowed to use lead-based solder. If like me you put together commercial products, you are out of luck and condemned to working with lead-free solder. Strictly speaking, you could still use lead-based solder for your prototypes, but to avoid the risk of contamination I use only lead-free solder.
The period shortly after the first Restriction of Hazardous Substances (RoHS) regulation took effect in 2006 was fairly dramatic for the electronics industry. Some lead-free processes

were not yet ready for use, and there was only limited experience with the new methods. During that period I repeatedly woke up drenched with sweat after yet another nightmare about razor-sharp tin whiskers constantly growing between the leads of fine-pitch SMT components, ultimately resulting in smoke signals.



As far as automated soldering is concerned, most of the problems have been solved by now, and in practice the tin whiskers and other potential disasters have turned out to be minor issues, although I have the impression that modern consumer electronics fails faster and more often than in the past. Of course, this could also be due to the quality of the components concerned. With the growing number of low-cost imports from Asia, it has systematically declined.

Unfortunately, manual lead-free soldering is still a bit of a challenge. And if we can believe the countless photos circulating on the Internet, soldering on computer motherboards is very hard on the fingers, and the higher melting temperature of lead-free solder certainly doesn't help. To see this for yourself, just Google for pictures matching the search term "solder fail", and be sure to have your burn ointment close to hand.

With my old soldering stations, soldering relatively large through-hole components on PCBs with plated-through holes was not exactly easy with lead-free solder, especially on solder pads in or next to large copper planes. The irons were usually set to a temperature of 420 to 450 degrees C, and in some cases I needed two irons at the same time to get the solder to melt. It's hardly surprising that I went through solder tips pretty fast. Although working with lead-free solder is a good deal easier using modern soldering stations with interchangeable active tips, my feeling is that soldering on large copper surfaces is still not straightforward. Desoldering through-hole components and SMD rework are still tricky tasks, leading to more failed components and half-finished products. In the very rare cases where I work with lead-based solder, it's always a relief. If you intend to get into electronics or you occasionally solder something as part of your hobby, I can only recommend that you avoid lead-free solder.

Good soldering stations with active tips are expensive, and actually there are no specific health hazards from working with lead-based solder. Of course, you shouldn't eat it. Lead-free solders usually contain a somewhat more aggressive flux, and the flux vapours are not especially healthy. This doesn't mean it's okay to start smoking lead-based solder like a chimney – that should be obvious.

Recycling

It's also questionable whether banning lead from electronics is really that much better for the environment. Instead of the current cheap disposable electronics, it would have been better to aim for higher quality and more durable products. This brings us to the recycling of products, and there as well the government busybodies and bureaucrats know how to make your life miserable. In most European countries there are organisations for recycling discarded devices and reusable waste. In Belgium we have Recupel, Bebat and Fost Plus, among others. I am fully in favour of recycling, but association with the above-mentioned organisations is not only expensive but also involves a lot of paperwork. Fortunately, I'm not bothered much with this as long as I do not sell completely finished products or batteries and keep the use of plastic packaging within reasonable limits. From what I hear from colleagues and competitors in Germany, the situation in Belgium is not so bad. Stiftung EAR, the German equivalent of Recupel, takes its role very seriously and has already imposed hefty fines in some cases. In Germany, rules are rules.

Despite all the regulations and prohibitions, components such as mercury switches and light-dependent resistors with cadmium sulphide (shiver!) are still readily available from Asian sources, not to mention some potentially lethal products that do not comply with any standards but are still for sale in shops here. Is this a sign of a double standard amongst our European policy makers? ⏮

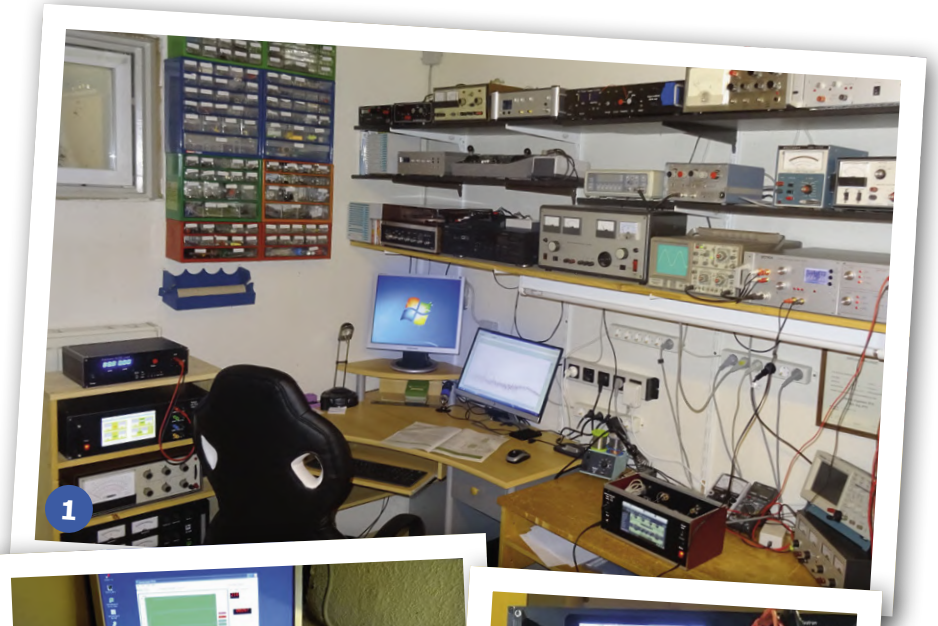
200082-02

Lab, Sweet Lab

A glimpse of The Holy Place — no unauthorized entry.

By **Eric Bogers** (Elektor Netherlands)

In July 2019 Elektor organised a small competition on their Labs website to discover who has the nicest/untidiest/most interesting/most remarkable home lab. In this episode we present the lab from Rainer Schuster, who will be familiar to many Elektor readers.







Rainer's lab gives an extremely organised and tidy impression — something to be envious of… He writes: "My 'home lab' is nearly as old as (the German edition of) Elektor itself — at least as far as the equipment goes. It all started back in 1972 with a multimeter and a power supply, and over the years all the other equipment that you can see in **Photo 1** was added." Most of the test equipment were kit sets from the (long gone) company RIM Electronic and from the Heathkit, who, until the '80s, supplied justifiably famous and excellent kit sets.

On the workbench you can catch a glimpse of the project Rainer was working on when the photo was taken: a guitar amplifier for his son, based on the "Guitarix" app that runs on a Raspberry Pi 4.

But home-built equipment is also part of the lab, such as the "Transistor Curve Tracer" published in the February 2009 issue of Elektor (**Photo 2**) [1].

Photo 3 shows a couple of his own developments: the "HV Power Supply / Tube Curve Tracer" [2] (top) and the "Electronic Load for DC and AC" [3] (bottom), both of which have been posted on the Elektor Labs website.

And finally, Rainer's lab can also be seen in the video [4] which he made for the occasion of the *Elektor Video Olympiade* in 2017, where he earned an honourable third place with his description of a few of the infamous 'joke circuits' Elektor magazine used to publish in the annual *Summer Circuits* issue.

200086-03

We Links

[1] Transistor Curve Tracer: www.elektormagazine.com/magazine/elektor-200902/18980

[2] HV Power Supply / Tube Curve Tracer: www.elektormagazine.com/labs/hv-power-supply-with-tube-and-semiconductor-curve-tracer

[3] Electronic load for DC and AC: www.elektormagazine.nl/labs/electronic-load-for-dc-and-ac

[4] Video: www.youtube.com/watch?v=2f75lK12Wjo&list=PL2a34OA-WuybkArjgAnXiIFqUc3KYf19A&index=4

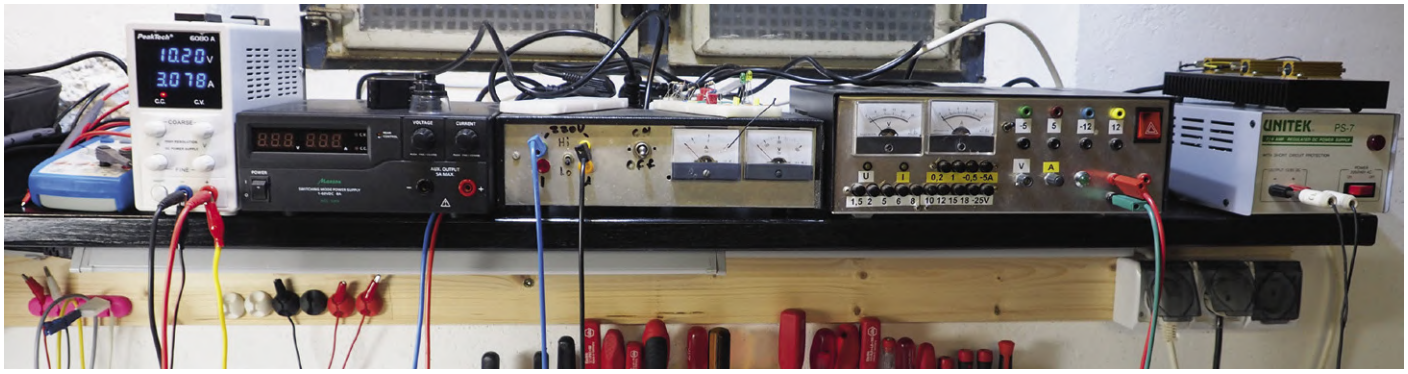
REVIEW

The PeakTech 6080 A Lab Power Supply

A neat lab PSU for not much money

By **Dr Thomas Scherer**

It's well known that electronics engineers never have enough power supplies. I'm no exception to this rule, as you can see from a photo of the home lab in my basement. The neat PeakTech 6080 A power supply is unusual amongst all the PSUs on the market today. It has been designed using linear technology with a traditional power transformer. The result is a quiet output voltage, highly accurate displays and all for not much money at all...



It's well known that electronics engineers never have enough power supplies. I'm no exception to this rule, as you can see from a photo of the home lab in my basement. It all looks a bit squashed up but I'm sure I could make room for just one more… The small PeakTech 6080 A lab power supply offers power and accuracy for a very reasonable price.
Actually, the home lab is already quite well equipped with power supplies. A 12 V 9 A unit (far right) is ideal to power small low-voltage drilling/grinding machines such as a Dremel or Proxxon etc. You can also see the homemade Elektor Precision Power Supply [1] from 1980! 40 years old and still accurate to 0.1%, good design is the key. Next in line is a relatively high-impedance mains voltage source with safety insulation and a load capacity of around 60 W, it's not beautiful, but it's practical and saves lives.
The digital lab power supply delivers a lot of power with a maximum of 60 V at 8 A — this one is obviously not homemade! Now the PeakTech 6080 A has been added to the lineup, it only delivers 3 A at 15 V max, but stands out because of its other features.



Figure 1: The lab power supply is protected by thick foam packing.

Unboxing

The PeakTech PT 6080 A power supply did not reach me via the normal route that other parcels have taken when they arrive at my door. This time I got a call from a neighbour who lives a few blocks away. He said his parents had seen some packages fall from a passing mail van carrying parcels and my name and address was on one of them. It was Christmas Eve so suspicious of a prank I asked if it was gift wrapped and maybe, was the van being pulled along by… reindeer? He remained serious so I thought it would be best to go and investigate. Sure enough it turned out to be the PT 6080 A power supply unit I had recently ordered. The package was a bit scuffed but not badly damaged.
You may well be wondering the significance of this anecdote: In fact it turned out to be a good drop-test for the power supply,



Figure 2: Front and rear view of the PT 6080 A (Image: PeakTech).

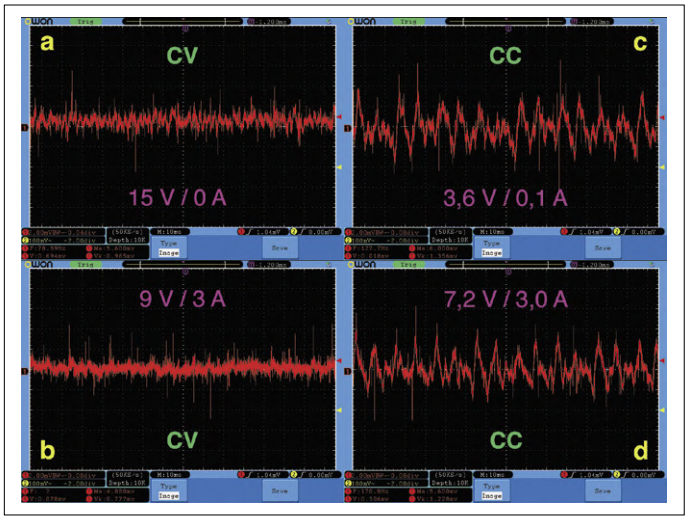


Figure 5. Output voltage noise levels under various conditions.

the packaging with its internal foam cushions **(Figure 1)** did an admirable job of protecting the unit. It survived unscathed and is fully functional.

The unit is 80 x 160 x 225 mm (W x H x D) and its layout is more 'portrait' than 'landscape' (**Figure 2**), which takes up less bench space. The thick, heavy heat sink on the back looks well proportioned to keep the power components cool. It is the first indication that this power supply unit is designed using more traditional, linear technology. This type of supply needs to dissipate high levels of power when delivering high current at low output voltages. The second indication of the chosen technology is the unit's weight: 2 kg is relatively heavy for a 45-W power supply. This indicates there is an iron or ferrite-cored transformer inside running at mains frequency. Switched-mode supplies will use smaller, lighter transformers switching at a much higher frequency.

The lab power supply has an output voltage range adjustable from 0 to 15 V via its two safety 4-mm banana sockets on the front panel which accept shrouded or standard banana plugs. Output current limit can be set from 0 to 3 A using the 'A' coarse and fine adjustment pots. Output voltage can be set similarly using the 'V' coarse and fine pots. Two four-digit displays show both the output voltage and current level. Two LEDs labelled 'C.C.' and 'C.V.' indicate whether the unit is operating in constant current or constant voltage mode. **Figure 2** shows the C.V. LED lit. When the current limit is reduced from its maximum setting the C.C. LED will be lit instead.

Precision on display

What really interests me with regard to power supplies is the accuracy of the displayed voltage and current levels. I also have a digital power supply unit, where the voltage and current are set digitally using two rotary encoders. This switch between coarse and fine adjustment (in a ratio of 1:10) can be toggled by pressing the knob. Budget priced digital power supplies usually only show the value of what has been setup, not what is actually coming out of the sockets. As a result, there are always small discrepancies in the range of 100 mV or a few tens of milliamps in the displayed value and the actual value. Cheaper digital power supplies often cut costs by omitting the digital voltmeter chip thereby making the displayed values less representative.

The PT 6080 A is different. Instead of encoders, normal potentiometers are used here. The 'gear ratio' from coarse to fine adjustment is about 1:7. For me, the analogue setting adjustments took some getting used to. Unlike digital power supplies, you do not immediately see what you are setting. The applied voltage and current flow are measured digitally and average A/D converter chips can manage about three or four measurements per second so there is always a slight delay between the pot movement and the displayed values. I wouldn't say this is bad, just different from the digital lab power supply I'm used to. **Figure 3** shows an impressive aspect of this power supply: The displayed values are almost identical to those of shown on my multimeter. There is a discrepancy of 10 mV = 0.07 % for the voltage and 0.3 mA = 0.08 % for the current (the deviation is similar at 3 A). This is surprising, I calibrated the



Figure 3: Displayed value vs DVM measurements (voltage above, current below).

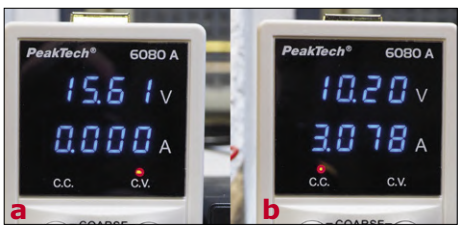


Figure 4: Maximum voltage (a, CV) and maximum current (b, CC).

multimeter against a 0.1 % accurate reference voltage source before I started measuring. The PT 6080 A is therefore much more precise than the manufacturer's specification of 0.5 % ±5 digits and you can have confidence in the displayed readings. I certainly didn't expect that!

Noisy or quiet?

The next important aspect is the electrical 'cleanliness' of the output voltage under load. A low-power linear lab PSU would not be chosen to primarily power digital electronic circuits. It is more suited as a power source for analogue, audio frequency applications or RF circuits. For these applications it is important to provide a supply free from noise spikes and interference. Except for very expensive laboratory equipment, the output voltages supplied by digital lab PSUs are almost always contaminated with interference and noise signals related to the switching frequencies (and harmonics thereof) used by the switched-mode controller. PSUs based on linear regulators should fare better in this respect, is that true of the PT 6080 A? The manufacturer states that the residual ripple of the voltage is 0.5 mV_{rms}. Figure 5 shows four screenshots of output voltage waveforms under different load conditions. The sensitivity (2 mV/graduation) and time base (10 ms/graduation) are the same for all of the images. Noise and hum in CV mode are within the specified range both under no load (a) and loaded (b) output conditions. The output voltage is not quite as good when in CC mode. Here the ripple is around 2 mV_{eff} regardless of the load. The clearly visible low-frequency 128 Hz (c) or 171 Hz (d) fundamental is not a harmonic of the mains frequency so is not likely to be mains hum.

Under the covers

To check out the build quality I removed six screws holding the sheet metal cover on and took a look inside (**Figure 6**). The heavy toroidal transformer takes up a lot of the internal volume. It uses a tapped primary winding so it can run on 230 V or 115 V mains. A 30-V winding with a centre tap supplies the front control/display board (**C**). On the back is the circuit board with a bridge rectifier made from four 5-A diodes and a 4,700 µF/50 V electrolytic capacitor to reduce the ripple. A power transistor is mounted on the back heatsink area. The small black box to the right of the diodes is a relay, which can be heard switching when the output voltage is adjusted around 7.5 V. Below this threshold, the relay switches a transformer secondary winding tap of 12.5 V to the rectifier, above this threshold it switches to the 21.5-V tap. This mechanism reduces the power dissipated in the output transistor when the PSU is operated at high current with a low voltage output setting.



Figure 6. Inside the PT 6080 A is a mains transformer, power board and controller board with SMDs.

To sum up

By choosing the PeakTech PT6080A (available from the Elektor Store for less than €60) you will be getting a good quality, low noise power supply with digital displays showing impressively accurate output voltage and current readings. The electronic components and PCB are of good quality and the unit layout is well designed. The output voltage has no measurable mains hum and is well regulated, showing very little drop under load. The output voltage noise level in CV mode is good and the power supply does not get particularly hot in use, even under high, continuous load conditions. This is mostly due to the low-voltage switchover by the relay, which is a feature not often seen in linear bench power supply units in this price bracket. The unit is also rated as short-circuit proof.

All in all I was impressed by this small lab power supply and would give it a 'recommended' rating. ⏮

200009-03

4 SALE @ WWW.ELEKTOR.COM

→ PeakTech 6080 A DC Lab Power Supply
www.elektor.com/19111

Links and literature

[1] Elektor Precision Power Unit (1980): www.elektormagazine.com/magazine/elektor-198009/44524/

[2] PT 6080 A, Data: www.peaktech.de/productdetail/kategorie/acdc-stabilisierte-labornetzgeraete/produkt/peaktech-6080-a.html

welcome in your ONLINE STORE

EDITOR'S CHOICE



JOY-iT Joy-View 13 Touchscreen

It's not easy to find a small, high-quality monitor that is conveniently portable – the options are fairly limited. The Joy-View 13 makes the choice a lot easier, and thanks to the presence of both HDMI and USBC ports, it can be connected to virtually all modern devices.

The Joy-View 13 portable touchscreen monitor is a nicely finished unit that provides excellent video and can be used in many applications and with many different devices thanks to the built-in capacitive touchscreen, the versatile connection options and the variety of power supply options. If you are looking for a monitor that is compact and easily portable, the Joy-View 13 is an excellent choice. And if you would like to have something a bit larger, you can also opt for the 15.6-inch version of this monitor, the Joy-View 15.



Harry Baggen (Elektor Labs)

www.elektor.com/19162 | www.elektor.com/19163

Elektor Bestsellers

1. DVD Elektor 2010 through 2019
   www.elektor.com/19194



2. Explore the Raspberry Pi in 45 Electronics Projects (3rd Edition)
   www.elektor.com/19190
3. Raspberry Pi 4 B (2 GB)
   www.elektor.com/18965
4. Programming with Node-RED
   www.elektor.com/19224
5. Microcontroller Basics with PIC
   www.elektor.com/19188
6. Raspberry Pi 4 B (4 GB)
   www.elektor.com/18964
7. Elektor Raspberry Pi Electronics Kit
   www.elektor.com/19187

Explore the Raspberry Pi in 45 Electronics Projects (3rd Edition)



The completely revised 3rd edition describes 45 exciting and compelling projects, such as a flashing lights, driving an electromotor, processing and generating analog signals, a temperature control and a lux meter. More complicated projects are also included, such as a motor speed controller, a web server with CGI (Common Gateway Interface), and client-server applications.

Member Price: €31.46

www.elektor.com/19190

JOY-iT JT-RD6006 DC Power Supply Bundle

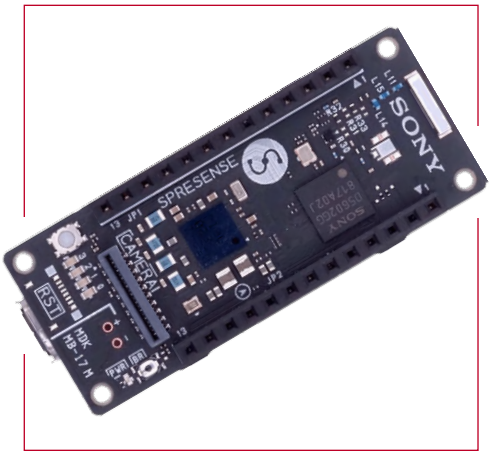


This bundle (available exclusively from Elektor) contains the programmable laboratory power supply JT-RD6006, an industrial power supply unit (60 V, 6 A), a large aluminium housing and a plug-in WiFi module which enables the JT-RD6006 to be controlled by other (mobile) devices via wireless connection. The bundle will be supplied in a practical hard case.

Member Price: €175.46

www.elektor.com/19211

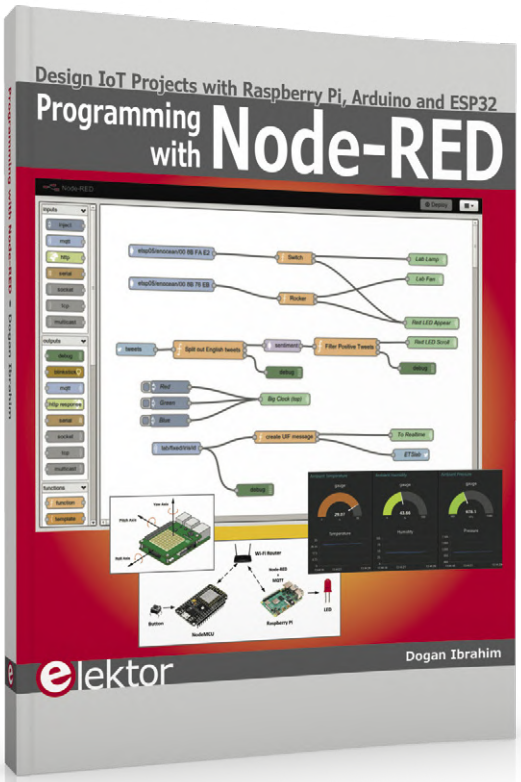
Sony Spresense Main Board



Spresense is a compact development board based on Sony's power-efficient multicore microcontroller CXD5602. It allows developers to create applications in a very short time and is supported by the Arduino IDE as well as the more advanced NuttX based SDK. The main board uses a processor developed by Sony for IoT and sensing applications. The main board can be operated alone or with the extension board.

Member Price: €71.96

www.elektor.com/19233



Programming with Node-RED

NEW

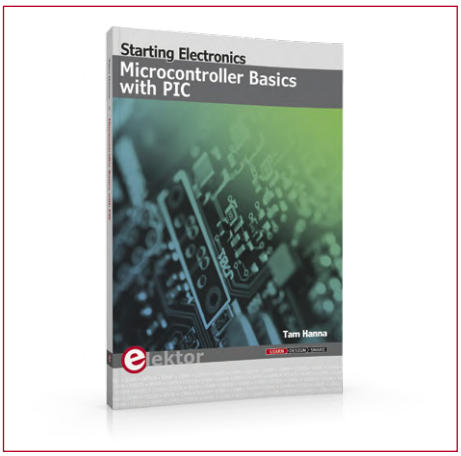
Produced by IBM, Node-RED is an open-source visual editor for wiring the Internet of Things. Node-RED comes with a large number of nodes to handle a multitude of tasks. The required nodes are selected and joined together to perform a particular task. Node-RED is based on flow type programming where nodes are configured and joined together to form an application program. There are nodes for performing complex tasks, including web access, Twitter, E-mail, HTTP, Bluetooth, MQTT, controlling GPIO ports, etc. One particularly nice aspect of Node-RED is that the programmer does not need to learn how to write complex programs.

The aim of this book is to teach how Node-RED can be used in projects. The main hardware platform used with most of the projects in this book is Raspberry Pi 4. Chapters are included to show how Node-RED can be also be used with Arduino Uno, ESP32 DevKitC, and the ESP8266 NodeMCU microcontroller development boards.

Member Price: €33.75

www.elektor.com/19224

Microcontroller Basics with PIC

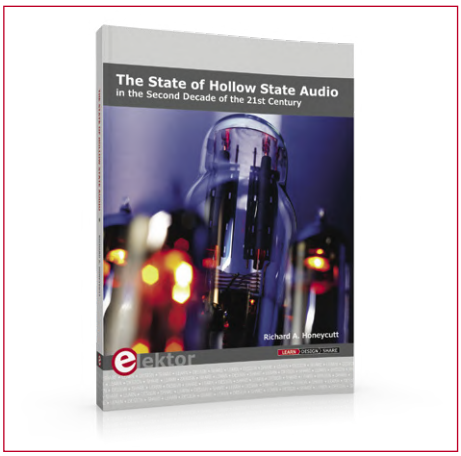


In this book the author presents all essential aspects of microcontroller programming, without overloading the reader with unnecessary or quasi-relevant bits of information. Having read the book, you should be able to understand as well as program, 8-bit microcontrollers. The introduction to microcontroller programming is worked out using microcontrollers from the PIC series.

Member Price: €31.46

www.elektor.com/19188

The State of Hollow State Audio



Audio tubes are currently experiencing a comeback, especially in the music and high-end audio industry. This new book provides answers to the following questions, among others: Do tubes produce better sound and how do hollow-state circuits work? How do you design hollow-state audio circuits? Can you recreate some of the classic hollow-state audio devices for modern listening rooms and recording studios? How can you intelligently modify hollow-state amplifiers to your taste?

Member Price: €31.46

www.elektor.com/19170

Tactigon ONE – Gesture & Motion Controller with AI



The Tactigon ONE is the ultimate wearable board, programmable with Arduino IDE. It features GPIOs, UART, an integrated battery recharge system, a fast microcontroller, and Bluetooth LE onboard! Measure linear and angular motion with a 3-axis gyroscope and 3-axis accelerometer. An extra 3-axis magnetic sensor provides more precision. The high-speed microcontroller unit and Arduino SDK allow for complex sketches execution.

Member Price: €71.96

www.elektor.com/19218

A Big Challenge

Safe products in the IoT era

By **Tessel Renzenbrink** (The Netherlands)

Seat belts, airbags, crush zones, speed limits and combatting drunk driving – over many decades, technological innovations and regulations have made cars ever safer. The same applies to other consumer products. We buy them with confidence and trust that they are safe. But the advent of the Internet of Things (IoT) opens a new Pandora's box. With more and more products being equipped with electronics, software and network connectivity, we are exposed to new dangers and risks. There are plenty of examples, such as networked toys that can be used to spy on children or vehicles with control systems that can be hacked and taken over remotely.

Professor Ross Anderson and his colleagues have investigated which measures are necessary to combine traditional product safety with the advent of the IoT. They did this on behalf of the European Commission. The EU is working on new legislation to ensure product safety in the future. They asked Anderson to identify what is necessary for this. Anderson, a professor of security engineering at Cambridge University, carried out the study with his colleagues Eirann Leverett, Senior Risk Researcher, and Richard Clayton, Security Researcher. The report with their findings was published in 2017 [1].

Trilemma

Anderson and his colleagues concluded that we still have a long way to go. The crux of the issue is that the way we ensure the safety of physical products is fundamentally different from how we ensure the safety of digital technologies. Products are tested and inspected before they are put on the market, and this is assured by a safety certificate or an approval mark. If there are significant changes to the product, it must be certified again. By contrast, the safety of software is essentially dependent on changes. Software is constantly monitored and regularly revised with patches and updates. Anderson addressed this in more detail in a presentation on their study at the Chaos Communication Congress (CCC) in December 2019 [2]. He called this the 'trilemma' of IoT products. If you stick to pre-market certification, you cannot modify your software, which means your product is unsafe. If you modify your software, you lose your certification. And if you combine certification with updating, which means going through the certification process again after each software update, the costs go through the roof.

Lifetime and complexity

Traditional products and digital services each have their own characteristics, which are now being combined. According to Anderson, we need to avoid ending up with the worst features of both worlds. For example, consider phones versus vehicles. The firmware and operating systems of phones are typically supported for about three years. After the support period ends, a phone becomes unsafe because the vulnerabilities of the software are no longer corrected. People are forced to replace their phones, even though the hardware is still perfectly okay. We need to make sure that this approach is not followed in the automotive industry in the future, says Anderson.

However, lifetime maintenance comes with its own challenges. Suppose a manufacturer is required to support a particular vehicle model for at least 20 years. A software engineer who writes a program for a vehicle with a planned market launch in 2022 will have to take into account that the code must continue working until 2042.

Complexity is also a challenge. The idea of the IoT is that devices can communicate with each other and take action independently. For example, autonomous vehicles that exchange information with the road infrastructure or refrigerators that do their own shopping. From a safety perspective, this means that testing devices on their own is no longer sufficient. It is also necessary to examine the risks arising from the interaction with other devices.

Recommendations

Product safety in the IoT era is a complex issue that does not have any simple solutions. In their report to the European Commission, the three researchers made several recommendations. The main recommendation is to establish a new European agency, dubbed the European Safety and Security Engineering Agency (ESSEA). The ESSEA must have the technical expertise necessary for tasks such as developing standards and informing policy makers. In addition, it must house a central database holding information about software vulnerabilities, failing components and errors in system integration. Presently this sort of information is held by a variety of agencies and companies. Bringing this information together would enable

people to learn from each other and avoid making the same mistake twice.
Another recommendation is that manufacturers that add networking capabilities to their products must ensure that it is possible to perform software updates.
In addition, the EU should amend a number of existing directives and regulations to accommodate the new developments. The Product Liability Directive, for example, should be extended to cover digital services. Under the provisions of the current directive, a manufacturer of navigation devices can be held liable for errors, but the makers of Google Maps cannot be held liable.



Photo: Gabor Kiss safety door. CC BY 2.0 https://www.flickr.com/photos/-nevi-/5404057758/

The EU as a line of defence against an Internet of Trash

During the presentation at the CCC, Anderson commented that has proven difficult to get the recommendations implemented. There are many different parties involved, each with their own interests. For example, Facebook and Google lobbied against the proposal to subject digital services to existing liability regulations. However, a number of things have been achieved since the publication of the report. For example, Regulation No. 2019/771 has been adopted, which stipulates that consumers of products with digital elements have the right to updates for at least two years, or longer if that can reasonably be expected. The latter means the products that are considered to last longer, such as washing machines, should also receive updates for a longer period.
The enormous task described by Anderson, Leverett and Clayton to ensure the safety of IoT products in the future contrasts sharply with the slow decision-making process in the European Union. However, in a related scientific article they write that their hope is fixed on the EU: “The EU is already the world’s main privacy regulator, as Washington doesn’t care and nobody else is big enough to matter; it should aim to become the main safety regulator too — or risk compromising the safety mission it already has.” [3]

200066-02

Links and literature

[1] Eireann Leverett, Richard Clayton, Ross Anderson, Editor G.Baldini, Standardisation and Certification of Safety, Security and Privacy in the ‘Internet of Things’, European Commission, Brussels, Belgium, 2017

[2] The sustainability of safety, security and privacy [video]. Presentation by Ross Anderson at the Chaos Communication Congress in Leipzig, 28 December 2019. : https://media.ccc.de/v/36c3-10924-the_sustainability_of_safety_security_and_privacy#t=1397

[3] Eireann Leverett, Richard Clayton, Ross Anderson, Standardisation and Certification of the ‘Internet of Things’, 2017

Elektor 1.2 GHz Multifunction Frequency Meter (1992/93)

Crossing the magic 1 GHz border

By **Jan Buiting** (Elektor Retronics Editor)

The project called “High Frequency Counter” dates back to December 1992, early 1993 as far as the three article instalments go in Elektor magazine. Browsing the back issues, it was a time when the major projects developed in the lab reached a certain level of industrial design and ergonomics, meaning the enclosure and the front panel layout were important factors in the whole design, as well as software control, easy testing by the user, and an option to buy the project as a kit of parts with a book complement.



As with many instalments I’ve made so far for the Retronics series, the subject was not chosen randomly from the vast Elektor archive stretching back to No. 1 from 1974. Rather, it’s because the item has survived Elektor’s unstoppable custom of consigning “hardware” to the bin after a few years, especially lab prototypes. In this case, the bin was actually a large dumpster placed under the lab windows at the former Elektor Head Office in Beek, Holland, back in July 2006. As with a few more instruments I’ve been able to describe in Retronics articles, I was able to prevent the subject of this instalment from taking a 10-foot flight out of the window and a hard landing, so no need for me to “dumpster dive” at that time.

Not lost in translation

The discrepancy between the working title of the project and the actual magazine article title “1.2 GHz Multifunction Frequency Meter” in Elektor magazine is indicative of the steadfastness of the four ‘language groups’ working at Elektor HQ and always bending over backwards to localize their ‘national’ edition on the basis of material supplied by predominantly Dutch and

German lab staff. The job extended considerably beyond plain translating.

The German author of the project, Bernard C Zschocke, originally submitted his project as “Hochfrequenz-zähler”, which having passed the lab’s German-English dictionaries, eventually ended up on the instrument front panel decal as “High Frequency Counter”. With hindsight, not the best name for the instrument as it expresses neither what’s being ‘counted’ and how ‘high’ the frequency went. So at least in the English department where I worked, the article series was duly headed: *1.2 GHz Multifunction Frequency Meter*. I remember discussing with some editorial and lab colleagues at that time that ‘frequency counter’ is a repeatedly seen and heard misnomer. You cannot count ‘frequency’ — only occurrences (like pulses) per unit of time (like seconds). In vain, the wording “Frequency Counter” slipped through and appeared in capitals on the instrument front panel.

The author’s name was a tongue twister with its bewildering sequence of four consonants in a row and it got typo’d and mangled many times except of course in Elektor’s German department. Fortunately, none of these mishaps reached the Elektor audience. Rumour has it though that for the German edition only, the author had requested the pseudonym “Urban” which at one point got mangled to ‘U-bahn’ by a certain editor.

Back in 1993

Almost 30 years ago, the basic course a project would follow towards final publication in Elektor magazine and PCB sales was not fundamentally different from today. Authors would send project submissions in a stamped envelope and patiently await the *Go/No-Go/Maybe Later* verdict from the editorial team, given by post or telephone. After a (negotiated) ‘Go’, the project would be presented as an engineering draft to the lab team who were supposed to make the electronics repeatable for home construction (or even kitting), design a matching PCB, and create a software product like an EPROM. In the case of the 1992/93 frequency meter, the process took approximately a year although Mr Zschocke’s original material including his writeup was quite good, requiring little ‘post engineering’ and ‘editing’ in Beek.

According to part 1 of the article series, the project was “a worthy successor to our famous 1985 microprocessor-controlled frequency meter, thousands of which are in use all over the world”. That 1985 project (anyone have one? — let me know) was a power-hungry, 250 MHz, TTL component jungle compared to the lightweight, portable version from 1992 which featured an 80C32 microcontroller, a U664 (or equivalent) dedicated prescaler, an LCD, and battery powering. The upgrade was substantial, justified and a good reflection of what could be built by hobbyists back in 1993 with a good chance of success. And crossing the magic 1 GHz barrier!

Remarkably and perhaps wisely, the 1992 design was not compared in the articles to any professional instruments available back then like HP and Tektronix. Still, the project came with impressive specifications, a few of which I’ve reproduced in the **Concise Specification** textbox.

Is this vintage?

Looking at the 1.2 GHz Multifunction Frequency Meter schematic in **Figure 1** (reproduced from a scan) I could not suppress a tendency to “recast” the design using components from the year 2020. The use of shaded sections in the original schematic

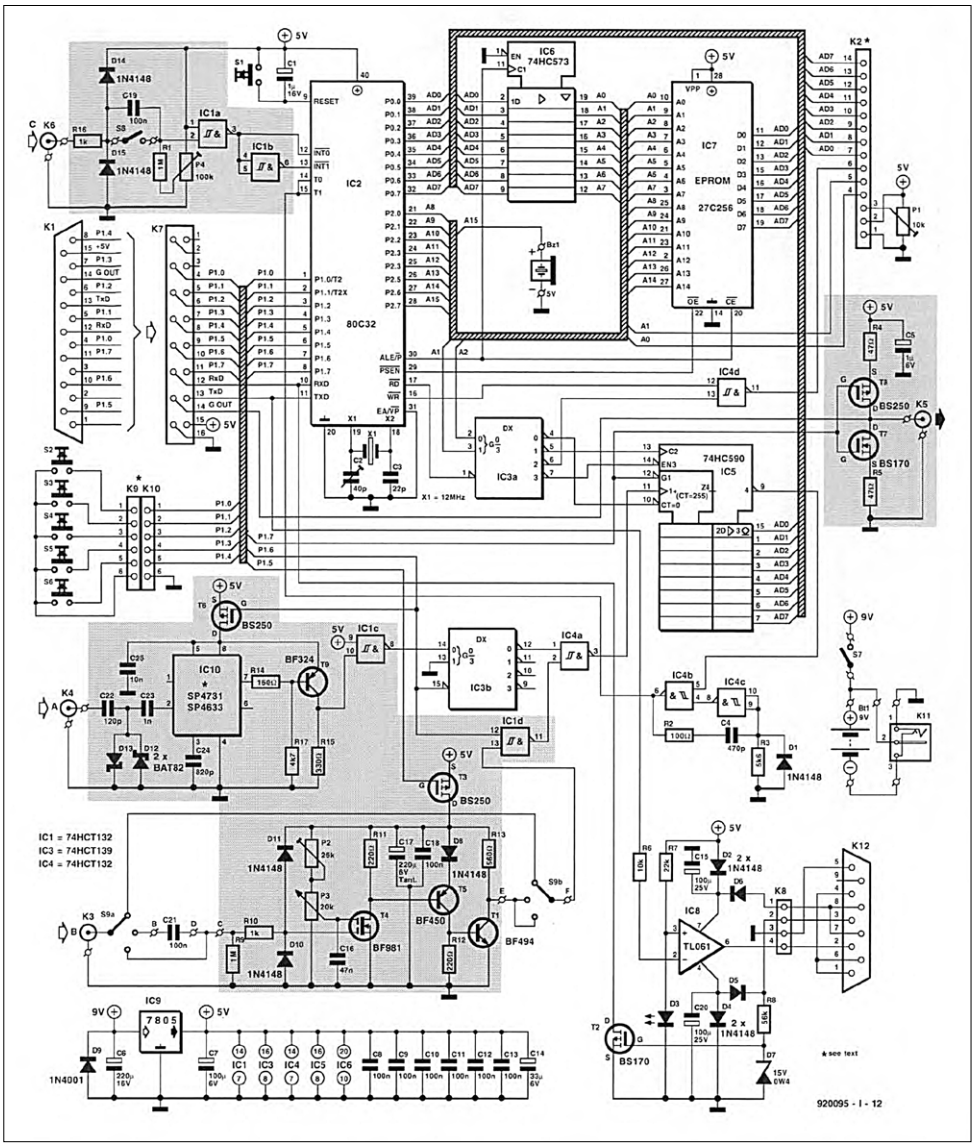


Figure 1: The schematic of the 1.2 GHz Multifunction Frequency Meter reveals a mid-1990s-style marriage between a 40-pin DIP cased microcontroller (here, a 80C32) and discrete electronics. With so much functionality carried out by software, the resulting BOM was remarkably small. (scanned reproduction)

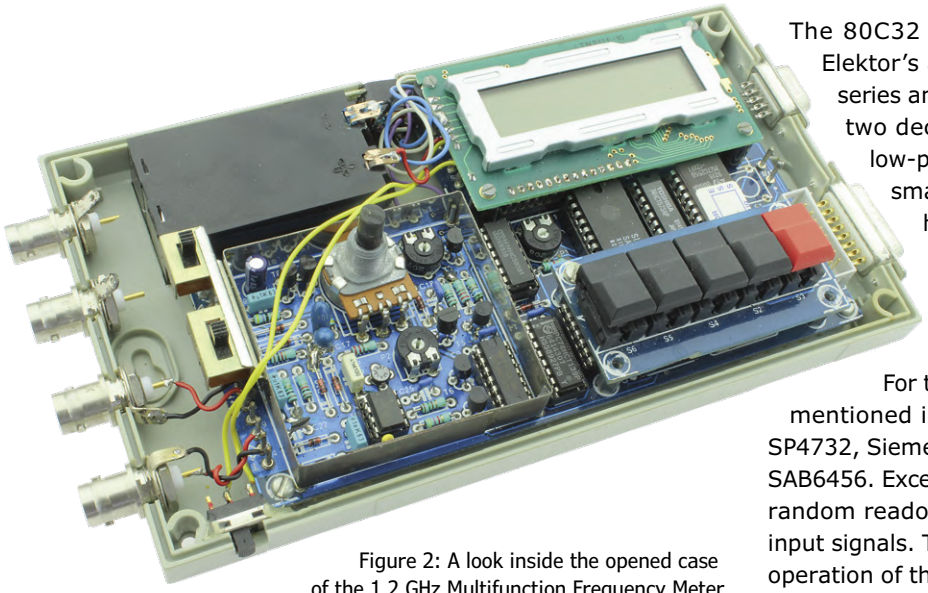


Figure 2: A look inside the opened case of the 1.2 GHz Multifunction Frequency Meter. Note the metal screening around the prescaler area where frequencies up to 1.2 GHz may appear (fingers crossed).

and the general quality of presenting a complex design are a great help. As a mental exercise only:

- 80C32, EPROM, IC5, IC3 (MCU section) → Arduino, ESP32
- K12, IC6, T2 and surroundings (pseudo RS-232) → a USB port
- 7805 (supply) → some recent LDO regulator
- 9 V supply → 6 V supply
- LCD → OLED
- BAT82's → HP28xx ultra low capacitance diodes

Concise Specification

Measurement modes:

- Frequency meter: 1 mHz to 1.2 GHz
- Period: (1/frequency): 1 ps to 4,000 s
- Revolution counter: 0.001 to 4×10^6 rev/min
- Event (pulse) counter: 1 to approx. 4×10^9
- Zero counter: 1 to 4×10^9

Generator modes:

- Timer: 1 µs to 4,000 s
- Pulse generator: 8,500 s (0.117 mHz) to 4 µs (250 kHz)
- Duty factor: 1:1 to $1:4\times10^9$
- Period number: 1 to 4×10^9

User-defined settings:

- Single/continuous measurement
- Floating gate time
- Intermediate measurement result display
- Period analysis (accurate measurement of low frequencies at short gate periods)
- Quiescent level
- Pulse polarity
- Sounder on/off

The 80C32 was a big thing at that time, coming from Elektor's apparent infatuation with the Intel 8051/8052 series and derivates which remained popular for at least two decades especially in Germany. The 'C32 was a low-power CMOS MCU using external (EP)ROM and a small amount of internal RAM, hence excellent for hobbyists. Countless projects of the more serious level that appeared in Elektor had the winning combination: 80C32 + 27C128/256/512, the latter available ready-programmed through the Elektor product service (EPS).

For the GHz prescaler chip, several alternatives were mentioned in the article, including Plessey's SP4731 and SP4732, Siemens' SDA4212, Telefunken's U664B, and Philips' SAB6456. Excepting the SP4732, all tended to oscillate causing random readouts on the display especially on no or too low input signals. This is normal however and a result of the basic operation of these chips. Come to think of it, I've been ignoring the random readouts of my home-built 600 MHz counter with an 11C90 prescaler chip and considered it a sign of utter sensitivity, not unlike a reflex receiver on the border of "going Mexican" i.e. self-oscillation. This design did have a 15 dB wideband preamp ahead of the prescaler, a thing I believe the Elektor 1.2 GHz counter is lacking.

The case for a case

A good deal of effort went into the design of a professionally looking case for the counter, including the "three-colour" front panel adhesive which was available ready made from EPS. As you can see from the picture of the opened case in **Figure 2**, the PCBs fitted perfectly in the Vero case (with two shades of grey). A full page in the article is dedicated to the front panel drilling template. The PCBs were:

- Main board (with a cutout for the PP3/6LR61 battery holder)
- Pushbutton board (for Digitast switches)
- LCD board (Philips LTN211-F10, replaced later by Sharp LM16A21 and Hitachi LM016L)

The part covering the construction the project at home from individual parts and pieces is of a length and detail you'd be hard pressed to find in today's electronics press dominated by embedded platforms. The prescaler area on the board, for example, had to be fully screened with tin plating, including a cover. Looking at Figure 2 again, the prescaler IC is seen inserted in an 8-pin DIL socket, probably with component exchangeability in mind. I would have expected to see this very high frequency chip soldered directly onto the board to minimize stray capacitances. Also, those twisted wires between the BNC socket and the 'A' input of the board are poor practice to say the least (the highest frequency available in the labs at that time was 512.000 MHz — and it still is today from a vintage HP 8640B RF generator).

Testing and aligning

The test procedure for the project reflects its 'early embedded' nature where an instrument's firmware has a number of routines built in to interrogate and test hardware components, and show the results on its own display. The overall test routine can be divided into three main groups

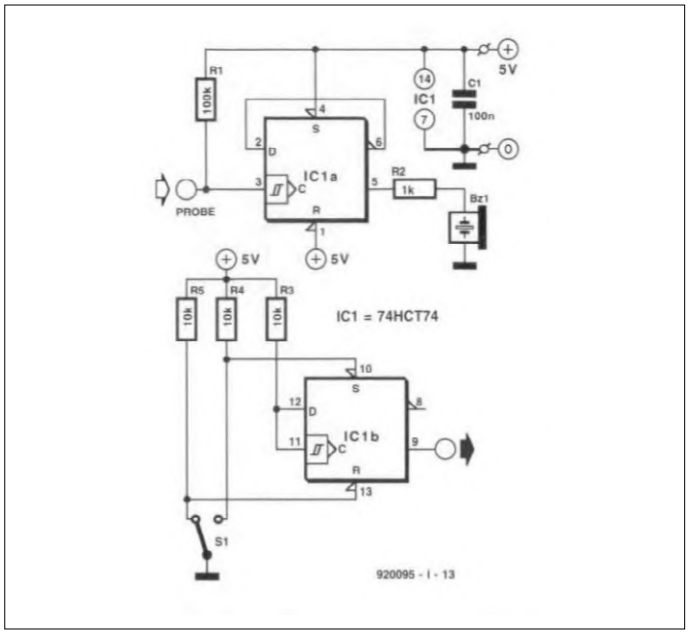


Figure 3: You can't test everything by poking bits and bytes around! In some cases, old fashioned hardware can come to the rescue. Here, a single 74HCT74 chip acts both as a logic signal generator and a logic probe with audible output. Wow!



Figure 4: The GHz Counter project bundle sold by Elektor Germany and some kit suppliers comprised a booklet and a very comprehensive kit of parts. The bundle first appeared in 1995 i.e. well after the publication of the GHz counter in Elektor magazine.

- LCD
- Main board
- Serial I/O

Assuming the CPU, EPROM, LCD, xtal and address latch function properly, the test routine 'menu' is kicked off by pressing the MENU or ENTER key at power on, and the same key combination to leave the menu (BREAK). The three subsystem tests look pretty exhaustive and must have been a good aid to localizing trouble in such a complex circuit.

Although I find it hard to imagine, some users of the instrument may not have had an oscilloscope available to aid in testing the counter. Not having a PC with an RS-232 port wasn't as bad though as the relevant test covered a more or less optional feature of the counter: remote control and readout.

The simplicity and old-school design of the auxiliary tester that appeared in the magazine I found worthy of reproducing here in **Figure 3**. A single 74HCT74 bistable does the job, acting as a simple pulse source and an audible 'digital probe' at the same time.

Like you I wondered about the frequency stability and resultant resolution of the instrument and was saddened by the expected "it all depends on the 12 MHz crystal". In the prototype a cheap computer xtal was used. The article says: "[adjusting the central oscillator] … is best done by applying a known, high-precision test frequency to the meter, and adjusting the trimmer capacitor C2 until this frequency is displayed". With 28 years hindsight the CPU should have been clocked at 10 MHz which would have allowed referencing from a professional frequency standard, an xtal oven, or even Elektor's own 10 MHz Frequency Standard. In today's money it would have been a *GPS disciplined oscillator*.

The Full Monty

About one and a half years after the magazine publications, the EPS board sales indicated further commercial potential of the project. The Elektor staff working in Aachen initiated to have kits put together by an advertiser in their magazine (probably Geist Elektronik) and sell it under the Elektor Germany brand, along with a book instead of photocopied articles. Again, this was a Germany-only undertaking and it may have been the first go of Elektor Germany's team at selling a bundle (or package) based on a design published in a magazine article. Today, the whole publishing house is doing it 24/7 in four languages.

The package, pictured in **Figure 4**, is rare today and apparently around one hundred were sold, which compares favourably against 2000 or so of the individual boards, EPROMS, and front panels. The white box contains the kit of parts to build the counter, that's including the predrilled case, EPROM, front panel adhesive, and all mechanical parts. The Heathkit times



Figure 5. Kit contents (1995). For the sake of authenticity, no attempt was made to empty all paper bags containing the electronic components inside.

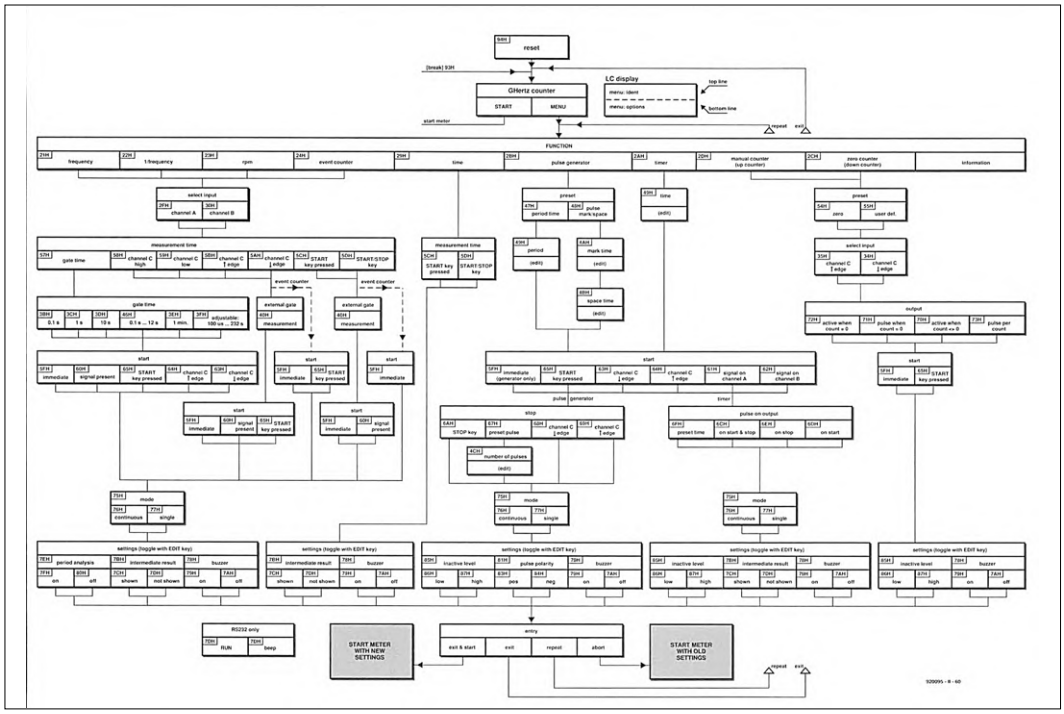


Figure 6. Alongside a description in words, a drawing was made back in 1993 of the counter's impressive menu structure and user options. The original was sized at 40 × 26 cm and came as a foldout sheet.

relive for a moment. The contents of the package I safeguarded back in 2006 is untouched and I was delighted to even find a length of solid, bare "you-never-know" wire and pieces of sheet metal to assemble the prescaler screening. Resistors, sockets, ICs, pushbuttons, the display, BNC sockets, the programmed EPROM, it's all there and neatly packaged mostly in small, white paper bags (**Figure 5**).

The book in the package is another perfect companion. It packs all three instalments of the article (December 1992, January and February 1993) into a size just over that of the actual counter once assembled in the Vero case.

Menu please!

The copy preparation of part 2 of the article grinded to a halt and ran into some delay when a desk editor of the Dutch department at Elektor said that the instrument's menu should be elucidated and illustrated in proper detail. So far, most of the lab staff had not selected much from the menu beyond '*Press Enter to Continue*' let alone ventured into the myriad of options hidden deep in the software. So, Dick Lubben at his quiet desk in Beek spent about two working weeks charting and documenting the counter's software structure to the finest detail, culminating in an enormous sheet showing every option, display text and error code imaginable. It was a fantastic feat, all applauded and the result was printed on a full page in part 2 as well as supplied as a foldout sheet tucked in with the German book. **Figure 6** shows an attempt at reproducing the originally 400 × 260 mm sized sheet printed on glossy paper (!).

Tantalum Tantrum

I tried to make the instrument work by inserting a fresh 9 V battery in the holder at the underside, and powering up with the small slide switch. Nothing happened and when I probed the + supply wiring at a few points with a DMM I found about +3 V behind the on/off switch. It was obvious that something was loading the battery excessively and I immediately suspected one or more of the electrolytic capacitors on the supply rails, notably the tantalum ones C14 and C17. With hindsight I was happy I used a battery for a few seconds only and not a benchtop supply as it would have blown out a few componets. I intend to do the necessary repairs to see if the instrument actually works — I'm confident.

The three instalments describing the 1.2 GHz Multifunction Frequency Meter are in Elektor's online archive and accessible as pdf files to Green and Gold Members [1][2][3]. The article instalments are also on the Elektor 1990s DVD. Please note that the project is no longer recommended for new construction as several parts are no longer available. ⏮

190382-C-01

ESTD 2004

Retronics is a regular section covering vintage electronics including legendary Elektor projects.

Contributions, suggestions and requests for instalments are welcome, please telegraph the Retronics Conservator on: jan.buiting@elektor.com, subject: Retronics (STOP).

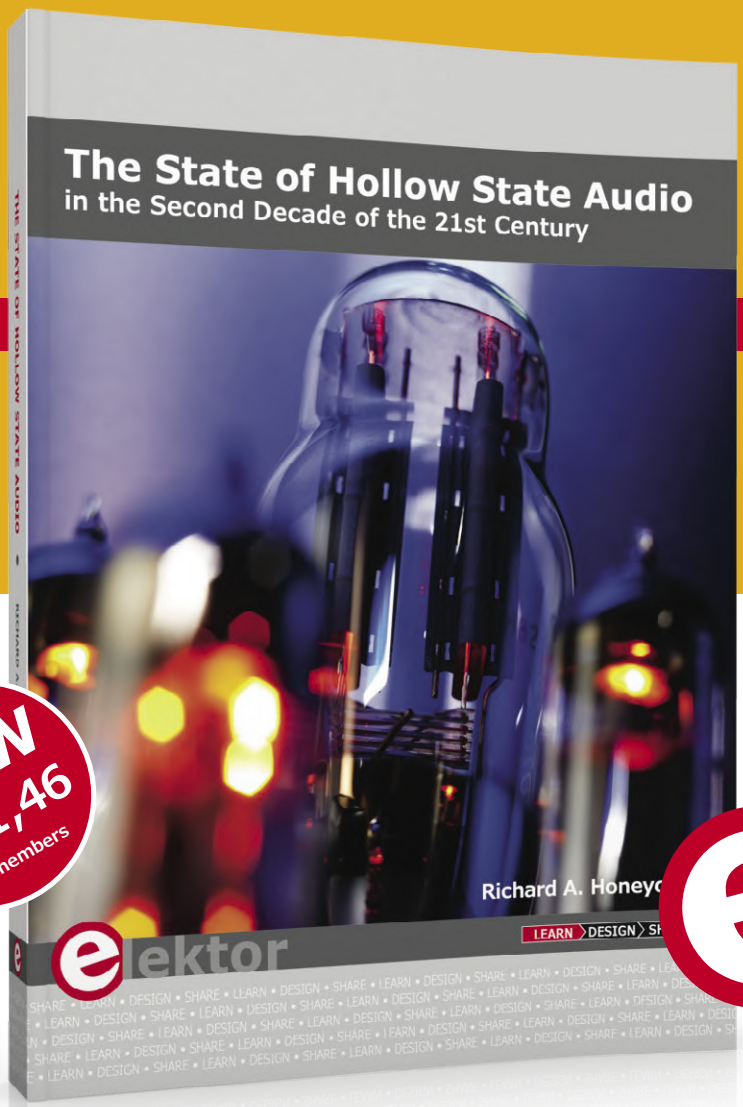
Weblinks

[1] 1.2 GHz Multifunction Frequency Meter, part 1. Elektor Electronics December 1992: www.elektormagazine.com/magazine/elektor-199212/32728/

[2] 1.2 GHz Multifunction Frequency Meter, part 2. Elektor Electronics January 1993: www.elektormagazine.com/magazine/elektor-199301/32787/

[3] 1.2 GHz Multifunction Frequency Meter, part 3. Elektor Electronics February 1993: www.elektormagazine.com/magazine/elektor-199301/32787/

The State of Hollow State Audio



NOW € 31,46 for members

Description:

Vacuum tube (or valve, depending on which side of the pond you live on) technology spawned the Age of Electronics in the early 20th century. And today, hollow-state equipment remains popular among high-end audio aficionados.

In The State of Hollow State Audio, Richard A. Honeycutt covers the history of hollow-state audio and then dives deep into the technical details, including hollow-state amplifiers, vacuum tube characteristics, and the differences in amp sound.

You'll learn: how hollow-state circuits work; how to design hollow-state audio circuits; how to find information on vacuum tubes; how to select tubes; how to troubleshoot hollow-state equipment; and more

elektor

Ready for a deep dive into hollow-state audio? Order The State of Hollow State Audio and get started.

Read more on:

www.elektor.com/the-state-of-hollow-state-audio

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 **$60.00 / £45.00 / €50.00 each**, which should encourage all Elektor readers to participate.

Participate!

Ultimately May 31, 2020, 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 2/2020 (March & April) is: **749DF**.

The book vouchers have been awarded to: Etienne Guyot (France); Ria Van Broeck (Belgium); Peter Tabatt (Germany) ; Tomas Bakke (Norway); Jose A. Gordo (Spain).

Congratulations everyone!

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

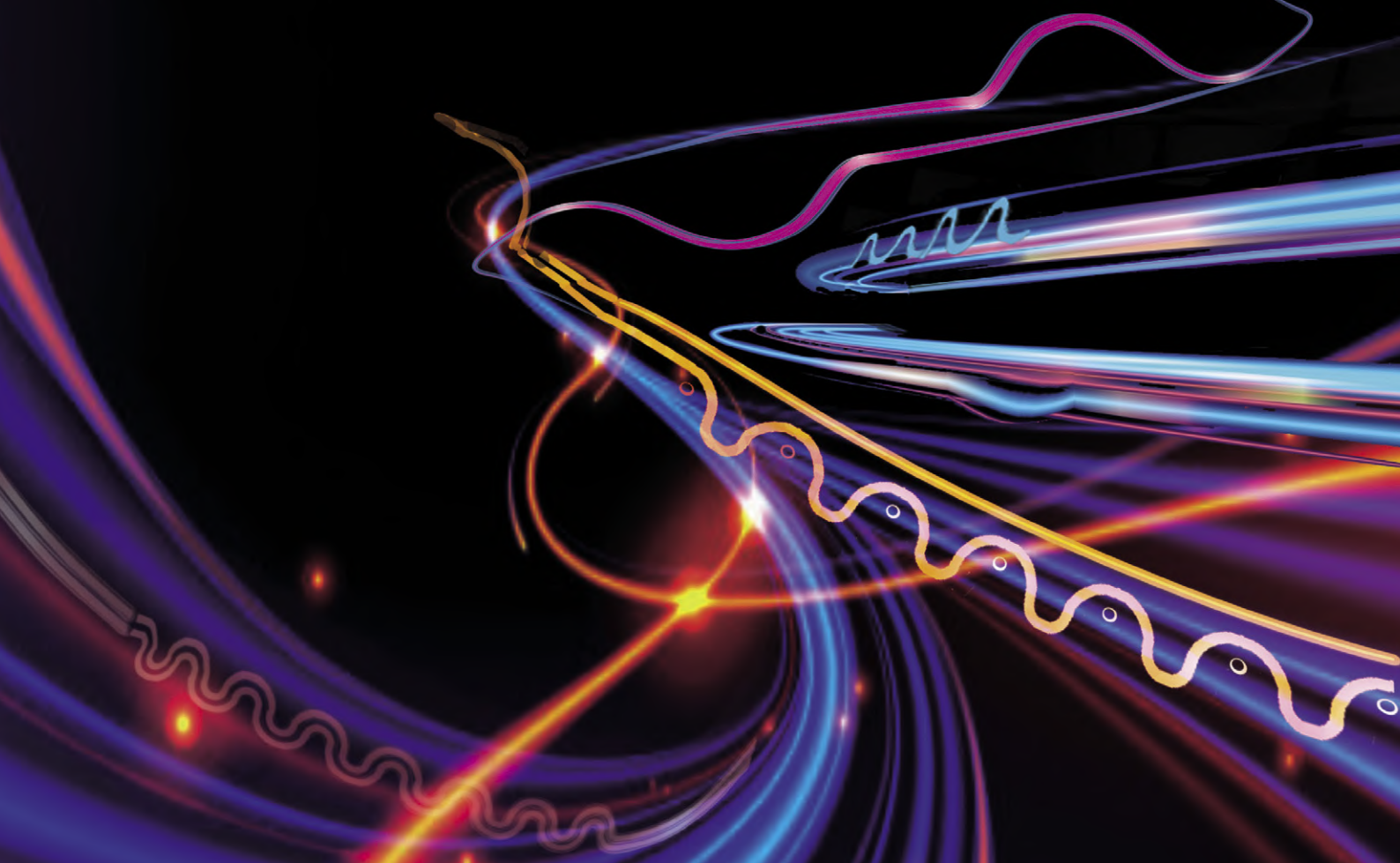
1	4	2	C	D	5	8	E	9	3	6	0	7	A	B	F
8	5	3	7	0	F	B	C	E	A	D	2	6	9	1	4
0	D	6	E	9	4	7	A	B	1	C	F	3	2	8	5
A	9	F	B	1	2	3	6	4	5	7	8	C	D	E	0
5	7	8	2	A	B	F	D	6	4	0	C	E	3	9	1
9	6	4	F	C	E	0	8	3	7	5	1	2	B	D	A
B	3	D	0	2	6	1	5	8	9	E	A	F	C	4	7
C	A	E	1	3	7	4	9	D	F	2	B	0	5	6	8
D	2	9	3	6	0	A	F	5	8	B	E	1	4	7	C
E	1	5	A	B	9	2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

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

PROTEUS DESIGN SUITE

High Speed Design Features

- Differential Pair Routing
- Length Matching / Net Tuning
- Automatic Phase Matching
- Use for USB, Ethernet, DDR3 etc.



Performance without the price premium.

Find out more and
configure your package on our
website or call the team on (+44)1756 753440

Labcenter Electronics

www.labcenter.com

electronica fast forward
the startup platform

powered by elektor

Win up to a
€75,000
marketing package from Elektor and exposure at the electronica trade fair

COMPETE TO LAUNCH YOUR STARTUP AT ELECTRONICA 2020



Is your electronics-related startup ready to disrupt? It is time to introduce your company's solution to the global market!

ENTER NOW! IT'S FAST AND SIMPLE

Visit **elektormagazine.com/fastforward** to read the Rules, Terms, and Conditions and to start an application. Good luck!

FINALIST BENEFITS

- **First Prize = €75,000** marketing package from Elektor, plus a booth at electronica 2022.
- **Second Prize = €50,000** marketing package from Elektor.
- **Third Prize = €25,000** marketing package from Elektor.
- **Get exclusive, specially priced (only €1,905!) booth space (5 m²)** at electronica 2020.
- Get **great exposure** before, during, and after electronica 2020. (10.-13.11.2020)

Sign up here

elektormagazine.com/fastforward

electronica Fast Forward is brought to you by

electronica Elektor