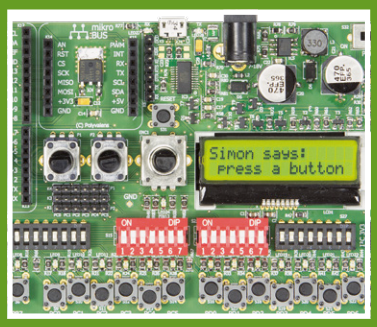
Edition 2/2017 | March & April 2017
www.elektormagazine.com

elektor

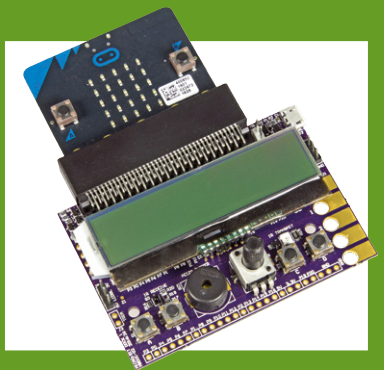
LEARN DESIGN SHARE



OBD2 Handheld
with Raspberry Pi
and new diagnose software



AVR Playground
Improves the way
to do Arduino



BBC micro:bit Dock
More I/O and easy access

RGBDigit Clock



colorful 7-segment display

Android on your Rpi part 1 • **Starter Kits and Development Boards** • **Swiss Pi Extensions** • Truly Random-Number Generator • **Debugging and Decoding Digital Communication using the SmartScope** • RFID starter kit for the Arduino Uno • **RGBDigit Clock** • IoT Gateway and Wireless Nodes • **Lead-acid Battery Activator 0-30V** • **NEC-style Remote Control Signals with Elektor Uno R4** • OBD2 Handheld using a Raspberry Pi • **Ethernet on the Android I/O Board** • AVR Playground • **Retronics: made in Belgium** • Walabot – a 3D Sensor for your Smartphone • **Dock for BBC micro:bit** • **Build Your Own Theremin** • **Gnublin 2 Linux Board** • PWM Motor Control • **Sensors Make Sense part 3** • Review: STEMlab125-10 • **Q&A Electric Motors** • Active Differential Probe v2 • **MicroPython** • **Peculiar Parts**

HIGH-PERFORMANCE 8-CHANNEL OSCILLOSCOPE

pico® Technology



The PicoScope 4824 is a low-cost, portable solution for multi-input applications. With 8 high-resolution analog channels you can easily analyze audio, ultrasound, vibration, power and timing of complex systems.

The PicoScope 4824 has the power and functionality to deliver accurate results. It also features deep memory to analyze multiple serial buses such as UART, I^2C, SPI, CAN and LIN plus control and driver signals.

Also features:

- High-performance arbitrary waveform generator
- Advanced digital triggers
- Serial bus decoding
- SuperSpeed USB 3.1 Gen 1 interface
- Windows, Mac and Linux software

NEW Software Update

Get the latest software release at
www.picotech.com/library/picoscope/picoscope-release-6.12.5

For more information please visit www.picotech.com/PS501

Prices are correct at the time of publication. Sales taxes not included. Please contact Pico Technology for the latest prices before ordering. Email: sales@picotech.com. Errors and omissions excepted.

Elektor Magazine
Edition 2/2017
Volume 43, No. 482
March & April 2017

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

www.elektor.com
www.elektormagazine.com

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

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

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

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

Advertising & Sponsoring:
Johan Dijk
Phone: +31 6 15894245
E-mail: johan.dijk@eimworld.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. 2017
www.eimworld.com
Printed in the Netherlands



Order must be

Based on discussions and correspondence with electronics engineers these past 2^5 years I estimate that roughly $\frac{1}{2}\sqrt{2}$ (70.7 per cent) of them never start reading here as the Publishers like to think. Instead, after removing the plastic wrapper and binning the address sheet, they dash off straight to a page showing an intricate or novel piece of technology, say, an RGBDigit Clock, something OBD, a Theremin, or a 1960's tubed oscilloscope. That's pages 6, 26, 46, 82 — and yes I admit I am also in the 70.7 % group.

As of this edition our French- and German language magazines are also on two-monthly publication at 132 pages per issue. This has permitted the joint lab/editorial/layout team to do a soft reset aiming at synchronizing the production scheme of articles and projects. Rather than forcing some existing material into the rigid Learn/Design/Share structure it was decided to enhance the article layout and abandon the fixed order and page allotments of the L/D/S sections.

Engineers can't do without labelling and classification though and consequently the page headers e-Labs Project and HomeLab Project have been retained and stronger than before emphasize the distinction between reader-contributed material on the one hand and tried and tested stuff developed and approved by our Labs. This distinction apparently has escaped many of you despite some of my feeble attempts here to elucidate, trigger and control. One notable section also honored with a page header block is Retronics, which now happily disrupts the flow of pages on new-fangled stuff rather than sit modestly at the back of the magazine always close to Hexadoku, another fireside favorite.

Signing off with this edition my valued colleague and former Elektor Netherlands Editor Harry Baggen is retiring after 38 years with the company. Many of you will know Harry not only from his magnificent audio projects like high-end amplifiers and loudspeakers, but also from technical correspondence and advice at which he excelled. The same for his writing of hundreds of articles in a manner best described as $S/N > 60$ dB.

Happy reading,

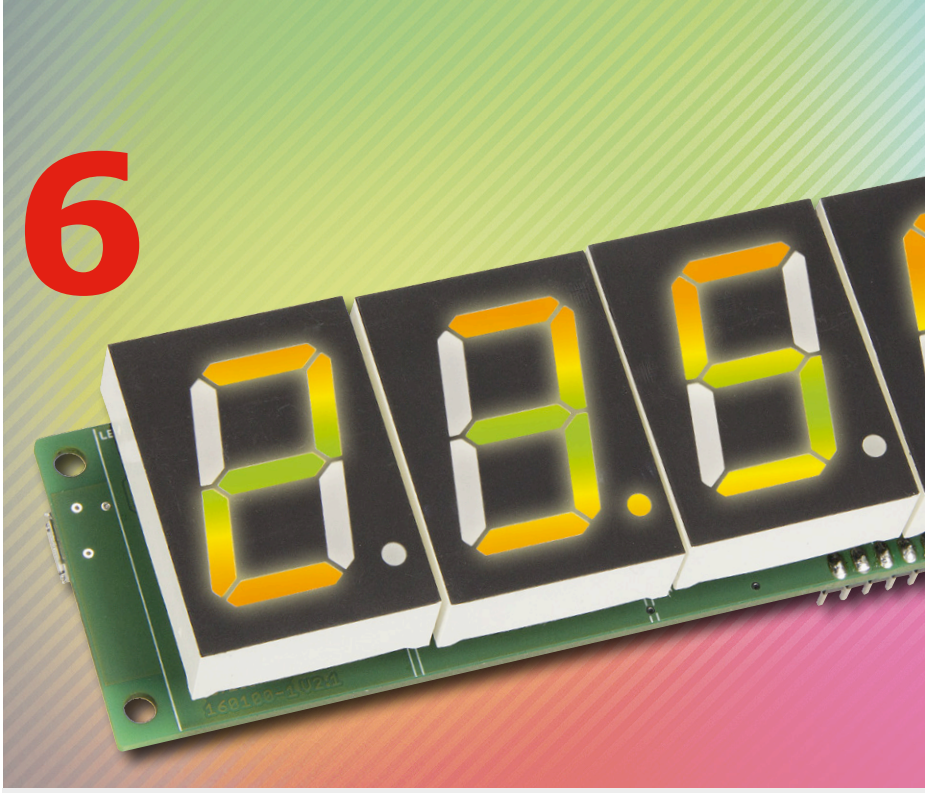
Jan Buiting, Editor-in-Chief

The Circuit

Editor-in-Chief:	Jan Buiting
Deputy Editors:	Thijs Beckers, Clemens Valens
Translators:	David Ashton, Jan Buiting, Martin Cooke, Ken Cox, Arthur deBeun, Andrew Emmerson, Tony Marsden, Mark Owen, Julian Rivers
Membership Manager:	Raoul Morreau
International Editorial Staff:	Thijs Beckers, Mariline Thiebaut-Brodier Denis Meyer, Jens Nickel
Laboratory Staff:	Ton Giesberts, Luc Lemmens, Clemens Valens, Jan Visser
Graphic Design & Prepress:	Giel Dols
Publisher:	Don Akkermans

This Edition

Volume 43 – Edition 2/2017
No. 482 March & April 2017



Regulars

11 The Elektor Community

31 Elektor Labs Pipeline

62 HomeLab Helicopter

82 Retronics: Made in Belgium
Bilingually, naturally

88 Q&A: Electric Motors
A first acquaintance

96 Peculiar Parts, the series
Bubble Memory

128 Elektor STORE

130 Hexadoku

Features

12 Starter Kits and Development Boards
A brief round-up

58 Debugging and Decoding Digital Communication
using the SmartScope

64 Review: STEMlab125-10
The little brother of the original Red Pitaya

108 RFID starter kit for the Arduino Uno
Temperature sensing and door entry system

112 Designing loudspeaker boxes
Free programs for speaker box calculations

Projects

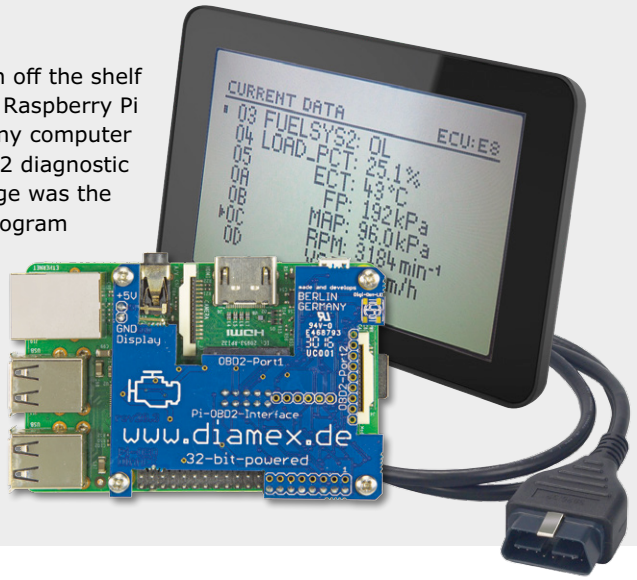
6 RGBDigit Clock
A colorful 7-segment display for your data

18 IoT Gateway and Wireless Nodes
Part 1: The hardware

OBD2 Handheld using a Raspberry Pi 26

With new diagnostic software

The Pi-OBD-HAT is an off the shelf add-on board for the Raspberry Pi which converts the tiny computer into a dedicated OBD2 diagnostic tool. One disadvantage was the need of a terminal program running on a PC for control. A much more convenient and neater solution is provided by the HHGui OBD2 diagnostic software.



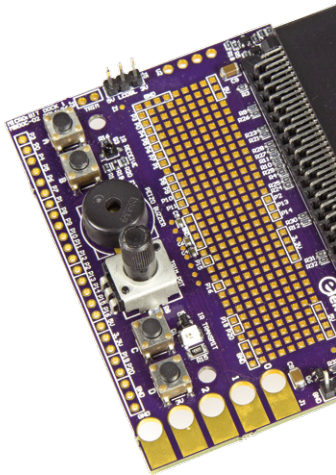
23 How to Produce NEC-style Remote Control Signals
Exclusively for the Elektor Uno R4

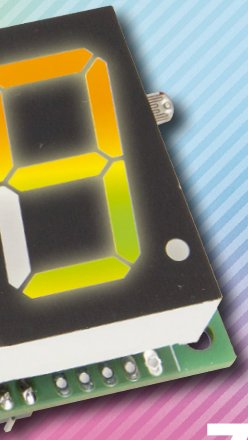
26 OBD2 Handheld using a Raspberry Pi
With new diagnostic software

32 Swiss Pi Extensions
Hardware extensions

42 Ethernet on the Android I/O Board
Using a USR-TCP232-T module

46 Build Your Own Theremin
Using JFETs instead of vacuum tubes





RGBDigit Clock

A colorful 7-segment display for your data

90 Dock for BBC micro:bit
"Base to micro:bit, you are cleared to dock."

97 PWM motor control
With added duty cycle boost

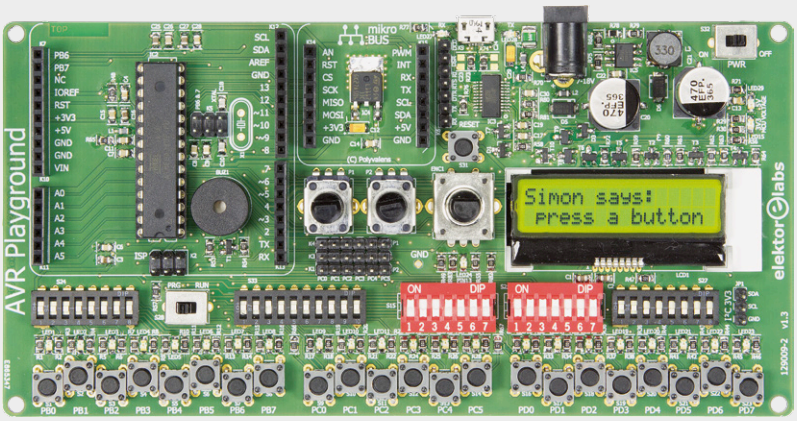
100 Sensors Make Sense (3)
For Arduino and more

106 Active Differential Probe v2
Now USB-powered

114 Gnublin 2 Linux Board
A build-it-yourself alternative to the Raspberry Pi

118 MicroPython
Python for small systems

120 Lead-acid Battery Activator 0-30V
Also shows the battery quality

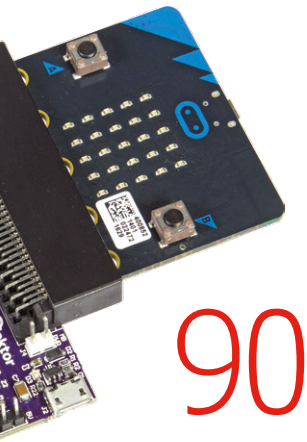


AVR Playground

Improves the way to do Arduino

The board presented in this article is a hybrid of an Arduino Uno and a traditional microcontroller development board, intended for 'doing Arduino' without bad connections and loose wires (although it doesn't disallow it).

50



90

50 AVR Playground
Improves the way to do Arduino

66 Truly Random-Number Generator
Meets NIST standard with cheap components

74 Walabot – a 3D Sensor for your Smartphone
Also sees through walls

76 Android on your Raspberry Pi (1)
Using GPIO Pins for measurement and control functions

Next Editions

Elektor Magazine 3/2017

Studio Mic Preamp • Li-Ion charger • Spycam detector • Outside Lighting System • L-Board — a T-Board alternative • nWatch • 4-Channel Remote Control with XBee • IoT Gateway and Wireless Nodes — Part 2: Software • and much more.

Elektor Magazine edition 3/2017 covering May and June is published on 13 April 2017. Delivery of printed copies to Elektor Gold Members is subject to transport.

Elektor Business Magazine 1/2017

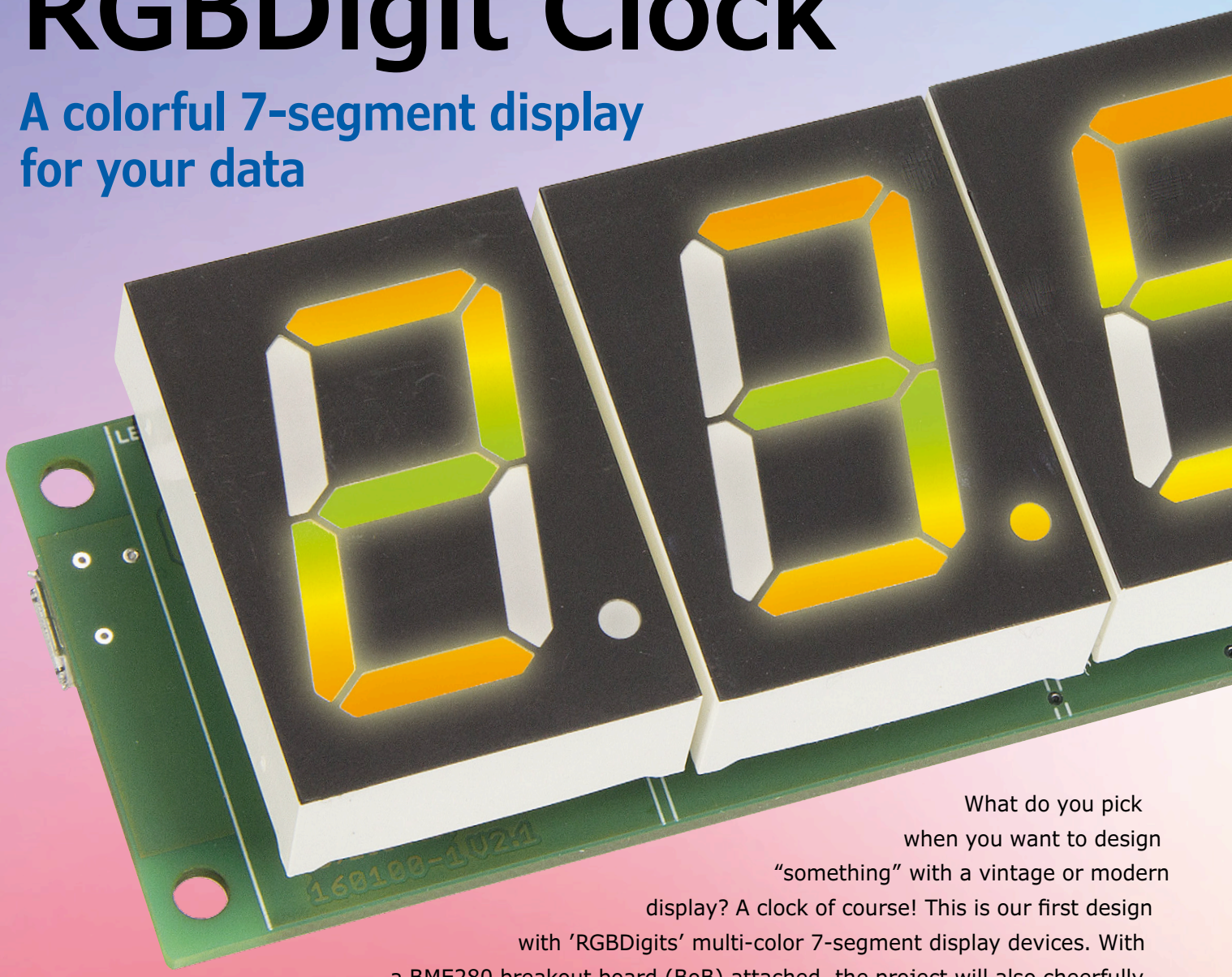
Embedded, Microcontrollers & Tools

Hubs Become Central to the IoT (Altera/Intel) • Server Based Solutions for Self-Organizing Networks (Artesyn) • From Makers to Market using 96Boards (Arrow); Industry Predictions for 2017 (ByteSnap) • WaWision (embedded projects) • Infographics • from the Garage up to Industrial Production (Loetronic) • LPCXpert (NXP) • Network Engine SoC (Renesas) • The Business Case for Commercial Embedded Linux (Wind River) • Talking Heads • A Desktop Prototyping Tool for Custom PCBs (Voltera) • and more.

Elektor Business Magazine edition 1/2017, special edition for Embedded World Nürnberg, is published on 1 March 2017. Content and article titles subject to change.

RGBDigit Clock

A colorful 7-segment display for your data



What do you pick when you want to design "something" with a vintage or modern display? A clock of course! This is our first design with 'RGBDigits' multi-color 7-segment display devices. With a BME280 breakout board (BoB) attached, the project will also cheerfully display temperature, humidity and air pressure.

By **Coen de Bruijn** (The Netherlands)

The clock is controlled by an ESP12 module, which enables synchronizing the clock with an Internet time server, change the clock settings from any mobile device or computer in the network, or transmit sensor data via Wi-Fi.

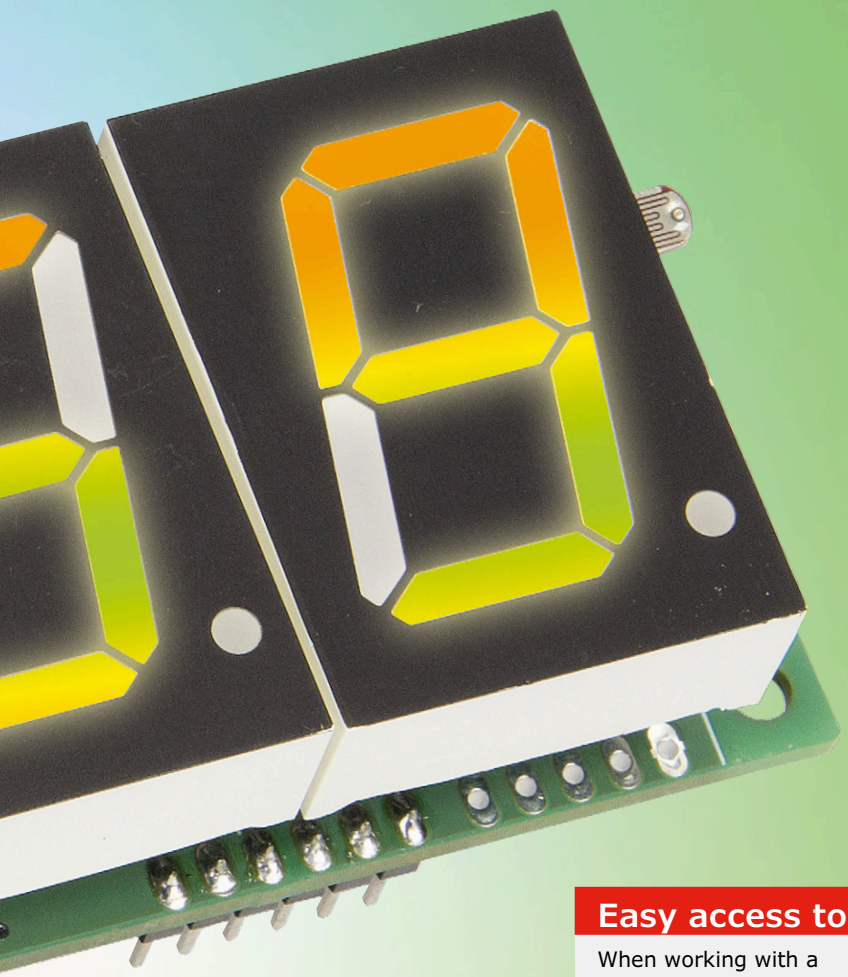
What are they?

RGBDigits are manufactured by a Dutch company with the same name. Every display contains eight 5050 RGB NeoPixel LEDs (seven segments plus decimal point) with integrated driver chips, which allow the user to control color and brightness of every segment/DP individually via a 3-wire bus (VCC, GND an DATA). Up to 10 displays can be daisy chained via their DATA IN and DATA OUT pins.

Each primary color LED of a NeoPixel can be set to 256 levels of brightness, resulting in a whopping 256 x 256 x 256 = 16,777,216 colors for every segment/DP.

How the circuit works

Figure 1 depicts the full circuit diagram for this project. The clock is 5-V powered via micro-USB connector K1. The power supply is protected by 2-A PTC resettable fuse F1, with Schottky diode D1 acting as a reverse polarity protection. IC1 is the main 3.3-V voltage regulator, and IC6 is the 3.3-V supply for the Qtouch touch sensors IC4 and IC5 (note separated supply to prevent interference). The two touch sensors Button0 and Button1 are used to control the display



PROJECT INFORMATION

Test & Measurement

RGB LED Clock Sensors

entry level
➡ intermediate level
expert level

3 hours approx.

Computer with Arduino IDE

€100 / $105 / £85 approx.

modes of the clock. S1 and S2 are only used for resetting the clock and for flashing the firmware. IC2 serves as an unidirectional level shifter between the 3.3 V at the ESP-12E side and the V+ (approximately 4.5 V) power supply of the displays. EEPROM IC3 is used to retain the clock settings after power off, while LDR R1 is used to dim the displays in the dark.

A 3.3-V FTDIstyle cable (or another 3.3-V USB UART) can be connected to K2 to flash the ESP-12E module; it can also be used for debugging applications.

K3 makes the 3.3-V power and three general purpose I/O pins of the ESP12E accessible for your own developments and/or future expansions.

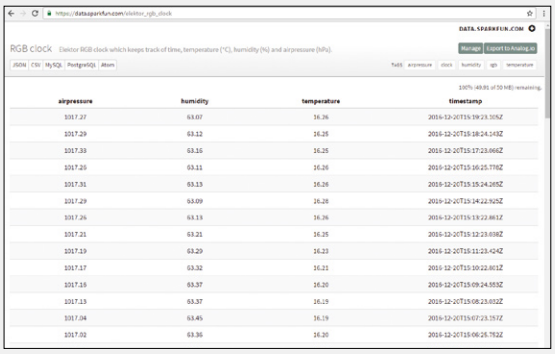
A BME280 BoB (Store #160109-2, **Figure 2**) can be connected to K4. The clock will run without it, but of course there will be no sensor data available. An onboard BME280 was no option — heat from the displays will affect the data.

Flashing the sketch

The ESP12E Wi-Fi module (which contains an ESP8266) is supported by the Arduino IDE. So in order to flash the

Easy access to your sensor data

When working with a project like this, or with any IoT-ish project for that matter, it is convenient at times to have access to a data storage server. The techies at Sparkfun recognized this and set up a service that is free to use for everyone. Accessible through *data.sparkfun.com* is a server that will store (and share!) your data in a convenient way. It works like a kind of database and allows you to store and view your data (and that of others) in a table (see **screenshot**).





First, create a datastream by clicking *create*. Fill in the fields and you'll receive a public key, a private key and a delete key. These keys are used to send data to the stream or to delete data. Our clock will need the public as well as the private key to send data to the Sparkfun data server.

From here you can export your data with the click of a button (top right) to *analog.io*, an online service for displaying (IoT) data. At analog.io you get to see a nice graph of your data and/or measurements.

software into the module, first select the ESP8266 (NodeMCU) in the Arduino IDE. Instructions on how to add the ESP8266 (NodeMCU) can be found at [1]. Follow the steps described carefully before you proceed.

Download the software archive for this project (160100-11.zip) from our website [2]. Unpack the zip file into an Arduino sketch folder on your hard drive, while ensuring subfolder DATA (containing the ESP12's webpage) is preserved in this folder.

Now follow these steps:

- connect a 3V3 (!) FTDI cable between your computer and K2 of the RGB Clock;
- connect a 5-V power supply to micro USB connector K1;

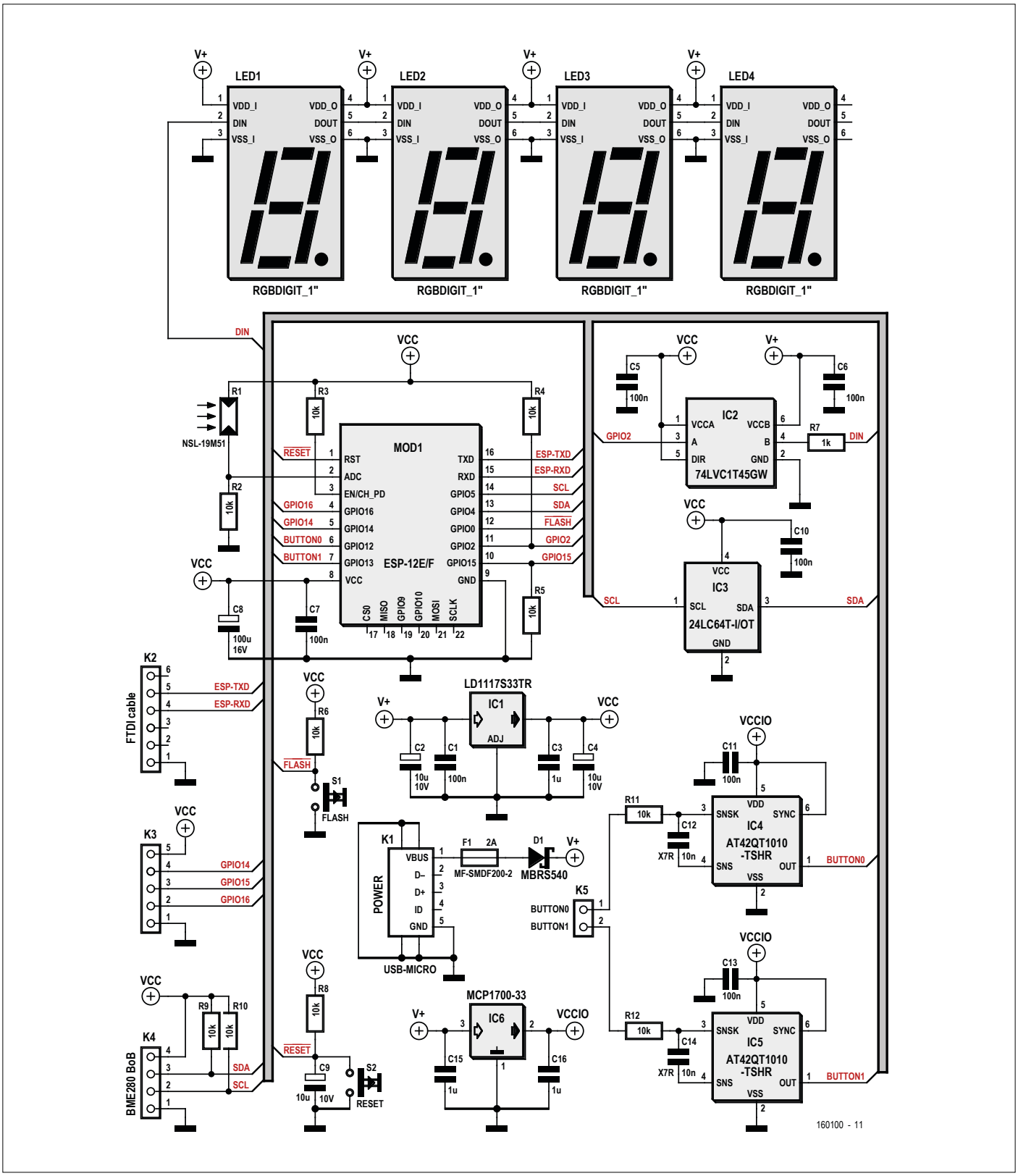


Figure 1. A fair number of connections need to be made. To keep the schematic readable a lot of those connections are bundled in a 'bus'.

- open the Arduino IDE;
- in the menu Tools → Board, select Board NodeMCU 1.0 (ESP-12E Module);
- in the menu Tools → Port, select the COM port associated with the FTDI cable.
- in the menu File → Open, select sketch 'esp12_rgb_clock.INO', included in the download from our website;
- on the RGB cClock board, press and hold reset button S2, press and hold flash button S1, release S2 and then release S1 to enable flashing of the firmware;
- compile and upload the sketch (CTRL+U).

Note that if the compiler exits with an error, this is probably caused by one or more missing Arduino libraries. The compiler will report the name of the first missing library, but you should also check whether all libraries used in the INCLUDE directives in the 'esp8266_rgb_clock' sketch are installed in your Arduino IDE before you start the compiler.

Uploading the clock's webpages

All settings of the clock can be configured via Wi-Fi. The initial connection should be made via the ESP8266's Access Point (AP). Connect your smartphone/tablet/laptop (or PC, if it has a direct Wi-Fi connection) to the open network called 'RGB Clock' that will appear in the list of available Wi-Fi networks after flashing the clock's firmware. However, the clock's webpages must first be uploaded to the ESP8266 before you will be able to get easy access to the settings of the clock. On [3] you'll find instructions how to do this under 'Uploading files to the file system':

- download the tool "Arduino ESP8266 filesystem uploader" from [4];
- unpack the tool into the tools directory (the path will be similar to <home_dir>/Arduino/tools/ESP8266FS/tool/esp8266fs.jar);
- restart the Arduino IDE;
- re-open the "esp12_rgb_clock.INO" sketch (if it didn't open automatically).
- make sure you have selected the correct board and COM port and closed the Serial Monitor;
- press and hold reset button S2, then press and hold flash button

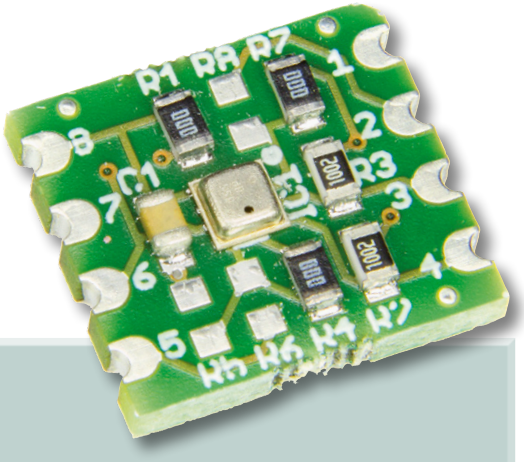


Figure 2. Our BME280 Break-out Board adds temperature, pressure and humidity sensors to the RGBDigit Clock.

COMPONENT LIST

Resistors

R1 = LDR NSL-19M51*
R2,R3,R4,R5,R6,R8,R9,R10,R11,R12 = 10kΩ
R7 = 1kΩ

Capacitors

C1,C5,C6,C7,C10,C11,C13 = 100nF, 50V, X7R, 0805
C2,C4,C9 = 10µF, 10V, tantalum, 1206
C3,C15,C16 = 1µF, 50 V, X5R, 0805
C8 = 100µF, 16V, 2312
C12,C14 = 10nF, 50V, X7R, 0805

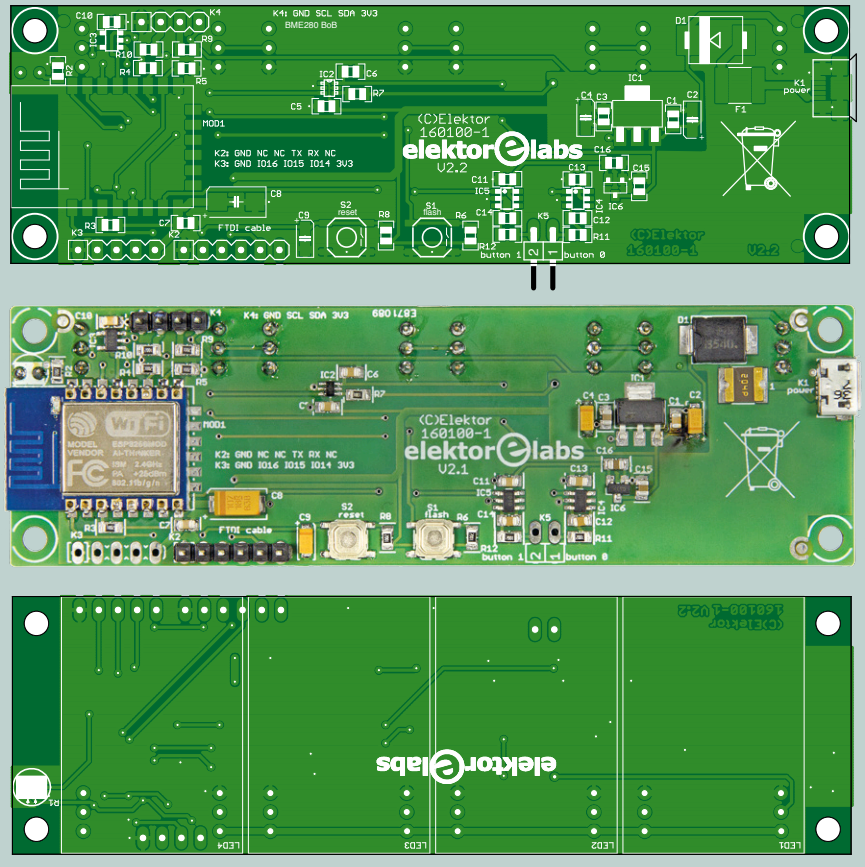
Semiconductors

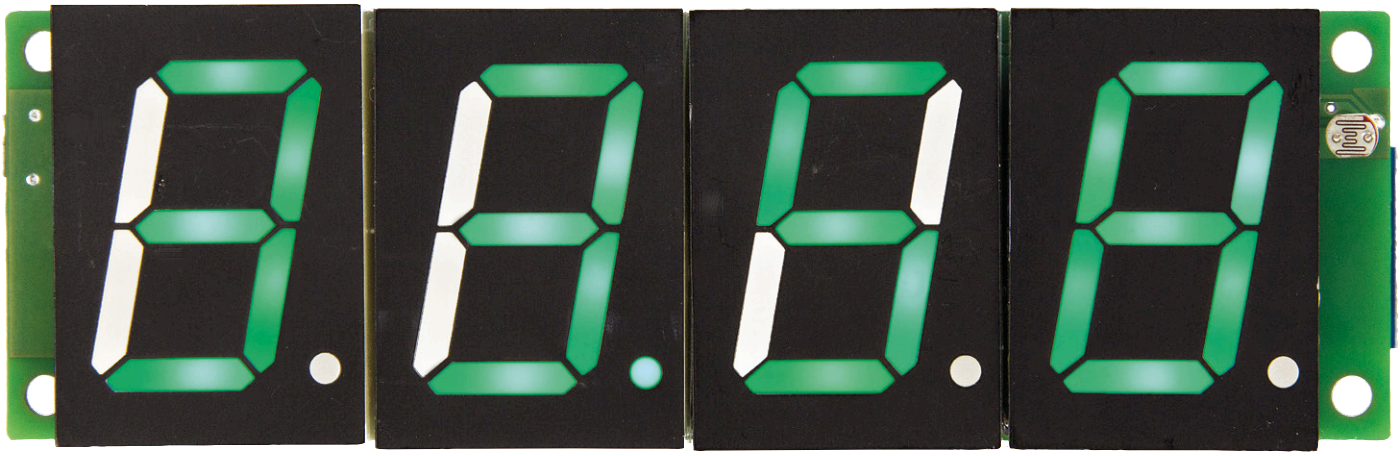
D1 = MBRS540
LED1,LED2,LED3,LED4 = 7-segment RGB display*
IC1 = LD1117S33TR
IC2 = 74LVC1T45GW
IC3 = I2C EEPROM 8K × 8 bit, type 24LC64T-I/OT
IC4,IC5 = Qtouch Touch Sensor type AT42QT1010-TSHR
IC6 = MCP1700T-3302E/TT

Miscellaneous

F1 = 2A PTC resettable fuse, type MF-SMDF200-2 (Bourns)
K1 = Micro-USB type B receptacle, underside mount
K2 = 6-pin SIL pinheader, 0.1" pitch, right angled
K3 = 5-pin SIL pinheader, 0.1" pitch, right angled
K4 = 4-pin SIL pinheader, 0.1" pitch, straight
K5 = 2-pin SIL pinheader, 0.1" pitch, right angled
MOD1 = ESP8266-12F
S1 = Tactile switch
PCB # 160100-1 v2.2

*mount at bottom side of PCB





S1, now release S2 and then finally release S1 to enable flashing of the webpages;
- select Tools → ESP8266 Sketch Data Upload. This should start uploading the files stored in the "data" folder that is included in the download from our website into the ESP8266 flash file system. When done, the IDE status bar will display the message 'SPIFFS Image Uploaded'.

If you fancy redesigning the clock's web page, that's also within reach. See the inset **How to create/edit a webpage for the ESP12.**

Now the fun starts!

On your PC, tablet or Smartphone, connect to the 'RGB Clock' Wi-Fi network, open a browser and connect to 192.168.4.1:81. In the menu (top left corner) select 'Wi-Fi settings' and enter the SSID and password of your wireless network. The clock will use this network to connect to the time server (NTP) to synchronize its time. In the sketch the server 'time.nist.gov' is defined as the NTP server for our clock. The clock will also send its sensor data to a server at Sparkfun (see inset **Easy access to your sensor data**).

A few remarks apply:

- the ESP12's access point will still be active after you have changed the Wi-Fi settings, so you can always modify them (and other settings) with your mobile device connected to the "RGB Clock" network;
- firewalls — like the one used at the Elektor offices — may block the NTP-protocol of an Internet time server, making automatic synchronization of the RGB clock impossible. We used a smartphone configured as a Wi-Fi hotspot as a quick bypass for retrieval of the time data for the clock. Once it is synchronized it can run without NTP data. Alternatively you can use the manual settings on the clock's webpage to set the correct time;
- if you would like to connect the clock to an unprotected network (i.e. no password), check the 'Unprotected Wi-Fi network' check mark in the Wi-Fi settings of the clock.

Every device with access to your Wi-Fi router can now access the clock to change its settings. Simply type 'clock.local:81' in the address bar of a browser to get access to the clock's webpages.

(160100)

How to create/edit a webpage for the ESP12

We built the internal website for the ESP12 using the HTML-Kit292 (www.htmlkit.com/download). This free program allows you to create and design your own website. Out web page comprises three files: *index.html*, *style.css* and *script.js*. Index.html is the main file and is programmed in HTML. It determines the structure of the website. In style.css we define how our page looks. Script.js determines what the page does.

The ESP12 webpage is built from several elements. Using these elements the layout and functionality are defined in style.css and script.js.

FROM THE STORE

- **160100-1** Bare PCB
- **160100-91** ready assembled module
- **160109-91** Ready built module BME280 BoB (optional)
- **SKU 17789** RGBDigit 7-segment display

Web Links

[1] www.arduinesp.com/getting-started

[2] www.elektormagazine.com/160100

[3] https://github.com/esp8266/Arduino/blob/master/doc/filesystem.md

[4] https://github.com/esp8266/arduino-esp8266fs-plugin/releases/download/0.2.0/ESP8266FS-0.2.0.zip

The Elektor Community

LEARN DESIGN SHARE

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

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

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

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

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

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

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

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

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

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

Become a member today!

GREEN
€5.67 per month
£4.08 / US $6.25

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

www.elektor.com/green

GOLD
€7.58 per month
£5.50 / US $8.42

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

www.elektor.com/gold

FREE

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

www.elektor.com/newsletter

facebook.com/ElektorIM

twitter.com/Elektor

Starter Kits & Development Boards

A brief round-up

In these very pages exactly a year ago we took a look at some new microcontrollers and development boards. Today we bring our (inevitably incomplete) survey up to date, looking at some new starter kits and some interesting new development platforms.

By **Viacheslav Gromov** (Germany), readers@gromov.de

Last year [1] we took a look at some mostly low-cost starter kits and development boards. These platforms have by and large survived the test of time, as the designs tend to be universal and produced in large quantities. That is not to say, however, that the more lavishly-equipped boards do not make a good starting point for development: quite the reverse, in fact, as it is often easier to start with a more complex special-purpose microcontroller board that may perhaps also be more orientated towards professional use. Likewise, it can be worth looking to the future when choosing a development environment: overcoming the bigger hurdles to learning presented by a more complex system may prove worthwhile in the long run. And of course it is easier to switch from a well-equipped microcontroller to a more spartan sibling than the other way around.

This time around, then, our tour of the world of microcontroller development boards will take in some new and more exotic options, including those that differentiate themselves by being aimed at specific applications. We will examine the boards in order of price: they range from a little over ten dollars to hundreds of dollars. We will not cover the standard IDEs that are used with each microcontroller family, but will focus on new software features and newly-introduced tools. Something for everyone, we hope! Before we start, we should remark that this survey is not intended to cover the huge number of IoT (Internet of things) development boards that include wireless interfaces. They will be the subject of their own article in our next issue.

Matchbox-sized

The XMC 2Go board we looked at last time, with its well-equipped XMC1100 (ARM Cortex-M0+, 48 MHz, 64 Kbyte flash), has been updated, with three new versions (see **Figure 1**): the **Current Sensor 2Go** with a TLI4970 SPI-bus current sensor, at about US$30 [2]; the **3D Magnetic Sensor 2Go**, with a type TLV493D-A1B6 three-dimensional magnetic field sensor, again about US$30 [2]; and the **H-Bridge Kit 2Go**, with an SPI-bus H-bridge driver IC type IFX9201SG, at around US$15 [3]. Each of the different sensors allows you to accomplish different tasks.

The TLI4970 uses two integrated Hall-effect sensors to measure the current flowing through an internal conductor. It employs a differential measurement technique to minimize the influence of external interference on readings. The results (12 bits, representing AC or DC currents of up to ±50 A) are digitized internally and then processed using a DSP implementing a configurable filter before being passed to the SPI port, resulting in an output signal that represents the current being measured as faithfully as possible.

The TLV493D-A1B6 measures the strength of the incident magnetic field up to ±130 mT on three axes, again to a resolution of 12 bits. The device also includes a temperature sensor to improve the stability of measurements, and the I²C interface to the microcontroller allows

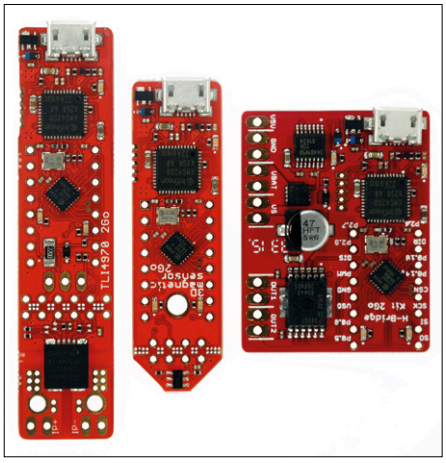


Figure 1. The Current Sensor 2Go, 3D Magnetic Sensor 2Go and the H-Bridge Kit 2Go.

it to fit in a tiny 6 pin TSOP package.

The IFX9201SG differs from the other ICs used in this series of boards in that it does not include a sensor. Instead, it offers a standard H-bridge circuit for driving small DC motors and other inductive loads at up to 6 A. The bridge is controlled using a direction pin and a PWM input (at up to 20 kHz), and an SPI port is provided for monitoring.

Common to all three boards, besides being based on the same microcontroller, is that the most important pins on the processor are brought out and that two user LEDs are provided. A valuable

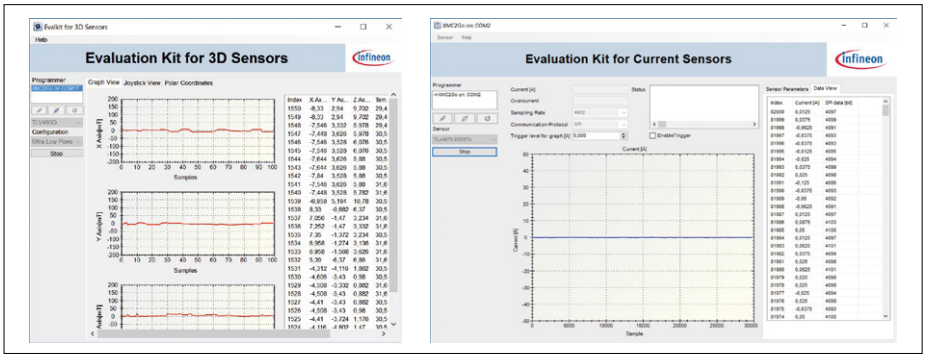


Figure 2. Windows-based software tools are available for both the 2Go sensors, allowing you to try them out before you embark on the programming proper.

feature of the two sensor boards is that the sensor part can be broken away from the main board; this is not possible on the H-bridge board. Instead, it sports a separate power supply connection, which facilitates building the board directly into your own designs. The magnetic field sensor can also be extended with a joystick or a rotary control, which allows you to turn the sensor into a stepless rotary encoder. All the boards in the series come with their own easy-to-use libraries and example projects for use in the DAVE development environment, and there are also additional software tools available to help make the most of the sensors (see **Figure 2**).

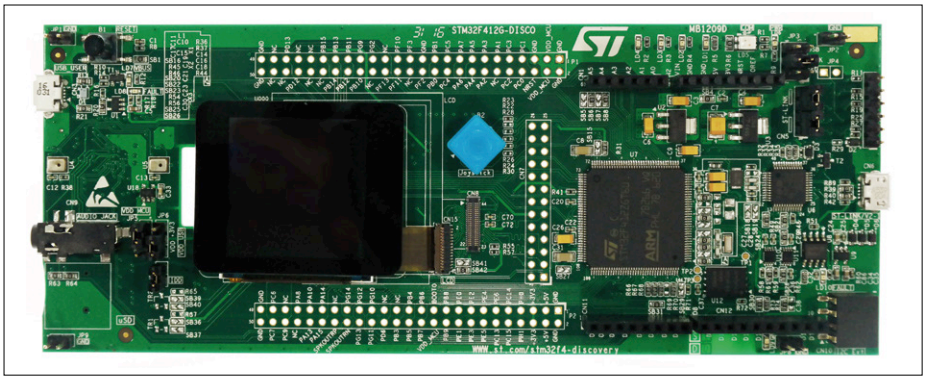


Figure 3. The elongated design of the STM32F412G Discovery board means that the Arduino and other connectors are not fouled when a shield is fitted.

Touch display included, US$25

ST Microelectronics has already gained a reputation for very low-cost but nevertheless rather well-fitted-out development boards, but now they have surpassed themselves. The **STM32F412G Discovery Kit** (**Figure 3**) [4] features the STM32F412ZG, a new member of ST's STM32F4 family of 32-bit ARM Cortex-M4-based microcontrollers running at 100 MHz and offering 1 Mbyte of flash. Highlights of the board include a 240x240 TFT touch display, an audio output with codec and a class D amplifier, two digital MEMS microphones, a 128 Mbit external quad-SPI flash memory, a microSD slot and a micro USB interface.

Of course, a large number of the microcontroller's GPIO (general-purpose input/output) pins are brought out in the form of headers following the Arduino layout, which until recently was largely only a feature of the Nucleo series of boards. Other basic features of the Discovery boards are retained: four user LEDs, a joystick (replacing user buttons) and the familiar ST-Link/V2-1 debugging interface. As soon as the board is connected to a USB power supply, the demonstration program will start up and give a good idea of the incredibly powerful possibilities of this unimposing little device. The demo includes a video and audio player with display, audio output and memory card handling. And all that for just a couple of tens of dollars: about the same amount as one would have to fork out for an Arduino Uno.

Programming and debugging on this board is best done using the Keil (**Figure 4**), IAR or openSTM32 environments, supported by the familiar STM32CubeMX initialization code generator and STM32Cube software libraries and example programs.

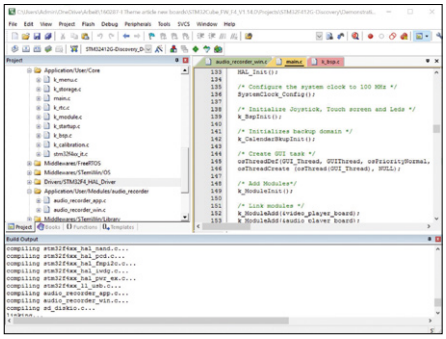


Figure 4. The STM32CubeF4 libraries and demonstration programs can be used with the free version of Keil's µVision development environment as long as you are careful not to overstep its code size limit.

ATtiny catches up

A few years ago the ATtiny series of microcontrollers were decidedly less powerful and less well equipped than their ATmega siblings. However, at Electronica 2016 a new series of ATtiny8xx devices was announced, with 14 to 24 pins and 4 Kbyte or 8 Kbyte of flash memory, running at up to 20 MHz. A new Xplained Mini development board was also announced. The ATmegas and ARM-based microcontrollers no longer look down on the ATtinys, which now to a large extent share common internal structures and peripherals with the larger devices. Particular examples of this include the event system and the core independent peripherals (CIPs), and the peripheral touch controller (PTC), which operates independently of the CPU without consuming processing power.

A more recent product is the **ATtiny817 Xplained Pro Board** (for around US$40), which uses the same microcontroller (see **Figure 5**) [5]. This is a clear indication of the way the ATtiny family is moving up in the world, as the board and processor are supported by the Atmel START online environment [6] (see **Figure 6**). This environment allows the most important libraries and peripheral settings to be configured for a new project from within a browser, before firing up the Atmel Studio IDE.

Like practically all of the Xplained Pro boards, this example is equipped with two touch buttons, headers, an embedded debugger (EDBG) and a configurable circuit to allow the device's current consumption to be measured. The big advantage of the Pro board compared to the Mini board is the ability to use Atmel's Pro expansion boards, which run the whole gamut from a temperature sensor to an OLED display.

Get in touch with the analog world

A brand-new member of Cypress' 32-bit ARM Cortex-M0+ family of PSoCs is

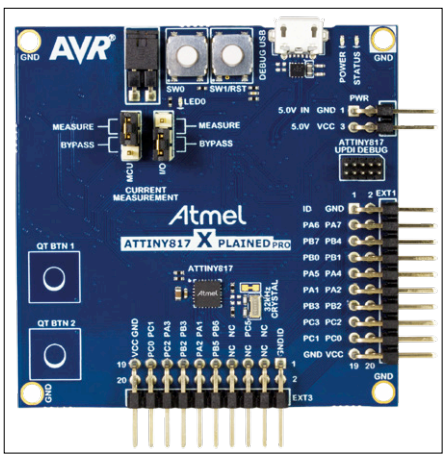


Figure 5. The Attiny817 Xplained Pro board is around the same size as the Xplained Mini board, but is more flexible (source: Microchip).

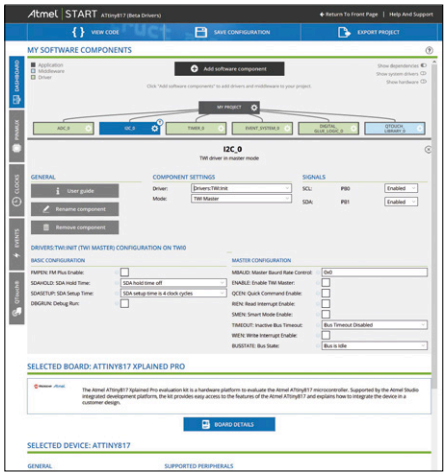


Figure 6. A screenshot of the Atmel START online environment. Each of the elements listed towards the top of the screen can be configured in detail before importing the customized project into Atmel Studio 7.

the PSoC analog coprocessor (48 MHz, 32 Kbyte flash, DMA features), which is specially designed for interfacing between the analog world of sensors and the digital world of its processor core (called the 'host processor' in the documentation), perhaps as a component in a larger-scale project. The coprocessor includes a huge range of analog peripherals, including two ADCs (12 bit and 10 bit), two seven-bit IDACs to generate currents, often required in touch-based technologies, four operational amplifiers, two comparators, a CapSense touch interface and a universal analog block (UAB) which can be configured as a filter, 14-bit ADC or 13-bit DAC. A number of analog multiplexers is provided to allow the various analog peripherals to be connected to device pins as required. Despite the focus on analog interfacing, the device also contains a useful set of digital peripherals including timers and standard digital interfaces. Of particular note is the highly flexible integrated LCD driver, offering sleep modes down to 2.5 µA. The microcontroller can operate from supply voltages as low as 1.71 V [7].

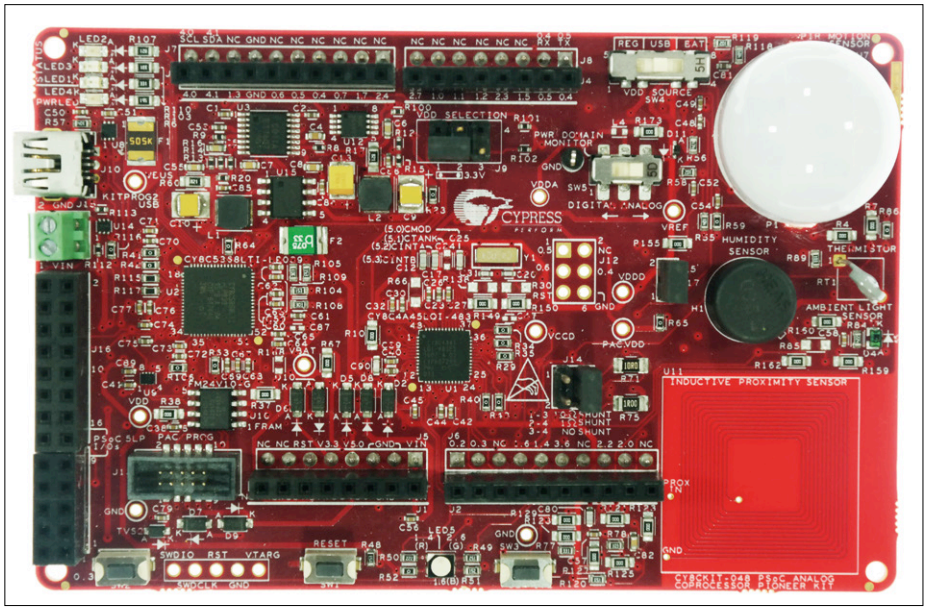


Figure 7. Contents of the CY8CKIT-048: besides the board and cable you also get headers to let you build your own expansion boards and a metal disk to test the inductive proximity sensor.

The **CY8CKIT-048 PSoC Analog Coprocessor Pioneer Kit** (around US$50) [8] has been developed to show off the possibilities opened up by this flexible PSoC technology: see **Figure 7**. The compact board includes five analog sensors: a PIR motion detector, temperature, humidity and light sensors, and a receiver coil for inductive proximity sensing laid out directly on the printed circuit board. Inductive sensing is all the rage, as it offers many advantages over capacitive technology. The sensor is designed in such a way that it can detect the proximity of a (supplied) metal disk, and it is so sensitive to tiny movements of the disk that it would in principle be possible to use it to construct a kind of touch sensor. In many cases the board offers various circuit possibilities for the sensors listed above, selected using switches or jumpers. Many connections are brought out to headers (pins on the bottom of the board, sockets on the top), and using these the board can be used as an Arduino shield or used to control an additional Arduino shield mounted on top of it. There are also basic digital interfaces: a user RGB LED and two user pushbuttons. A KitProg2 programmer and debugger acts as a bridge between the mini USB socket and the processor itself. Using this allows the board to be programmed and debugged directly from PSoC Creator with no additional hardware (**Figure 8**).

A grand entrance?

Last year Renesas announced a small demonstration board based on the 16-bit RL78 family of microcontrollers, and now the manufacturer is marketing its Synergy platform, based on 32-bit ARM Cortex-M cores. Normally one would expect to have to dig deep in one's pockets for a Renesas starter kit, but the **SK-S7G2**

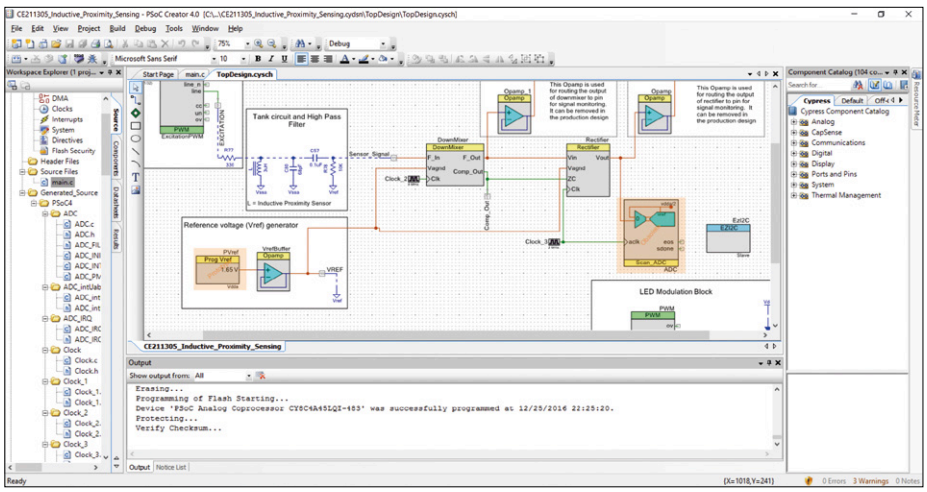


Figure 8. The internal circuitry of the peripheral module which is responsible for processing the output of the inductive sensor in PSoC Creator: this saves the developer from writing a lot of code.

Starter Kit, which employs a Synergy S7 processor (240 MHz, 4 Mbyte code flash, DMA features) marks a departure from the past (**Figure 9**) [9].

The kit is designed by the manufacturer as a platform to demonstrate a wide range of example applications. At around US$80 it is reasonably affordable and is even given away for free at trade shows. The application areas targeted by the S7 (which is based on a Cortex-M4 core) are human-machine interfaces (HMIs) and connectivity: the palette of features includes a graphical LCD controller and a JPEG codec as well as Ethernet capability with two integrated MACs. These features mean that the family is well aligned with the core themes of Industry 4.0, IoT and the Smart Home, and this is of course reflected in the design of the demonstration board. As well as myriad pin headers providing connections to the microcontroller's pins and Arduino-compatible header sockets, there is an on-board touch display, several touch controls, and USB, audio and Ethernet interfaces. There are various opportunities for expansion, including a Bluetooth low energy (BLE) add-on module. And finally there are three user LEDs and two user pushbuttons, along with an integrated J-Link debugger: the costly E1 emulator is no longer required with this kit.

Highly recommended is the brand-new e-book that has been produced to accompany the Synergy family and the kit: it is available at [10].

The standard development environment for the Synergy platform is Renesas' E²Studio, familiar from the RL78 family. An addition to the environment specifically for the Synergy platform is the Gallery (**Figure 10**) [11]. This includes all the most up-to-date libraries, additional software tools (for example for generating display graphics or for configuring security features) and of course plenty of documentation and support.

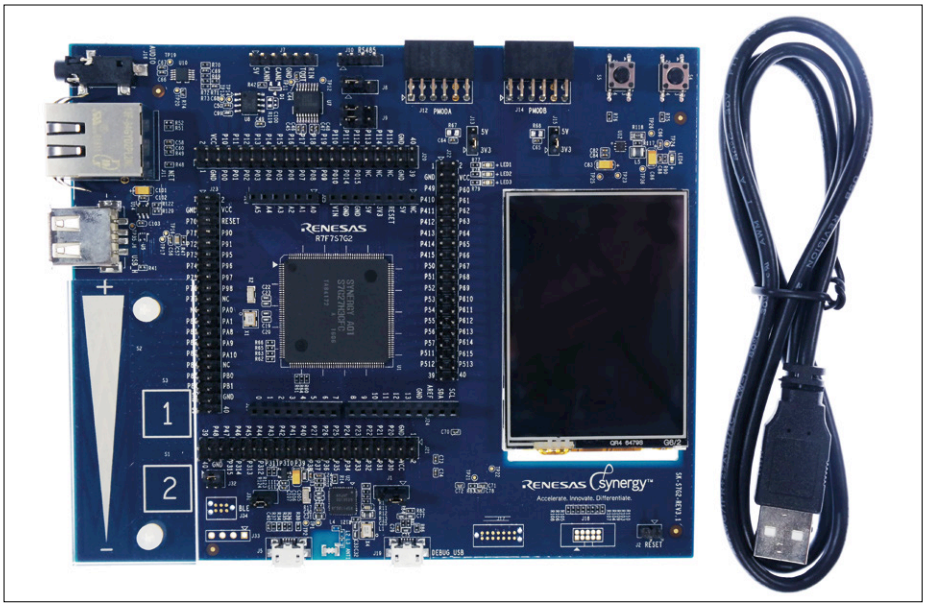


Figure 9. The SK-S7G2 comprises just the board and a USB cable, but is very low-cost. Nevertheless, the kit is more than adequate for debugging your own programs.

Wireless two ways

NXP has recently introduced the KW41Z (48 MHz, 512 Kbyte flash, DMA features) [12], a 32-bit microcontroller based on an ARM Cortex-M0+ core aimed squarely at wireless applications. Roughly speaking it combines the features of the KW21Z and the KW31Z in the 2.4 GHz band, and is able to communicate using both the Bluetooth low energy (BLE) 4.2 and the IEEE 802.15.4 protocols, as well as other protocols such as Thread. The device is therefore called a 'dual mode' microcontroller. As usual for NXP (and formerly for Freescale) a 'Freedom Kit', the **FRDM-KW41Z**, is available (see **Figure 11**). The kit, which costs around US$150, consists of two identical boards to allow convenient testing of communications.

The boards include an OpenSDA debugger for direct use with Kinetis studio. There is a printed antenna on the board, and a connector is also provided for an external antenna. Other features include a coin cell holder, two user LEDs (one RGB), two user buttons, an analog temperature sensor, and an FXOS8700CQ

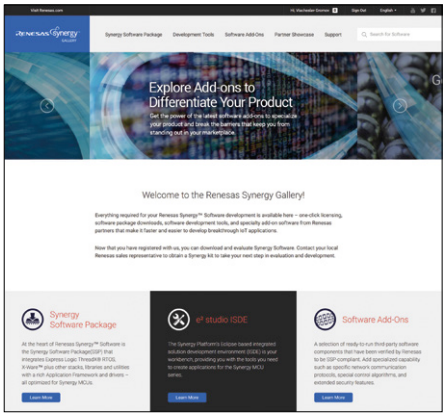


Figure 10. The main content categories are listed towards the bottom of this screenshot of the Synergy Gallery.



Figure 11. The kit includes two boards and two USB cables.

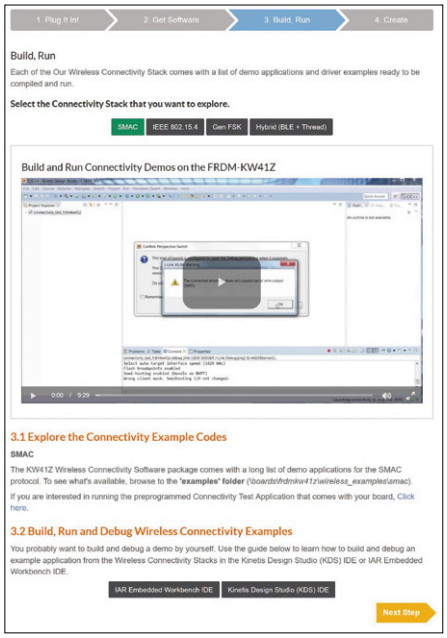


Figure 12. Once everything is installed and set up, you can follow one of the 'getting started' videos to get an example program running.

three-axis I²C accelerometer and magnetometer.

Also as is customary for Freescale/NXP, the documentation is faultless. Particularly worthwhile is the interactive 'getting started' guide on the product web page (**Figure 12**) [13], where a series of videos gives step-by-step instructions for installing the development environment and KW41Z software, trying out the example programs, and setting up your own projects.

Analog goes digital

Alongside the familiar microcontroller manufacturers such as Microchip, ST, NXP et al., it is easy to forget that there are companies better known in the analog domain who also produce microcontrollers, along with high-quality development kits and application examples. An example is Analog Devices, one of whose most recent offerings is the ADuCM3029 (26 MHz, 256 Kbyte flash). As is usual for devices based on an ARM Cortex-M3 core, this microcontroller is fully-featured in all respects [14]. It particularly shines when it comes to power-saving modes: current consumption can be reduced to below 60 nA.

For only around US$200 you can have your own **EVAL-ADuCM3029 EZ-KIT** (**Figure 13**), which basically consists of a microcontroller board with two user buttons, three user LEDs, a buzzer, temperature and acceleration sensors (ADXL363 and ADT7420 respectively), a mini USB port, two battery holders, Arduino-compatible header sockets and an external J-Link Lite debugger. The board can be programmed using the familiar development environments from Keil and IAR, or the manufacturer's own (but sadly not free) CrossCore Embedded Studio [15]. There is plenty of example software and documentation available. A point that weighs considerably in favor of choosing this microcontroller family for your next project is the experience of the manufacturer in the analog domain. Analog Devices also makes sensors and other microcontroller peripheral components, and provides libraries and other support for them. A wide range of sensor shields is available for initial experiments.

Another example of Analog Devices' forays into the digital world is the **ADZS-BF707-BLIP2** board [16] and the technologies behind it. This board, which costs around US$200, offers a wide range of interfaces and an ADZS-BF707 32-bit microcontroller clocked at 400 MHz. The two on-board camera sensors allow you to experiment with reliably recognizing objects, people or even faces. To develop with the board you will need the additional ICE-1000 emulator as well as CrossCore Embedded Studio (for which a 120-day license is included with the board). Nevertheless, the system works well and gives a good first impression (.

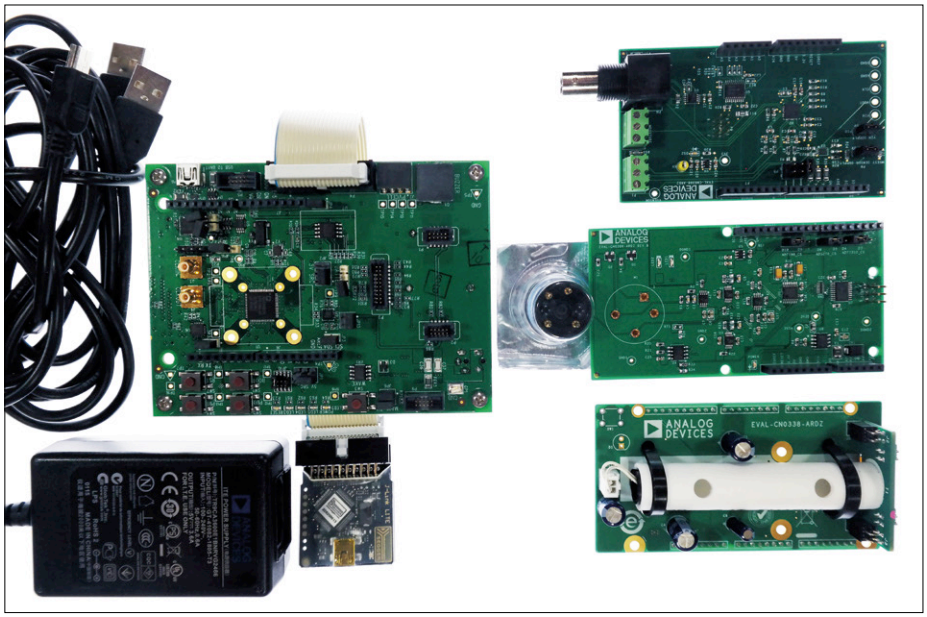


Figure 13. The main elements of this kit are the debugger, the board itself, the USB cable and the mains power supply. On the right is a small selection of sensor shields (pH, temperature and humidity, and gas sensors).

The many faces of energy harvesting

Semiconductor manufacturer Linear Technology has recently partnered with Würth Elektronik, experts in the domain of passive components and connectors, to create the **Gleanergy Demonstration Kit** (the 'Gleanergy' moniker is a portmanteau of 'glean' and 'energy'), which costs around US$400: see **Figure 14**. The kit includes two printed circuit boards and a USB transceiver [17]. The first board, with part code DC2344A [18], deals with the analog side of energy harvesting. It includes a primary battery, a secondary (rechargeable) battery, a supercapacitor, a Peltier module, two solar cells and all the necessary support circuitry. There is a host of jumpers to allow board options to be configured as well as a large number of test points and other connections. At the heart of the board are four Linear Technology regulator ICs. Two of these are general-purpose devices suitable for use with solar cells, piezo transducers or thermoelectric generators. The main difference between them is the type of battery they support: they LTC3330 works in conjunction with the primary battery, whereas the LTC3331 works with the rechargeable battery. In addition, there is an LTC3107 which can work with the Peltier module and the primary battery, and an LTC3106 which supports the solar cell in conjunction with either battery.

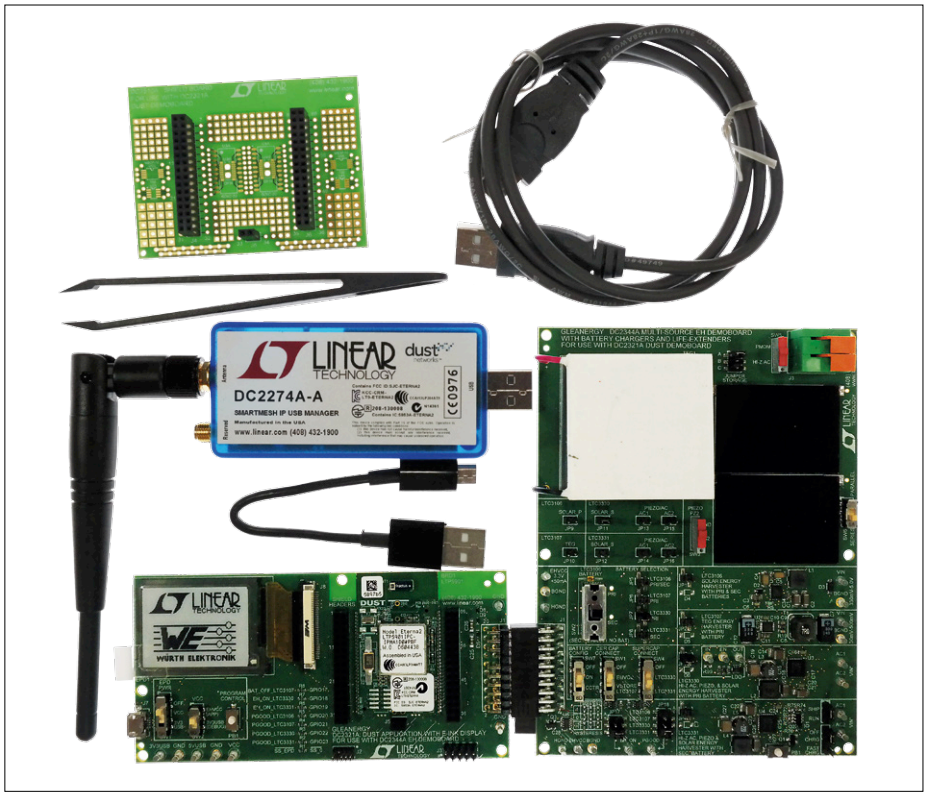


Figure 14. What you get for your money: two printed circuit boards, a high-quality cable, a USB transceiver with antenna, an expansion kit and even a pair of tweezers.

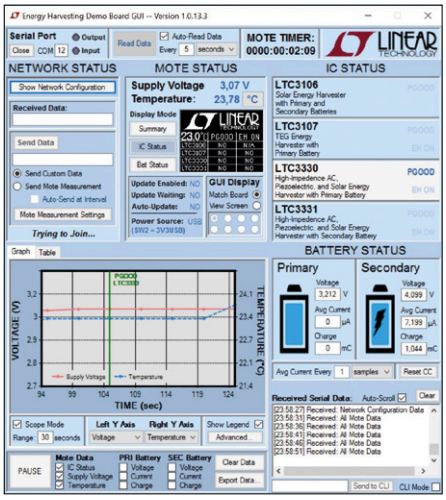


Figure 15. The QuikEval tool is designed to help you develop your own energy harvesting applications.

Jumpers and switches on the front of the board and resistors fitted to the back of the board allow various configurations to be tested, and the individual regulators can be connected in series or parallel, or taken out of circuit entirely. The combinations are too numerous to explore here, but the accompanying documentation describes the board and all its options in scrupulous detail.

The second board, with part code DC2321A [19], can be plugged directly into the side of the DC2344A. The main components on the board are the LTP5901 'Dust mote' module and an e-ink display. The LTP5901, which is based around an ARM Cortex-M3 core, is a member of Linear Technology's SmartMesh IP Wireless family of devices. These are designed around the IEEE 802.15.4e standard, which also forms the basis of ZigBee networks. The 'Dust' technology allows a range of different types of structured radio network to be created, for example for IoT applications. Within the Gleanergy platform the job of the LTP5901 module (itself designed for low-power operation) is to monitor the energy harvesting process, and then send the results out over the micro USB port and show them on the e-ink display. A demonstration tool is provided that allows you to examine all the most important settings of the regulators, with the aim of helping you translate the prototype into your own energy-harvesting design (**Figure 15**). A blank shield, type DC2510A, is provided for you to add your own circuitry. The whole thing can of course be used wirelessly with the help of the DC2274A-A SmartMesh USB dongle, and a software development kit (SDK) is also provided. Overall the demonstration kit is a worthwhile purchase if you want to shorten the development time for your own energy harvesting or wireless product. ⏮

(160287)

Internet Links

[1] www.elektormagazine.com/150687

[2] www.infineon.com/cms/en/product/promopages/sensors-2go/

[3] www.infineon.com/cms/en/product/evaluation-boards/H-BRIDGE+KIT+2GO/productType html?productType=5546d46250cc1fdf015124e027823420

[4] www.st.com/en/evaluation-tools/32f412gdiscovery.html

[5] www.atmel.com/tools/atiny817-xpro.aspx

[6] http://start.atmel.com

[7] www.cypress.com/products/psoc-analog-coprocessor

[8] www.cypress.com/documentation/development-kitsboards/cy-8ckit-048-psoc-analog-coprocessor-pioneer-kit

[9] www.renesas.com/en-eu/products/software-tools/boards-and-kits/renesas-starter-kits/renesas-synergy-sk-s7g2.html

[10] www.renesas.com/en-us/products/synergy/book.html

[11] http://synergygallery.renesas.com/

[12] www.nxp.com/products/:KW41Z

[13] www.nxp.com/products/:FRDM-KW41Z

[14] www.analog.com/en/products/processors-dsp/microcontrollers/arm-cortex-m3-processor/aducm3029.html

[15] www.analog.com/en/design-center/processors-and-dsp/evaluation-and-development-software/adswt-cces.html

[16] www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/adzs-bf707-blip2.html

[17] www.linear.com/solutions/5744

[18] www.linear.com/solutions/7717

[19] www.linear.com/solutions/7662

HOMELAB PROJECT

IoT Gateway and Wireless Nodes

Part 1: The hardware



Sometimes you want to have specific functions in a home automation system that are not available with any commercial product. That's why the author decided to develop his own system, which features wireless communication between the various nodes and the central gateway. The gateway uses MQTT to send measurement data to an OpenHAB server, which processes and displays the data. His project has now been copied and extended by other users. In a series of two articles, the author describes the key elements of his system. More detailed information and the corresponding software are available for free on the Internet.

By **Hennie Spaninks** (Netherlands)

Home automation systems are enticing. There's something magical about having your lamps go on and off by themselves, a heating system that knows when you are at home, or receiving a message when someone is at your door. There are quite a few systems currently available, but most of them suffer from the following shortcomings:

- They often work in only one direction – you can send commands, but you never know whether they actually arrive. That's not a big issue when you're at home sitting next to a lamp and reading, but when you are on vacation in a foreign country it's nice to know that everything is working as it should.
- They often can only be operated using a specific app, and each vendor has their own app with protocols that do not work with other systems. As a result, you end up with a whole bunch of apps on your smartphone.
- The nice thing about home automation is that you can program them to send commands in response to sensor readings. However, because most commercially available systems are not compatible with each other, you have to use an external service (such as *IfThisThenThat*) to get a working system.
- Many wireless systems are based on Wi-Fi. In modern residential buildings made from reinforced concrete, the range of Wi-Fi signals is very limited.
- Security is not always optimal, so there is a distinct chance that your neighbors will be able to control your lamps.
- All in all, there are plenty of good reasons to roll up your sleeves and build your own home automation system.

Overview

For the end nodes, which means the sensors and actuators, you ideally want reasonably affordable devices which are also energy efficient. The central controller obviously has to support

standard network protocols, including TCP/IP. That is why the author opted for a two-tier system consisting of end nodes, a gateway and a central control unit. The overall system is shown schematically in **Figure 1**.

- For communication with the end nodes, the system uses wireless duplex links operating at a relatively low frequency. In Europe the available frequency bands for this purpose are 433 MHz and 868 MHz. The 433 MHz band is fairly crowded, so we opted for 868 MHz. Transceivers for this band which can be controlled over an SPI bus are available from HopeRF. We chose the version with the highest transmit power: the RFM69HW. This module has integrated hardware encryption, so security is not an issue.
- The RFM69 does not have a TCP/IP stack, so fixed-size data blocks of 66 bytes are exchanged over the radio link. A gateway is necessary to convert the data from the wireless network to TCP/IP and vice versa.
- To distribute the data from the sources (sensor nodes) to the recipients (for example, a smartphone), we opted for a different standard protocol which runs over TCP/IP: MQTT. It is a messaging protocol designed to send short messages to several recipients in a simple manner. Because MQTT is a standard protocol, data streams from our home automation system can also be used in other MQTT-based systems.
  MQTT requires a server to act as the message center, which is called the broker. A recipient (such as a smartphone) can subscribe to a data stream, after which it receives messages for that stream through a push service. To implement our MQTT broker we use Mosquitto [1], an open source MQTT broker implementation which is available for several platforms (including Raspberry Pi).
- Our home automation system should preferably be compatible with devices from other vendors. This means that it must also be able to communicate using other protocols. For this we opted for a system based on OpenHAB [2]. This open source software can convert messages in various protocols to a universal message stream. For example, you can use it to control Philips Hue lamps.
- OpenHAB has a standard app (for Android and iOS) which communicates with your own OpenHAB system over the Internet using a secure connection. That means you don't have to worry about configuring firewalls or encryption for the link between your smartphone and your home automation system. OpenHAB can be accessed on the local area network through a Web browser.
- OpenHAB has an underlying database which makes it easy to make charts of measured values. Creating rules and scripts to automate tasks is easy in OpenHAB. OpenHAB also has external interfaces for sending alerts via email or the OpenHAB app.
- We use readily available and relatively low-cost components: Arduino as the controller for the wireless nodes, and a Raspberry Pi as the platform for Mosquitto and OpenHAB.

Protocol The protocol for the end nodes

Various functions are defined for the end nodes. They include sensors for temperature and humidity, switches, PIR motion detectors for input, and relays and LC displays for output. End nodes can autonomously transmit measurement data at regular intervals. This measurement data can also be sent on request in response to read commands.

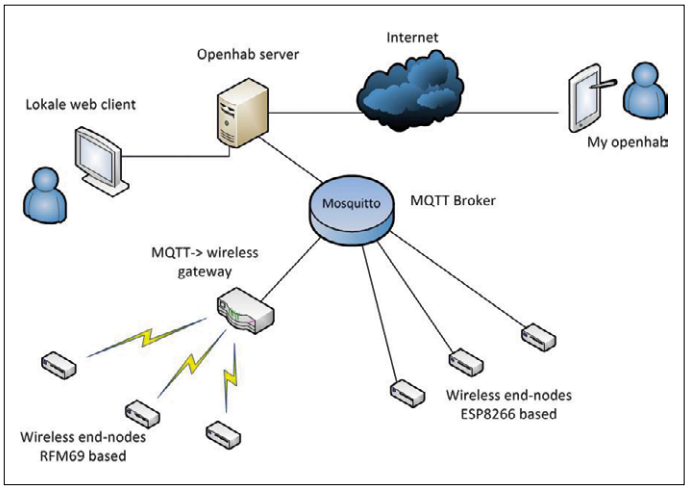


Figure 1. The architecture of the home automation system.

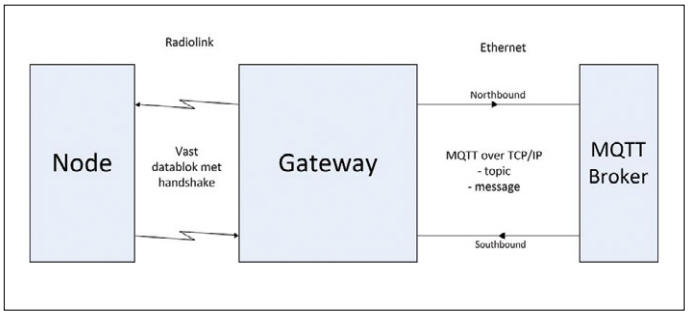


Figure 2. The various data streams passing through the gateway.

Table 1. Device identifiers.

DevID range	Function
0-15	System devices
16-31	Binary output (relay, lamp)
32-39	Integer output (dimmer, PWM)
40–47	Binary input (switch, PIR module)
48-63	Decimal input (temperature, humidity)
64-71	Integer input (keypad, switch)
72	String transfer (LCD display, RFID reader)
73-90	Reserved for future use
90-99	Error messages

The data flow through the gateway is shown in **Figure 2**. End nodes can be programmed for various functions. To ensure that the gateway handles communications with each end node correctly, there is a fixed allocation of functions to devices. Based on the device identifier (DevID), the gateway knows how to handle the incoming data. **Table 1** shows an overview of the DevIDs recognized by the gateway.
Devices 0 to 15 provide system functions. They are described in **Table 2**.
The error messages generated by the gateway are listed in **Table 3**.

Table 2. System functions.			
DevID	**Name**	**RW**	**Function**
00	Uptime	R	Minutes since node start
01	TxInterval	RW	Transmit interval in seconds (0 = no periodic transmission)
02	RSSI	R	Radio signal field strength
03	Version	R	Software version of the end node
04	Voltage	R	Battery voltage
05	ACK	RW	Flag for acknowledging sent commands
06	Toggle	RW	Flag for toggle function of switch on end node
07	Timer	RW	Flag for timer function of pushbutton on end node
08	Btnpress	RW	Flag for sending button press message
09	TXreply	R	Number of repeats necessary on the radio link

Table 3. Error messages.		
Error ID	**Name**	**Description**
90	Link error	The radio link is interrupted.
91	Syntax error	There is a syntax error in the MQTT message.
92	Invalid device	The addressed DevID is not present in the end node.
99	Wakeup	Message sent when the node starts up.

Table 4. MQTT examples.		
Topic	**Message**	**Description**
home/rfm_gw/sb/node02/dev16	ON	Switches on the LED at node 02.
home/rfm_gw/sb/node02/dev16	READ	Queries the status of the LED at node 02.
home/rfm_gw/sb/node03/dev01	300	Sets the transmit interval of node 03 to 5 minutes.
home/rfm_gw/sb/node03/dev01	0	Disables periodic transmission from node 03.
home/rfm_gw/sb/node18/dev48	READ	Reads the temperature from node 18.
home/rfm_gw/sb/node05/dev02	READ	Reads the radio signal field strength at node 05.
home/rfm_gw/sb/node05/dev03	READ	Reads the software version of node 05.

MQTT messages

The MQTT protocol works on the basis of subscribing to topics (see [3]). After a recipient has subscribed to a topic, the broker ensures that all messages related to that topic are sent to the recipient.

The format of the MQTT topic subscription is

`home/rfm_gw/direction/nodeID/deviceID`

Here `direction` specifies the direction of the data flow: “nb”

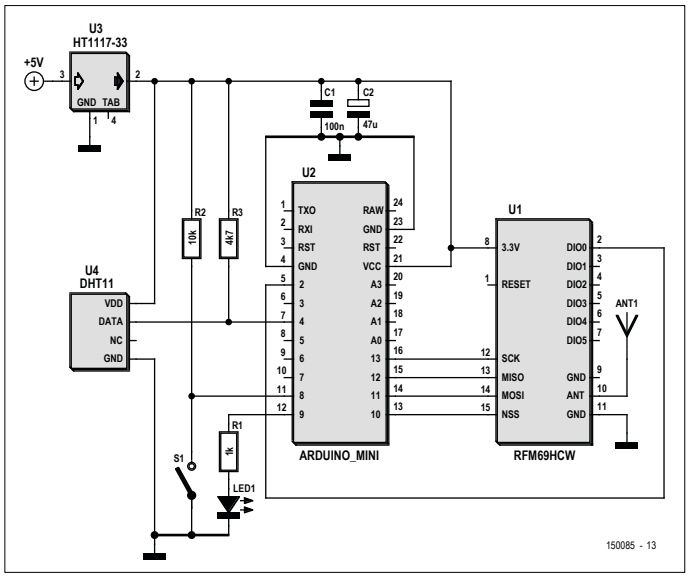


Figure 3. Schematic diagram of an end node.

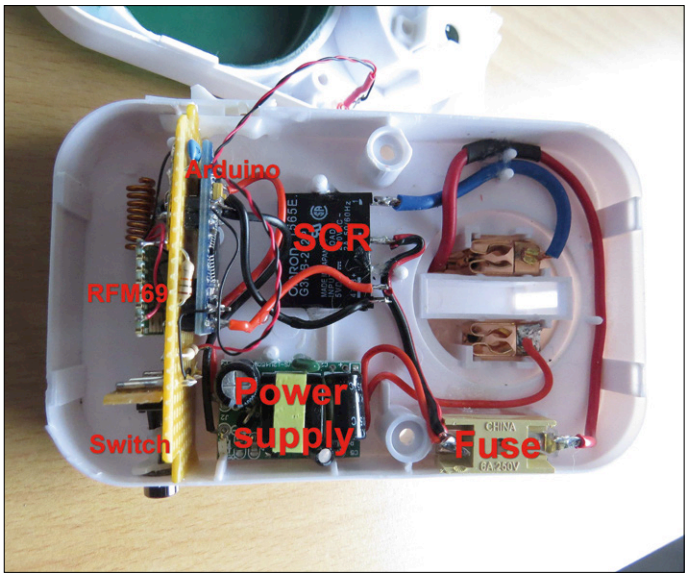


Figure 4. A DIG node in a plug-in timer housing.

(northbound) means from the node to OpenHAB (sensor data), while "sb" (southbound) means from OpenHAB to the node (commands).

- `nodeID` is the node identifier. Each node has a unique ID (address), which is determined when the code is compiled. The gateway always has node ID 01.
- `deviceID` is the function identifier, as previously described.

The MQTT message (payload) depends on the device and the direction.

- *Southbound*: The message contains commands for the node (ON, OFF, READ) or settings for the node, such as the number of seconds for the transmit interval.
- *Northbound*: The message contains the value of the parameter belonging to the DevID.

The gateway subscribes to all southbound messages for all nodes in the network by means of the following wildcard topic:

```
home/rfm_gw/sb/#
```

Of course, OpenHAB needs to receive all messages from the nodes, so it subscribes to the appropriate topics:

```
home/rfm_gw/nb/#
```

Table 4 shows a number of examples of MQTT messages.

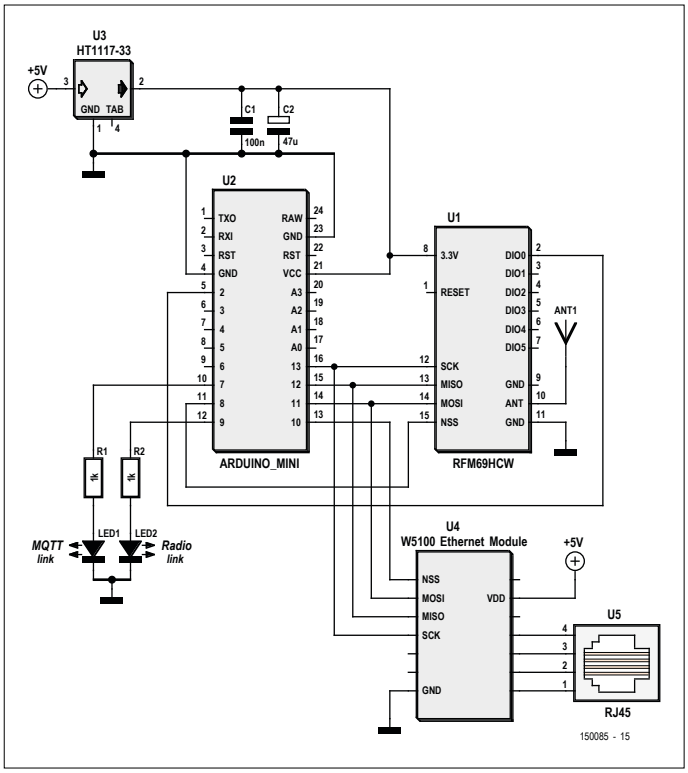


Figure 5. Schematic diagram of the gateway.

End nodes

The end nodes are based on Arduino boards. The following conditions and constraints are taken into account in the design:

- The RFM69 module has an operating voltage of 3.3 V, and the maximum permissible voltage on its inputs is 3.9 V. To keep things simple, we use an Arduino that also operates from 3.3 V. We opted for the Arduino Pro Mini, which operates from 3.3 V and can be connected to the RFM69 without level conversion. You can also use the Arduino Buono R3, which can be switched to 3.3 V.
- The RFM69 module briefly draws a significant amount current (130 mA) when transmitting, so the supply needs to have sufficient capacity.
- The Arduino and the RFM69 communicate with each other over the SPI bus. The standard Arduino pins are used for that purpose.
- The RFM69 may not be operated without an antenna. A piece of wire 8.6 cm long gives excellent results in most cases.

The schematic diagram of one of the end nodes is shown in **Figure 3**. This node is equipped with a DHT11 temperature and humidity sensor connected to pin 4 of the Arduino. A pushbutton is connected to digital I/O pin 8, and an LED is connected to pin 9.

The RFM69 module is connected to the Arduino over the SPI bus (SCK, MOSI, MISO and NSS). The 3.3 V supply voltage is provided by an AMS1117 voltage regulator, with enough decoupling capacitors to suppress current spikes.
The software loaded into the Arduino determines the function of the end node. The following end node devices have been developed so far:

- DHT is a node with sensors for temperature and humidity. This node has a digital output and a pushbutton.
- DIG is a simplified version of the DHT node. It only has a pushbutton and a digital output.
- RFID is a node with an RFID reader. When an RFID device is detected, the node sends the RFID identifier to OpenHAB.
- LCD is a node with a liquid crystal display. Text strings can be sent from OpenHAB to the display.
- RC is a node equipped with a 433 MHz transmitter module. It can be used to operate switches in the Dutch *KlikAanKlikUit* home automation system [4].
-

The software can easily be adapted to add or modify functions. The construction of a node is strongly dependent on its function. For example, the case of an inexpensive mechanical timer switch can be used to house a DIG node. An example of this is shown in **Figure 4**.

The gateway

The gateway consists of an Arduino, an RFM69 module and an W5100 Ethernet module. **Figure 5** shows the schematic diagram. The gateway also uses an Arduino operating at 3.3 V. On the gateway the SPI bus is shared by the RFM69 and W5100 modules. The SCK, MOSI and MISO signals are connected to both devices in parallel. Different Slave Select (SS) signals are used to select the individual devices on the bus: pin 8 for the RFM69 and pin 10 for the Ethernet module.
An AMS1117 also provides the 3.3 V supply voltage for the gateway. Two LEDs are provided to indicate the status of the gateway.
A separate module is used for the Ethernet connection. If you use an Arduino Buono board, you can use a Wiznet Ethernet

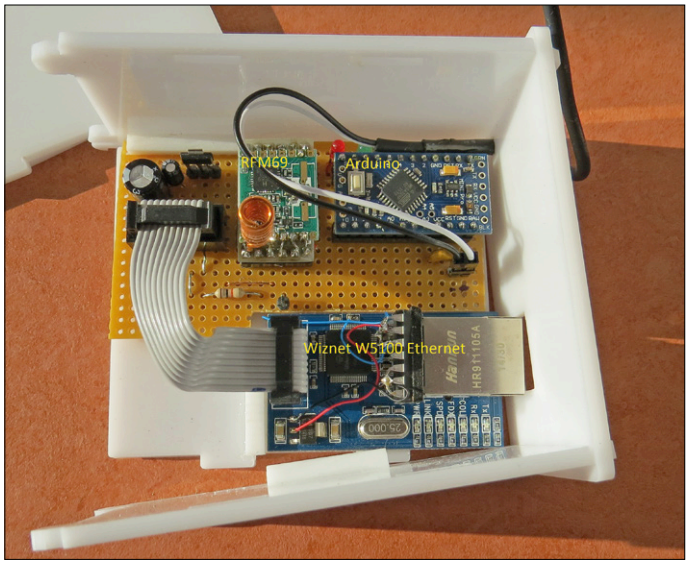


Figure 6. The author's fully assembled gateway.

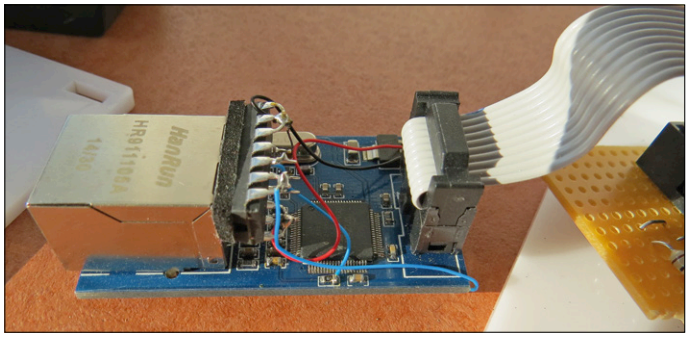


Figure 7. The 4011 is attached to the housing of the Ethernet connector and generates the SEN signal for the W5100 from the CS signal.

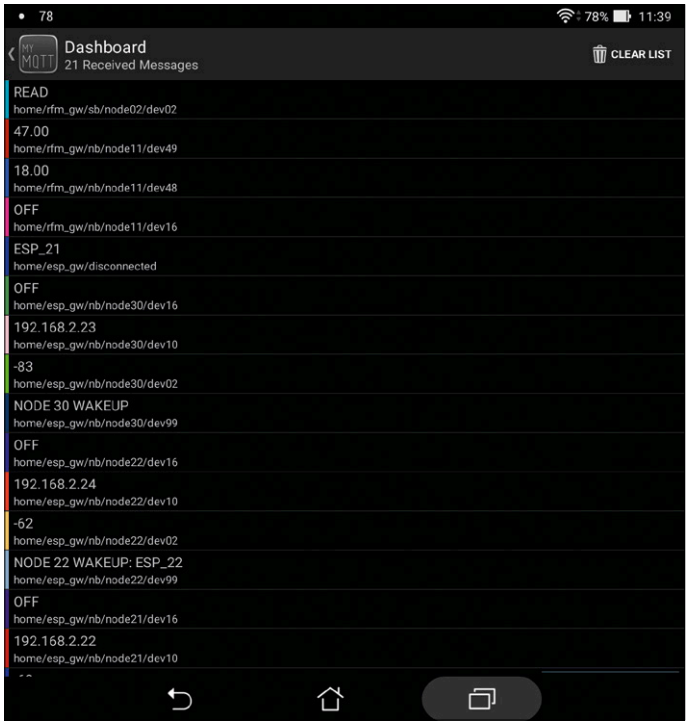


Figure 8. This screenshot shows some examples of MQTT messages.

shield instead. Only one gateway is needed for the entire system. For this reason, the author built the gateway on a prototyping board (**Figure 6**).

There are two things you have to pay attention to when building the gateway, as described below.

W5100 bug

The hardware of the W5100 is not directly suitable for sharing an SPI bus with other devices. When an SPI bus device is not using the bus, it is supposed to release the signal lines so that they can be used by other devices. The original W5100 is not designed to do this. The author solved this problem by using an inverter to generate the SEN control signal from the CS signal (see [5] for more details). The necessary inverter is integrated in newer versions of the W5100 module. The module which the author bought on eBay did not include this feature, so the inverter had to be added externally. As can be seen in **Figure 7**, a type 4011 IC was used for this. There you can also see that it is easy to connect the SEN signal line via a pull-up resistor on the circuit board. Remember to connect the unused inputs of the 4011 to ground.

Short-circuit via ICSP header

If you use a standard-format Arduino with the W5100 shield, the Arduino board is connected to the 5 V supply voltage of the Ethernet shield through the ICSP header. That pulls up the 3.3 V supply voltage line on the Arduino, which can result in damage to the RFM69. This can be avoided by cutting off the VCC pin of the ICSP header or bending it aside so that it does not make contact.

To be continued

In the following article we will describe the software for the gateway and the nodes, as well as the configuration of Mosquitto and OpenHAB. If you can't wait, you can already download the software at [6]. You must have Mosquitto available in order to test the gateway. It's also handy to have an MQTT client for testing. The MyMQTT app is a good choice for Android, and on a Windows system you can use Chrome Lens or MQTT.FX. **Figure 8** gives an impression of the messages you can expect to see.

This project has now been built by several other people. Extensions, issues and experience are discussed and shared on the forum site [7]. ⏮

(150085)

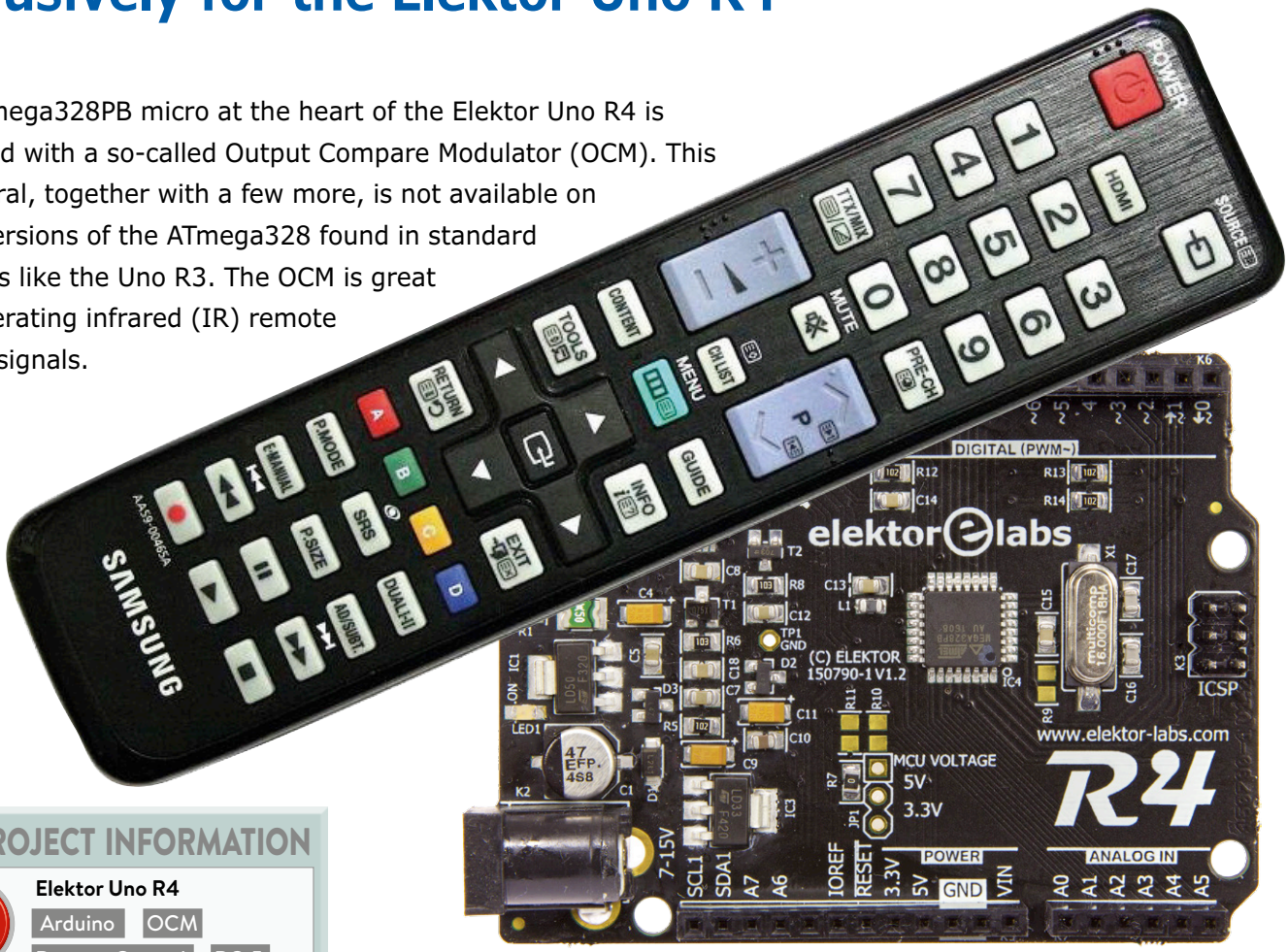
Web Links

[1] https://mosquitto.org
[2] www.openhab.org
[3] www.elektormagazine.com/tags/journey-into-the-cloud
[4] www.klikaanklikuit.nl (Dutch!)
[5] http://john.crouchley.com/blog/archives/662
[6] https://github.com/computourist/RFM69-MQTT-client
[7] http://homeautomation.proboards.com/board/2/openhab-rfm69-based-arduino

How to Produce NEC-style Remote Control Signals

Exclusively for the Elektor Uno R4

The ATmega328PB micro at the heart of the Elektor Uno R4 is equipped with a so-called Output Compare Modulator (OCM). This peripheral, together with a few more, is not available on other versions of the ATmega328 found in standard Arduinos like the Uno R3. The OCM is great for generating infrared (IR) remote control signals.



PROJECT INFORMATION

Elektor Uno R4

Arduino | OCM | Remote Control | RC-5

entry level
➡ intermediate level
expert level

1 hour approximately

Elektor Uno R4 computer with Arduino IDE

€25 / $30 / £20 approximately

By **Clemens Valens** (Elektor Labs)

The OCM combines the outputs of Timer3 and Timer4 — another Elektor Uno R4 exclusivity — and makes the result available on PD2, which is Arduino pin 2 and MCU pin 32. Here, combining means either AND-ing or OR-ing the two signals, as shown in **Figure 1**. The value of register PORTD2 determines which operation is executed. If PORTD2 is set to '1', then OR-mode is selected; if PORTD2 is set to '0', AND-mode is what you get. Signals similar to those from Figure 1 are commonly found in infrared remote controls where they are used to modulate the current through an IR LED.

My home, and yours too I reckon, is littered with remote controls (RCs). Some time ago when I needed a Philips RC-5-compatible RC I had to try them all only to discover that none of them used the RC-5 protocol, not even my Philips DVD player (using RC-6). Most of my RCs appeared to use some variant of the Japanese NEC protocol instead. This is understandable as most of the remote-controlled equipment in my house, and yours too I reckon, was made

in Asia. So, now that we have this nice OCM peripheral, let's use it to produce NEC-compatible RC signals.
An NEC-type RC code starts with a preamble or burst of 9 ms, followed by 4.5 ms of silence (a pause). Next the bits, 32 in total, are transmitted one after the other. A '0' consists of a burst and a pause of both 562.5 µs (1,125 µs in total) and a '1' is a burst of 562.5 µs with a pause of 1,687.5 µs (2,250 µs in total). The sequence must be terminated by a closing burst (postamble) to 'close' the last bit. All durations mentioned here are multiples of 562.5 µs. In what follows, a 'pulse' is the duration of a burst, and a 'period' is the duration of a burst + pause.
A burst is nothing more than a high-frequency signal, a carrier, switched on and off by the modulator. We can use Timer3 to generate a carrier and modulate it with Timer4. The other way around is equally valid, but in life one is forced to make choices. All timers in the ATmega328, including Timer3 and Timer4, are capable of generating interrupts when they overflow or when they exceed a certain value. This offers a nice mechanism to easily modulate the carrier with the data we wish to transmit: every time the interrupt fires, the next bit to transmit is read from the TX buffer and used to determine the pulse and period needed to send a zero or a one. The sketch that follows shows how we can accomplish this.

```
/*
 * IR Transmitter
 * Uses Output Compare Module (OCM), ATmega328PB
   only.
 */

#define CARRIER  38000
#define RC_CODE  0x6170807f

const int ir_led = 2;
const uint16_t pulse = 1125; // 2*562.5
uint32_t tx_buffer = 0;
uint32_t tx_index = 0;

#define TIMER3_ON  TCCR3B |= (1<<CS30) /* prescaler
   1 */
#define TIMER3_OFF  TCCR3B &= ~(1<<CS30)
#define TIMER4_ON  TCCR4B |= (1<<CS41) /* prescaler
   8 */
#define TIMER4_OFF  TCCR4B &= ~(1<<CS41)

ISR(TIMER4_COMPB_vect)
{
  if (tx_index!=0) // Any bits left?
  {
    OCR4B = pulse; // Set pulse length.
    if ((tx_buffer&tx_index)!=0)
    {
      OCR4A = OCR4B<<2; // Set '1' period.
    }
    else
    {
      OCR4A = OCR4B<<1; // Set '0' period.
    }
    tx_index >>= 1; // Next bit.
  }
  else
  {
    if (OCR4A>=2*OCR4B)
    {
      OCR4A = OCR4B + 10; // Send closing burst.
    }
    else
    {
      // Done, clean up.
      TIMSK4 &= ~(1<<OCIE4B); // Disable interrupts.
      TIMER3_OFF;
      TIMER4_OFF;
    }
  }
}

void ir_send_code(uint32_t code)
{
  // Copy code to TX buffer.
  tx_buffer = code;
  tx_index = 0x80000000; // 32 bits.
  // Setup timer4 for preamble.
  OCR4A = 2*12*pulse;
  OCR4B = 2*8*pulse;
  TIFR4 = (1<<OCF4B); // Clear pending interrupts.
  TIMSK4 = (1<<OCIE4B); // Enable compare interrupt.
  // Go!
  TIMER3_ON;
  TIMER4_ON;
}

void setup(void)
{
  pinMode(ir_led,OUTPUT); // Enable output.
  digitalWrite(ir_led,LOW); // OCM in AND mode.
  // Timer3 mode 4 (CTC), toggle output on match.
  TCCR3A = (1<<COM3B0);
  TCCR3B = (1<<WGM32);
  OCR3A = 16000000/CARRIER/2; // Toggle at 76 kHz ->
   38 kHz.
  OCR3B = OCR3A>>1; // 50% duty-cycle.
  // Timer4 mode 15 (Fast PWM), clear output on
   match.
  TCCR4A = (1<<WGM41) | (1<<WGM40) | (1<<COM4B1);
  TCCR4B = (1<<WGM43) | (1<<WGM42);
  ir_send_code(RC_CODE);
}

void loop(void)
{
}
```

It all starts in the function `setup` with the declaration of pin 2 (PD2, to which the IR LED is to be connected) as an output. Skip this step and nothing will ever come out of this pin. Making the

output low not only switches off the LED, it also puts the OCM in AND mode. Next we set up Timer3 in CTC (compare) mode 4 for a frequency of 38 kHz, a popular RC carrier frequency. Note how the timer seems to be set up for twice this frequency, but the factor of two is necessary to compensate for the toggling of the output pin on every compare match, effectively dividing the signal frequency by two and resulting in a 38-kHz signal at the output. Pin 2 is the OC3B output controlled by OCR3B, consequently we set this register to half the value of OCR3A to obtain a duty-cycle of 50%. Timer4 is configured in Fast PWM mode 14 where OCR4A determines the frequency and OCR4B the duty-cycle. The rest of its configuration is delayed till the function `send_code`.

This function starts by copying the code to transmit into the 32-bit TX buffer. Because a code starts with a preamble, the corresponding pulse (9 ms) and period (13.5 ms) values are loaded into the pulse (OCR4B) and period (OCR4A) registers. Here a multiplication by two is needed because the pulse and period are specified in microseconds, but since the MCU clock is 16 MHz and the prescaler can only divide it by 1, 8, 64, 256 or 1024, not by 16, we are forced to use 8 instead and we're off by a factor of two. The next step is to prepare for interrupts and start the timers.

The COMPB interrupt service routine is called at the end of a pulse. When this happens, and all the bits haven't been sent yet, the timer needs to be reconfigured for a period of either 1,125 µs (if the bit to transmit is a '0') or 2,250 µs (if the bit to transmit is a '1'), and set the pulse length to 562.5 µs (identical for '0's and '1's). As before, all the values are multiplied by two to compensate for the prescaler value of 8 which we would have liked to be 16. The 32-bit code must be terminated by a closing burst, otherwise the receiver can't tell if the last bit is a '0' or a '1'. When `tx_index` equals zero we know that all bits have been sent. We can now use the OCR4A register, for instance, with a special value to send a final burst, meaning that the ISR will run one more time. If we choose a special value smaller than 1,125 µs (but larger than OCR4B), this state will be easy to detect by the ISR the next time it is called. Naturally it is possible to use another mechanism with a static variable or something similar, but the register is available, so why not use it? The last thing the ISR does after the preamble, the 32 code bits and the stop burst have all been sent, is switch off the lights, ermm, the timers as they are no longer needed.
The function `loop` has remained empty because we send the RC code only once after pressing the Reset button.

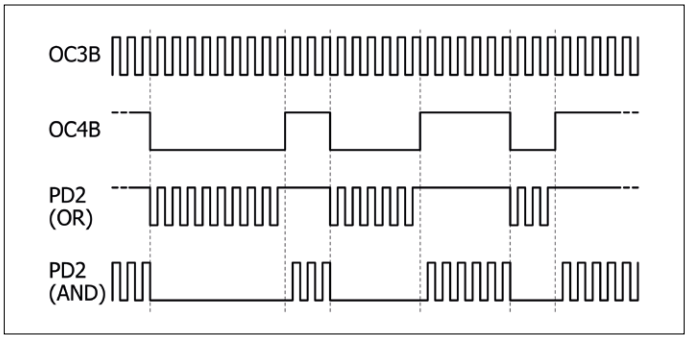


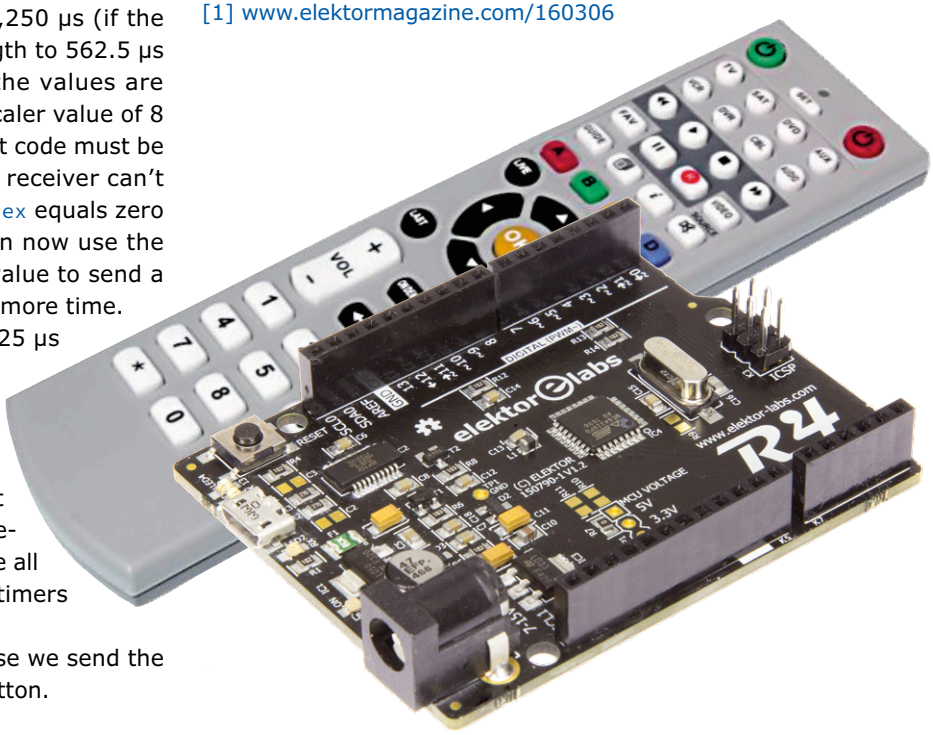
Figure 1. In OCM mode PD2 outputs either OC3B OR OC4B (PORTD2 = 1) or OC3B AND OC4B (PORTD2 = 0).

Transmitting a code is fully interrupt-driven and the rest of the program can do other things while this runs in the background. The program can poll the OCIE4B bit in the TIMSK4 register to check if the transmitter is still busy. Furthermore, as you can see, there is no interaction whatsoever between the timers 3 and 4 except for the ISR switching them both off at the end (which is optional), and there are no pins to be set or cleared. Everything is handled directly by the hardware. The downside of this simplicity is that it works on pin 2 only. ⏮

(160306)

Web Link

[1] www.elektormagazine.com/160306



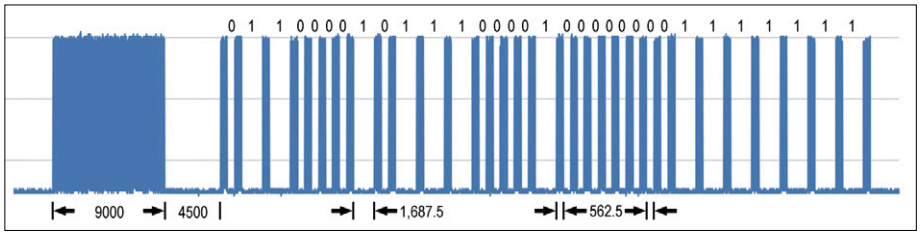


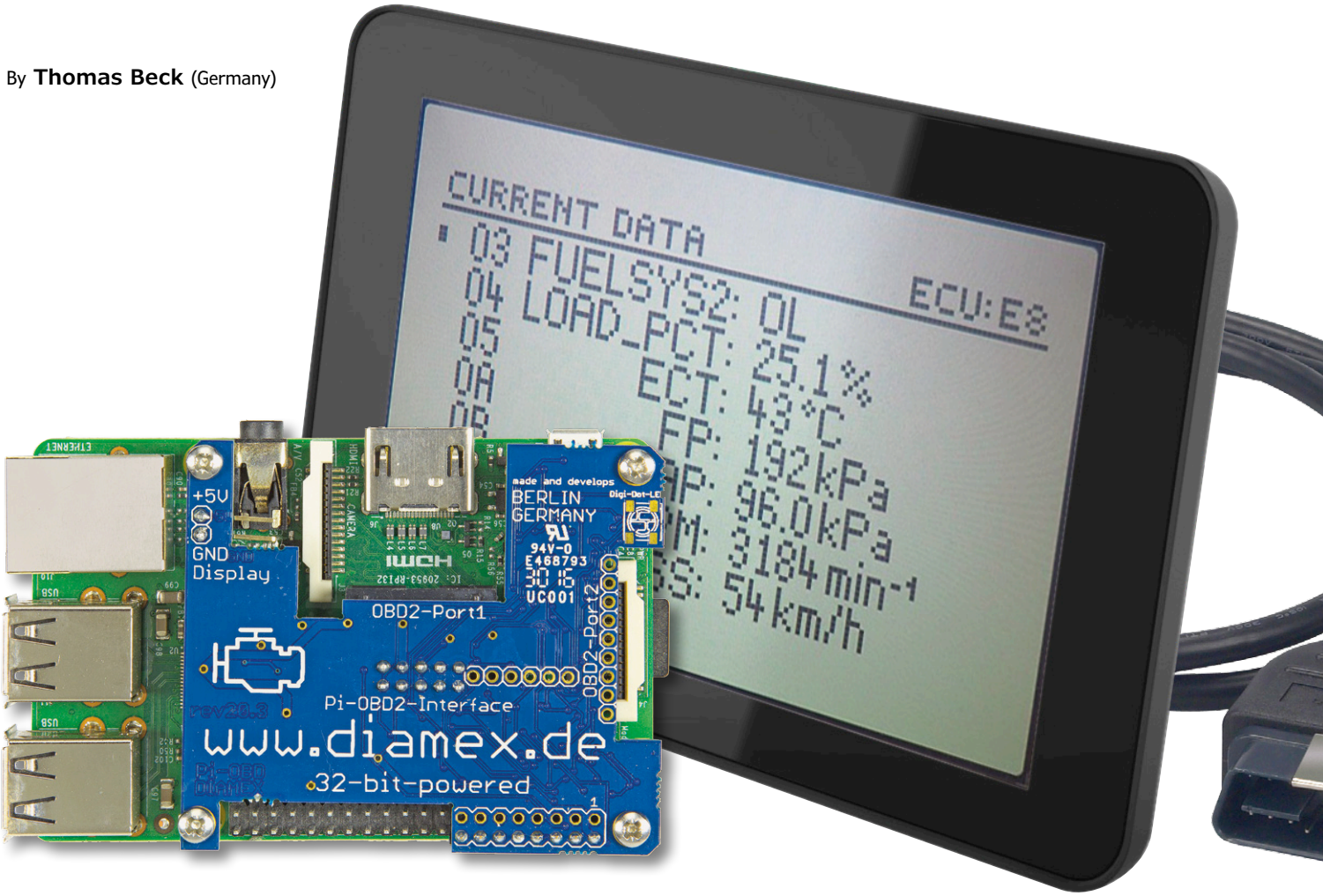
Figure 2. A scope plot of the signal on PD2 produced by the sketch.

SHOPPING LIST
→150790-91
elektor Uno R4

OBD2 Handheld using a Raspberry Pi

With new diagnostic software

By **Thomas Beck** (Germany)



The Pi-OBD-HAT is an off the shelf add-on board for the Raspberry Pi which converts the tiny computer into a dedicated OBD2 diagnostic tool. One disadvantage was the need of a terminal program running on a PC for control. A much more convenient and neater solution is provided by the HHGui OBD2 diagnostic software.

The GUI software HHGui is a variant of the HHemu program I wrote for the OBD2 handheld emulator [1] which is a sort of Swiss army knife tool for developers working with the OBD2 analyzer 'NG/ DIAMEX Handheld Open' (HHOpen). This unit is a handheld OBD2 diagnostic tool published in September 2009 in *Elektor Magazine* [2]. The OBD2 analyzer NG employs the DIAMEX DXM OBD2 module [3].

The development process

The software development process is a little unconventional and shows how firmware originally written for a microcontroller can be ported to a Raspberry Pi or PC, ideally without making any changes to the firmware source code.

HHEmu was originally conceived as a pure development tool to test new ver-

sions of the AVR microcontroller firmware for the OBD2 analyzer on a PC without the need to plug it into an OBD-equipped vehicle. In addition to the firmware HHEmu contains an OBD2 emulator which supplies simulated OBD2 control unit data to the firmware via a simulated DXM OBD2 module.

The next step in the development process HHEmu was provided with a serial interface to a PC. Together with a Bluetooth expansion board published in April 2010 [4] this gave the OBD2 analyzer NG a great deal of added flexibility. The firmware running on a PC within HHEmus can control a real NG analyzer and its DXM module using a serial interface to the Bluetooth adapter with the Bluetooth extension. This makes it possible to test new firmware versions on a real vehicle without first flashing the firmware into the OBD2 analyzer. This was a real milestone in the development process! In the meantime HHEmu has become the standard diagnostic software for DXM-based diagnostic interfaces. In addition the OBD2 emulator in HHEmu can now act as a data generator to test OBD2 diagnostics tools produced by other manufacturers or for the development of new OBD2 software. As well as the improved test possibilities for firmware development there are also other benefits of porting to a PC. With each future firmware update the diagnostic software automatically receives the functionally identical update.



Now HHEmu has been ported to the Raspberry Pi. After all the changes necessary for operation with the Pi-OBD hat and the removal of any OBD2 analyzer-NG specific menus the first version of HHGui is ready for publication.

Operational modes

We can say, because of its development path that HHGui is essentially OBD2 diagnostic software with a user interface that mimics the look of the OBD2 analyzer NG as well as internally running its firmware adapted to the PI-OBD module. The current range of the firmware functions available are given in the **Table**.

HHGui presents the OBD2 user menus and data on a simulated LCD display scaled to the larger graphic display of the RPi. This is a compromise so that in addition the official 7" touchscreen recommended for the RPi it also supports displays as small as 320x240 pixels. A modification of the Pi-OBD module is necessary as described in the first project update of the Labs-project [5].

The second level of functionality implemented in HHGui is the OBD2 simulator which can be used to represent the Pi-OBD module and up to eight configurable OBD2 controllers. This feature allows you to test the full range of software functions without the need for a Pi-OBD module or vehicle. Plugging into a real vehicle will also not give you access to all the possible OBD2 functions and subfunctions because many of these are exclusively related to the type of engine fitted (i.e. petrol (gas) or diesel powered vehicles). Another bonus is that this simulator is completely harmless, there's no chance of accidentally deleting any of the vehicle's OBD2 data. The range of OBD functions provided by the simulator is identical to the range of functions in the firmware. Commands to control the OBD2 simulator entered via the keyboard are described in the Labs HHEmu project [1].

What's possible using OBD2 diagnostics?

So returning to the main functions of the OBD2 diagnostic software, what can we expect to achieve with the services available via the OBD2 interface?

When you take a close look at the ISO 15031-5 [6] specification (or the identical SAE J1979 [7]) it becomes clear that via the OBD2 port we can read emission-relevant information and trouble codes and partially delete them. This information is mainly generated by the vehicle's ECU. Vehicles with automatic transmission can also produce OBD2 trouble codes from the Power-train Control Module. Less often the vehicle will be fitted with additional control systems or even multiple ECUs in the case of hybrid vehicles. General settings such as automatic door locking or inhibiting the warning gong sound for seat belt reminder cannot be made via the OBD2 connector. With the exception of the Malfunction Indicator Lamp (MIL) none of the other fault indicator lamps on the instrument cluster or the service warning lamp or service interval can be reset. These actions can only be performed using the manufacturer's diagnostic equipment which is usually specific to the vehicle model.

That may seem a little disappointing at first. The advantage of this manufacturer standardized OBD2 diagnostics is that now after 20 years since its introduction in the US (and since the year 2000 for petrol-powered vehicles in Europe), it works with almost every vehicle. Since 2008 the proliferation of different protocols for transferring OBD2 data has been drastically reduced so that now only the CAN protocol can be used.

Take a closer look at all the available OBD2 sub-functions you will find an incredible amount of sensor information, counter values and other measurements (altogether more than 100) that's sure

The OBD2 functionality of HHGui

HHGui supports the OBD2 services contained in ISO15031-5. A more detailed description can be found on the project page [5], for more on the sub-functions refer to [6], [7] or [8]. The range of supported services will depend on the vehicle type and its OBD2 control unit. HHGui will interrogate the respective control unit and display the supported sub-functions.

- Support for up to eight OBD2 ECUs (Electronic Control Units) according to ISO 15765-4 [9]
- OBD2-Service 0x01: Read Live or Current Data; support also for Parameter Identifier (PIDs): 0x00...0x60, 0x70, 0x80, 0x8D sub-functions
- OBD2-Service 0x02: Read Freeze Frame Data, supports PIDs as for Service 0x01
- OBD2-Service 0x03: Read confirmed Diagnostic Trouble Codes (DTC)
- OBD2-Service 0x04: Clear DTCs, stored values, MIL status, OBD2 monitor and other data
- OBD2-Service 0x07: Read Pending DTCs
- OBD2-Service 0x09: Read Vehicle Information; support also for (InfoTypes): 0x00, 0x02, 0x04, 0x06, 0x08, 0x0A, 0x0B
- OBD2-Service 0x0A: Read permanent DTCs

to make you want to explore further. Apart from the sensor information you would expect to have access to from the Engine Control Unit you can also read data originating from other equipment on the vehicle such as the brake control module.

In addition to the raw OBD2 data some values undergo processing before they are displayed on the instrument panel. Some examples are:

- value smoothing (outside air temperature, fuel gauge and generally any instrument with a pointer driven by a stepper motor);
- values with a plateau function with a limited range of interest (coolant temperature);
- values with added offsets, to provide a margin for error and ensure that displayed values will not allow the driver to unknowingly exceed any legal restriction or simply to comply with the manufacturers recommended limits (speedometer, engine RPM).

Data supplied from the OBD2 interface can also be raw sensor values without any processing so we can read the actual speed rather than the value displayed on the speedometer.

There is also data that you might be interested in reading which the manufacturer does not display such as oil temperature or turbocharger boost pressure. Engine load values may also be of use.

The average user might also be interested to know the state of readiness of the vehicle for its next inspection and test (the annual MOT in the UK, or vehicle safety inspection in the US). To get this information it's necessary to run all the OBD2 test programs (OBD2 monitors) at least once after fault conditions have been erased. This condition can be checked in HHGui when in Inspection/Maintenance readiness menu the status of all the OBD2 monitors is displayed as '… monitoring ready: YES'.

In addition to this easily understandable OBD data we also have more esoteric information such as lambda probe readings more of interest to the specialist. A list of all the available Parameter IDs (PIDs) is too long to include here. The latest full-version of the official specification is available at considerable expense, as an alternative it's worthwhile taking a look at the Wikipedia entry [8].

Now we turn to a subject which is of more interest to developers.

HHGui: Structure and Function

Functional Principle

The firmware is executed in a thread — all inputs and outputs of the firmware are processed and responded to in additional threads. All of the microcontroller registers and interrupts need to be simulated.

Actual implementation

HHGui consists of six threads. **Figure 1** is a simplified block diagram showing the interaction between the threads. The source files use different names as a result of the program development history. The Pi-OBD module thread is called

PI-OBD: The Hardware

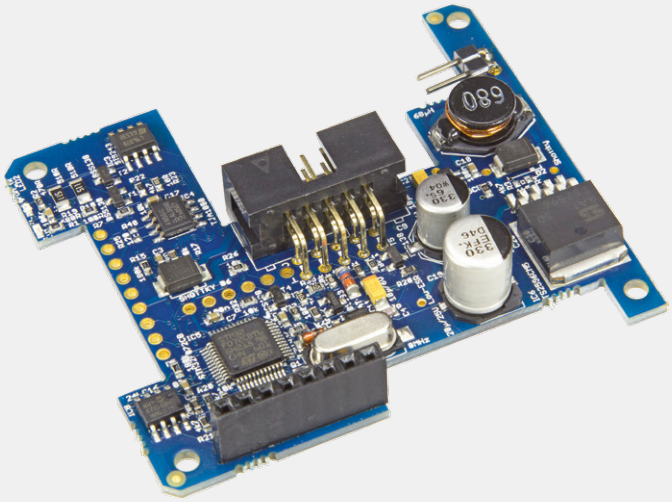
The PI-OBD-HAT [10] from Diamex turns a Raspberry Pi into an OBD2 diagnostics adapter. Fitted with the 7" touch screen you have a handy and powerful stand-alone diagnostic tool for vehicles with an OBD2 interface. The RPi and its display are powered directly by the vehicle's own 12 V supply.

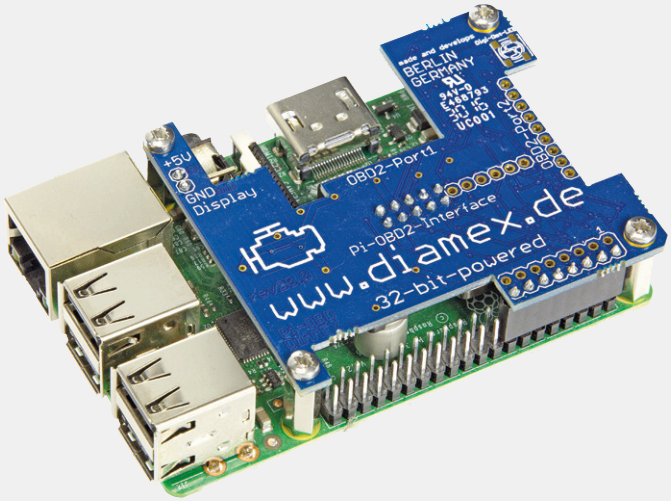
Any of the Raspberry Pi variants can be used, but the RPi 2 Model B or RPi 3 Model B are preferred owing to their higher processor speed. The PI-OBD hat uses just eight pins of the RPi headers to provide power, connections for serial data interface and reset signal to activate the PI-OBD bootloader. Serial communication takes place using the TxD and RxD pins at 115,200 Baud with 8N1.

Plugging it all together

Plug the Pi-OBD hat into the Raspberry Pi headers. The 8-way header socket plugs onto eight of the I/O pins along the edge of the RPi board. The hat is held in place using the threaded spacers and screws supplied.

The Raspberry Pi is powered by the vehicle battery via the OBD2 plug. No other external power source is required unless





DxmThread and in the Firmware for den OBD2 analyzer NG HhopenThread. The shorter names are used in the following description.

The NG-Analyzer firmware is executed in the HhopenThread. This thread communicates with other threads in various ways. Display data for the LcdThread is passed via the SPDR register; RGB values for the LED back light are passed to LcdThread via registers OCR1A/B/C. The UDR0 register is used to pass AT commands, OBD requests and responses to the DxmThread. A firmware reboot is made via the WDTCR register or from MainThread of HHGui. Key presses from the MainThread are received in the PINA register. HhopenThread is controlled via the Interrupt service routines for CounterThread and DxmThread. Details are given in the descriptions of the other five threads.

The MainThread takes care of evaluating command line parameters, initializing the serial interface, generation of other threads and at the end of the program

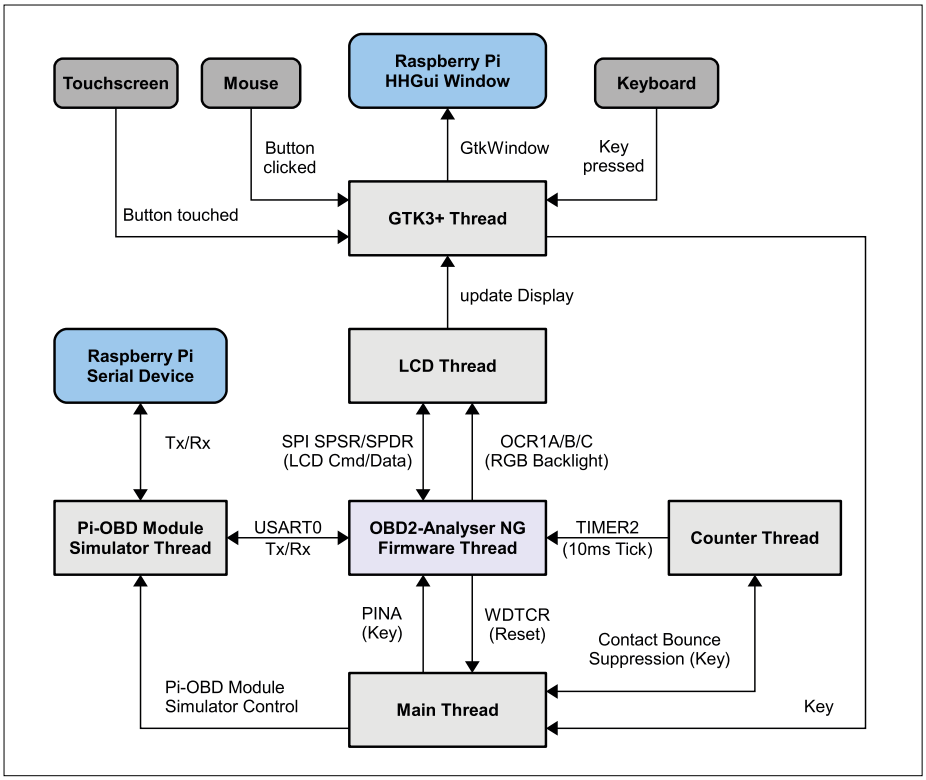
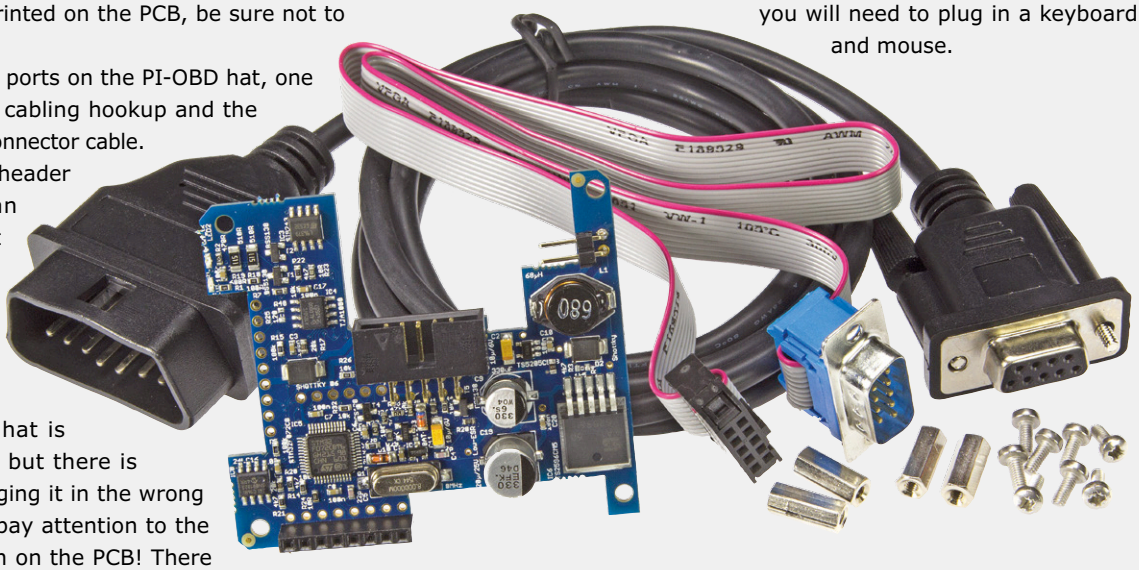


Figure 1. HHGui: Block diagram.

you are testing or configuring the unit away from the vehicle when it will then be powered via its USB connection. Power for the 7" Raspberry Pi display is provided by the two-pin header on the hat. The photo is from under the RPi with the OBD hat fitted and the display PCB above. A description of the header pin connections is printed on the PCB, be sure not to get them mixed up!

There are two OBD2 ports on the PI-OBD hat, one is suitable for 'free' cabling hookup and the other with a sub-D connector cable.

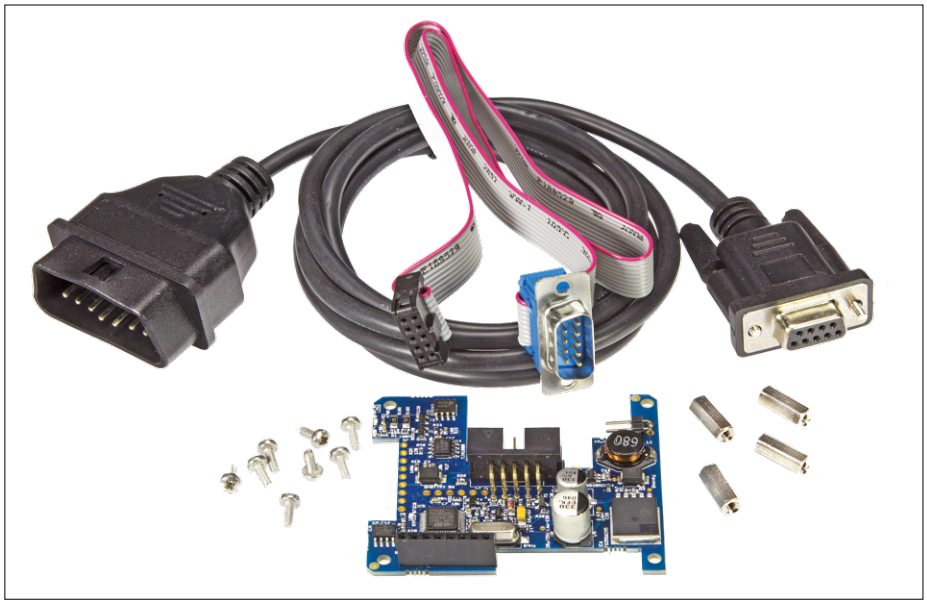
A single-row 9-way header plug (or socket) can be used to connect to Port 2 along the board edge. The port 2 connector has the advantage it can be plugged in and out whilst the hat is attached to the RPi but there is also the risk of plugging it in the wrong way round. Always pay attention to the printed identification on the PCB! There can be no confusion on port 1 because the connector on the board has a detent position and the 9-pin sub-D connector at the other end of the cable can only be connected one way round. The flat band connector on this port must be inserted before the hat is assembled onto the RPi, it can't be plugged in or withdrawn without first releasing the hat.



Fire it up

To begin using the PI-OBD HAT you will need some basic knowledge of the Raspberry Pi environment, the use of Linux and the software installation. For the purposes of configuration and operation, unless you are using a touch screen with the RPi, you will need to plug in a keyboard and mouse.

The first exercise now is to enable the RPi's serial interface but a slightly different procedure is necessary for each different RPi model and space limitations here makes it necessary to provide this information via this link [10]. The unit will be ready for use after a functional test using a terminal program.



returns these resources. The thread also evaluates key press information sent by GtkThread in an endless loop. It is necessary to differentiate between the keys used to control the OBD2 simulator and the Up/Down/ESC/OK keys used for the HhopenThread. For these keys used by the HhopenThreads it is necessary for the firmware to ignore multiple contacts produced by contact bounce. The key press must be present for 40 ms in the PINA register in HhopenThread to be recognized. This 40-ms window is measured using the CounterThread. Keys used only for the OBD2 emulator produce changes in settings in the DxmThread.

The loop that evaluates pressed keys terminates when a watchdog reset is triggered from HhopenThread (by holding down the ESC key) or by terminating HHGui using the Q key.

LcdThread evaluates the commands and display data received via the simulated SPI interface from the HhopenThread and translates them into commands for the GTK3+ graphics library. The LED backlight RGB values in the OCR1A/B/C registers define the displayed colors together with the contrast setting (via the SPI interface). The LCD in the OBD2 analyzer NG uses an ST7565R display controller so LcdThread contains an ST7565R simulator for the ST7565R commands used by the Firmware.

The GtkThread is necessary because the GTK3+ main loop (function *gtk_main()* in the GTK3+ library) blocks the rest of the gtk code. GTK3+ drawing functions should only be called from inside GtkThreads or in context of the GTK3+ main loop. When LcdThread requests a display update, it installs a callback-function, which will be called later from the GTK3+ main loop. The GtkThread evaluates all kinds of input events and generates the Up/Down/ESC/OK button presses for the HhopenThread or key input events to control the OBD2 emulator.

The CounterThread is primarily used to simulate the interrupt generated by Timer2 which occurs every 10 ms. In response the `TIMER2_COMP_vect()` Interrupt Service Routine (ISR) is called every 10 ms. Apart from that it is used in the MainThread in contact debouncing as described earlier.

In the OBD2 software mode the DxmThread acts as an interface adapter and just transfers the data unchanged. The received data in the UDR0 register (USART0-I/O data register) which may be AT commands for the Pi-OBD module or OBD2 requests, are sent to the real PI-OBD module using the configured serial interface. After each byte is read the firmware is notified that the byte has been sent via a call of the `USART_UDRE_vect()`ISR indicating that next byte can be written to the register. The reply from the Pi-OBD module is received through the serial interface and passed to the firmware via register UDR0. The reply (OK, Pi-OBD module prompt character >, error or OBD2 data) is handled bytewise in the same way by writing to the UDR0 register and a call to the `USART0_RX_vect()`ISR tells the firmware that the register has been read.

In OBD2 emulator mode the DxmThread emulates the Pi-OBD module and one or more OBD2 controllers. There is no connection to the real serial interface. ⏮

(160204)

FROM THE STORE

→ **SKU 17944**
PI-OBD-HAT Module (built and tested)

→ **SKU 17415**
Official 7"- Raspberry Pi touchscreen

→ **SKU 17631**
Raspberry Pi 3

Web Links

[1] HHEmu: www.elektormagazine.com/labs/firmware-update-and-emulator-for-obd2-analyser-ng-wireless-obd2

[2] OBD2-Analyser NG: www.elektormagazine.com/magazine/elektor-200909/19167

[3] DXM-Modul (German!): www.diamex.de/dxshop/DIAMEX-DXM-OBD2-Modul

[4] Bluetooth for OBD-2: www.elektormagazine.com/magazine/elektor-201004/19297

[5] HHGui: www.elektormagazine.com/labs/obd2-for-raspberry-pi

[6] www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=66368

[7] http://standards.sae.org/j1979_201408/

[8] https://en.wikipedia.org/wiki/OBD-II_PIDs

[9] www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=67245

[10] Pi-OBD-HAT: www.elektor.com/pi-obd-hat-obd2-module-for-raspberry-pi

Elektor Labs Pipeline

Cool projects keep being posted on the Elektor Labs website. Some are more elaborate than others; all however are interesting and made from the most treasurable commodity on earth: ~~love~~ silicon.

Ultra-secure peer-to-peer communication system

Are your communications secure? There are many overhyped 'cryptophones' and crypto software programs available today, but can you really trust them? Trojanized versions of good programs like PGP and software key loggers for all kinds of operating systems are a common thing. By the year 2020, every computer connected to the Internet will have to be considered as compromised. If privacy is a real concern to you, build this crypto system and enjoy virtually unbreakable peer-to-peer communication!



https://goo.gl/OhXNEu

Build an electronic die (without cloud connection)

Sometimes you just want to make something simple, just for the fun of it or to get high on the delicious fumes of hot 60/40 solder and flux. This electronic die is such a project: easy to build, easy to get to work, and easy to use. Need more than one? Build more than one; it's that simple. No microcontrollers involved, no switched-mode power supply, no cloud connection, just a good old 555 and a 4017 Johnson counter. Perfect to get your mind off daily worries or to improve your soldering skills.



https://goo.gl/4tiAu1

Make your own smartwatch

The nWatch is a wearable development platform in the shape of a smartwatch. Based on an STM32 microcontroller, it is equipped with a full-color graphic touch screen, MP3 decoder, position sensors, Bluetooth 4.0 and much more. A 3D printable watch enclosure design is available too. But the really cool thing about it is its open- source design that you can modify. Intended as a learning system, this project was one of the winners of the 2016 Elektor Fast Forward Award and is scheduled for publication soon.

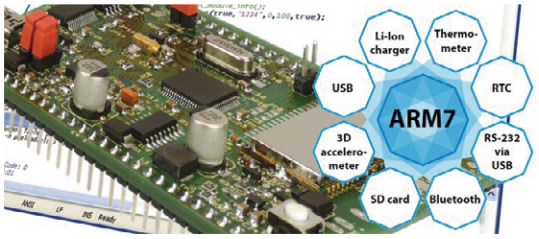


https://goo.gl/V1EUOC

The Scepter — IoT is nothing new

The Elektor Scepter is a small ARM7 board based on the LPC2148. The board features Bluetooth, a battery charger, an SD-card slot, an accelerometer, a thermometer and USB. Published in Elektor Magazine back in 2010 as an open source & hardware project the support files got somewhat dispersed after website redesigns, redirects, hosting changes, and what not. Now all seven projects and related articles have been compiled on the Labs website for you to enjoy. ⏮

(160293)



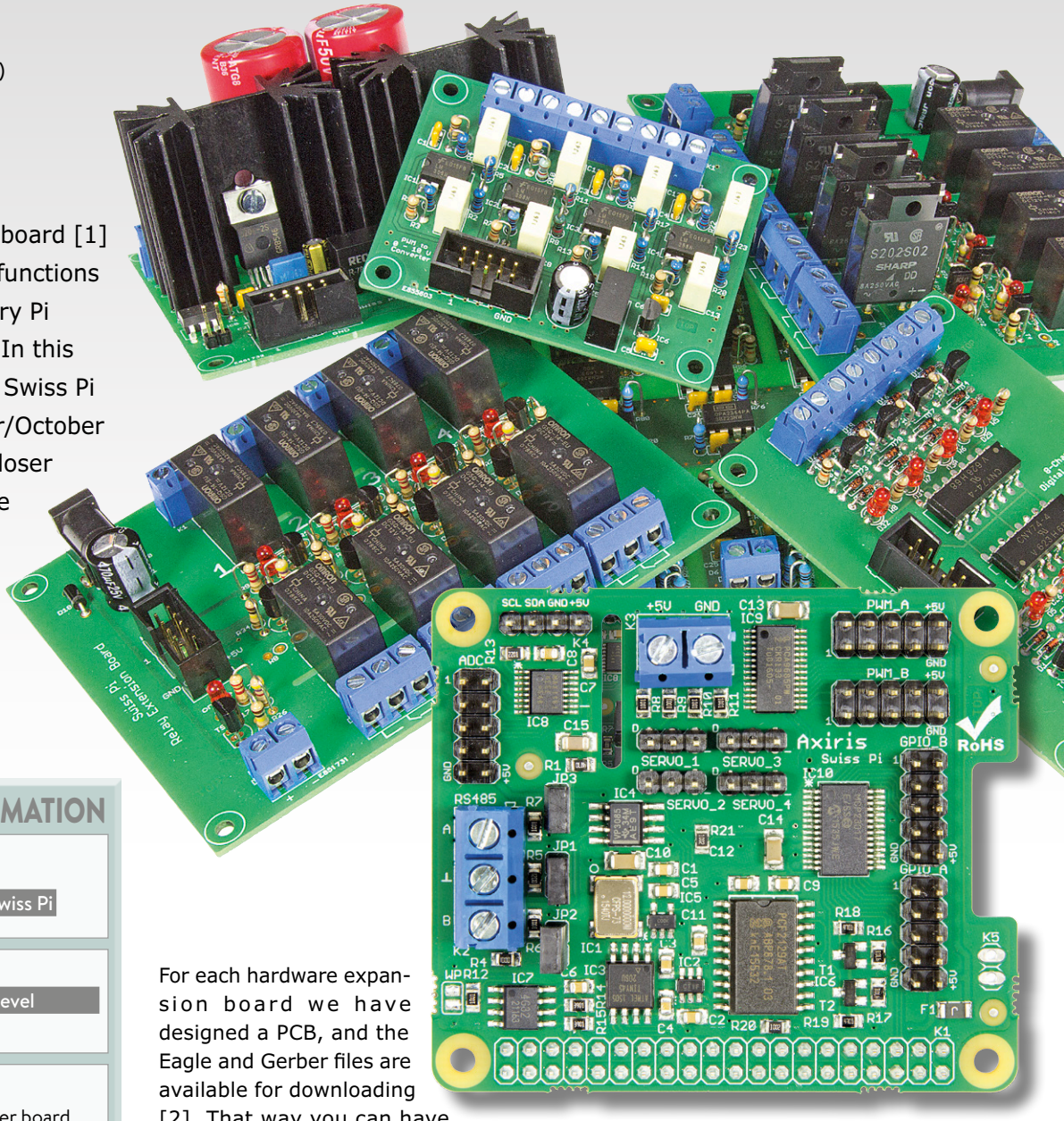
https://goo.gl/kuCiyr

eLABS PROJECT

Swiss Pi Extensions

Hardware extensions

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



The Swiss Pi expansion board [1] provides a lot of useful functions for the popular Raspberry Pi single-board computer. In this second follow-up to the Swiss Pi article in the September/October 2016 issue, we take a closer look at several hardware extensions developed specifically for the Swiss Pi board.

PROJECT INFORMATION

Microcontrollers
Raspberry Pi | Swiss Pi

entry level
➡ intermediate level
expert level

0.5 to 1.5 hours per board

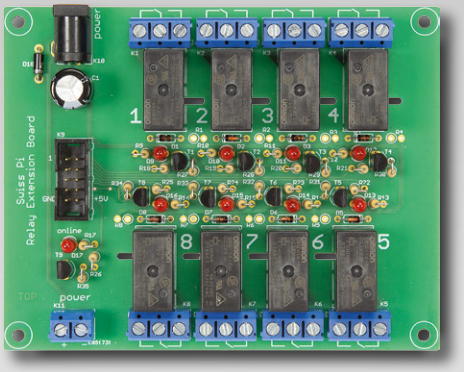
Soldering station with suitable solder tip

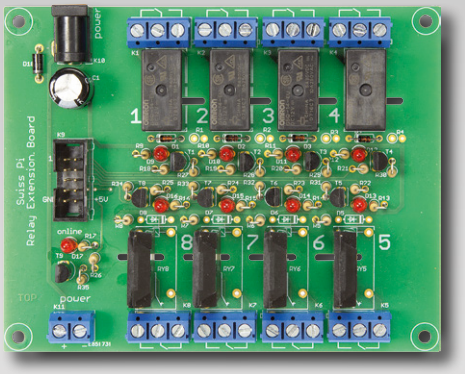
€25/£20/$30 to €50/£40/$60 per board

For each hardware expansion board we have designed a PCB, and the Eagle and Gerber files are available for downloading [2]. That way you can have the boards made by a PCB manufacturer of your choice. We have avoided the use of SMD components as much as possible, to make it easy to build the boards. If you want to go a step further and design something yourself, you can draw on these circuits for suitable inspiration. Naturally, these boards are not limited to use with the Swiss Pi; they can be used in a wide variety of environments. There is nothing to stop you from using them with your own microcontroller circuits or with an Arduino board.

Eight-channel relay board

The eight-channel relay board (**Figure 1**) uses relays from the Omron G5Q series. Sharp S202S02 solid-state relays can also be mounted on the board in place of conventional relays. Unfortunately those





relays are no longer being made, but for now they are still readily available on eBay. There is no other direct alternative because the S202S02 is by far the only solid-state relay that fits within the footprint of a G5Q relay. The S202S02 only has an LED at the input, so a series resistor is necessary.

Despite their compact dimensions, the G5Q relays fulfill all requirements with regard to isolation distances and clearances for use with AC line voltage. By contrast, the pins of the solid-state relays are closer together, so we provided slots in the PCB to fulfill the clearance distance requirements despite this limitation. Although the G5Q relays can switch a maximum of 10 A, the actual current has to be limited to about 5 A due to the width of the PCB tracks.

The relays are driven by standard NPN transistors, and there is a status LED for each relay. There is also a power indicator LED on the relay board, which is lit

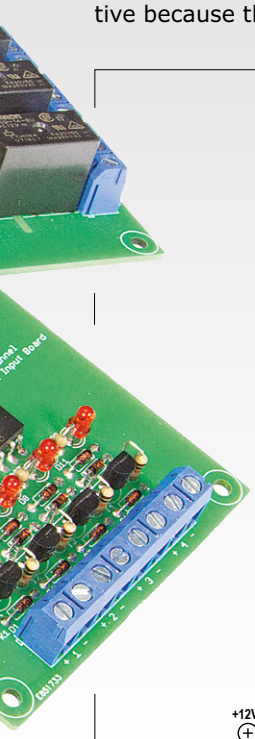


Figure 1. The eight-channel relay board is suitable for both conventional G5Q-SPDT relays and S202S02 solid-state relays.

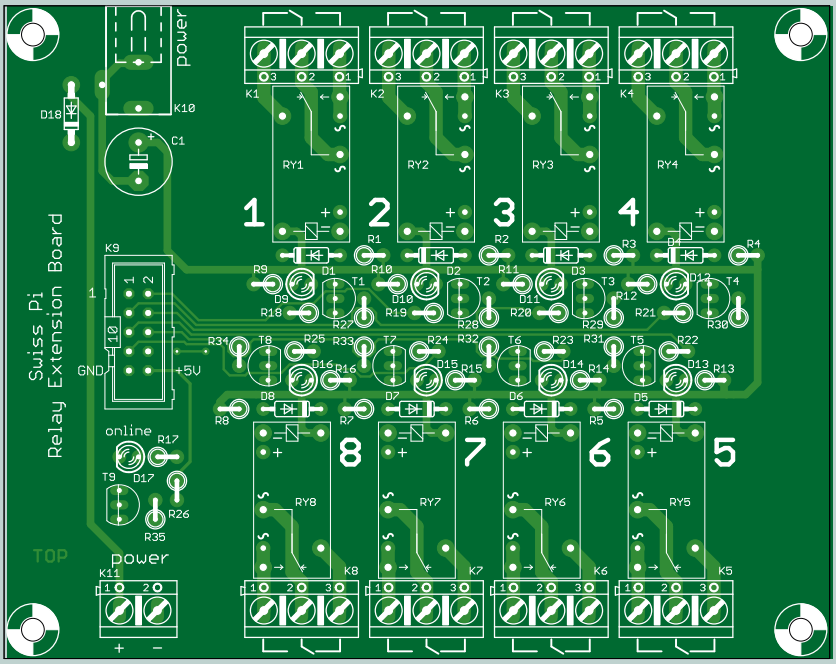
COMPONENT LIST FOR RELAY BOARD

Resistors
R1–R8 = 560Ω*
R9–R17 = 2.2kΩ
R18–R26 = 4.7kΩ
R27–R35 = 100kΩ

Capacitors
C1 = 100µF/16V

Semiconductors
D1–D8 = 1N4148**
D9–D17 = LED, 3mm, red
D18 = 1N4007
T1–T9 = BC547B

Miscellaneous
RY1–RY8 = G5Q-14-EU 12DC or S202S02***
K1–K8 = 3-way PCB screw terminal block, 0.2" pitch
K9 = 10-pin boxheader
K10 = DC power connector, coaxial
K11 = 2-way PCB screw terminal block, 0.2" pitch



*Mount when using an S202S02 solid state relay
**Mount when using a G5Q relay
***Mount per channel as desired; for the G5Q relays a different supply voltage can be chosen if necessary. In that case it may be necessary to alter the values of R1–R17 and C1.

when the supply voltage is present on the Swiss Pi board.
If desired, you can also use G5Q relays with a different supply voltage rating, such as 24 V. However, in that case a number of resistor values will have to be changed.

Eight-channel digital input board

This extension provides eight galvanically isolated inputs with an input volt-

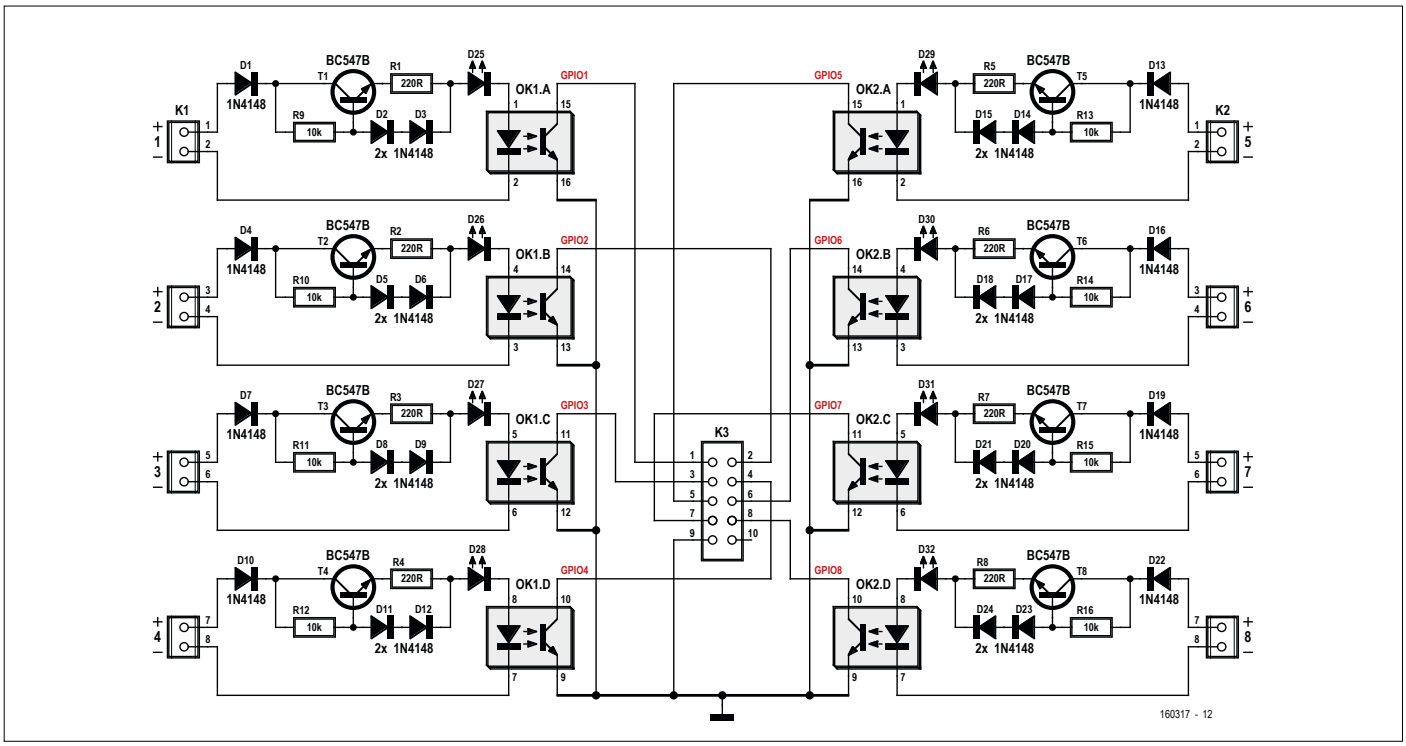


Figure 2. Eight galvanically isolated digital inputs are implemented with optocouplers.

age range of 5 to 45 V (see **Figure 2**). Galvanic isolation is provided by optocouplers. On the input side the current sources built around the BC547 transistors keep the current through the LEDs of the optocouplers and the indicator LEDs more or less constant, regardless of the input voltage. This current is equal to the forward voltage drop of two 1N4148 diodes minus the base-emitter junction voltage of the transistor, divided by 220 (the resistance value of R1–R8). The current through the 1N4148 diodes ranges from a few microamps to several milliamps over the input voltage range of 5 to 45 V. If you look at the characteristic curve of the 1N4148, you can see that this is precisely the region where the forward voltage is most strongly dependent on the forward current. As a result, the current supplied by the current sources is not truly constant over the entire input voltage range. In practice it ranges from 3 to 6 mA, depending on the input voltage.

If the voltage over the current source is reversed, the collector-emitter voltage of the transistor will be negative. However, the transistor will continue to operate as a low-grade NPN transistor with very low gain and a low reverse breakdown voltage. If the magnitude of the negative collector-emitter voltage rises above 6 V, the

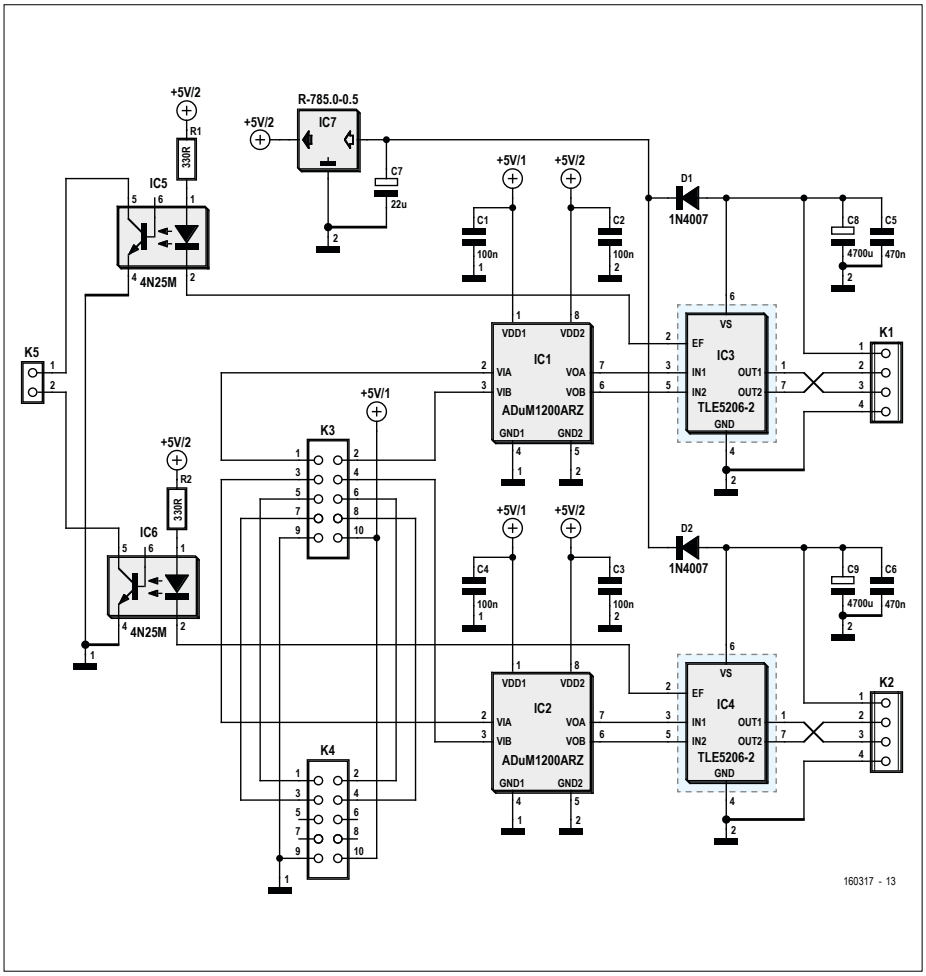


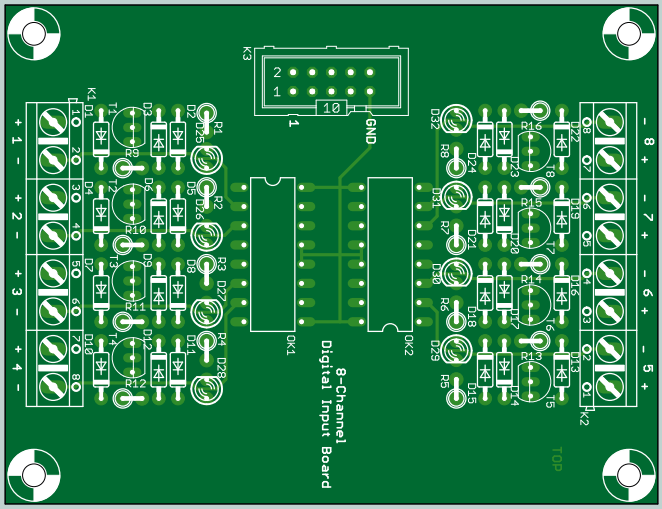
Figure 3. The galvanically isolated motor controller boards can be daisy-chained to drive up to eight DC motors (two per board).

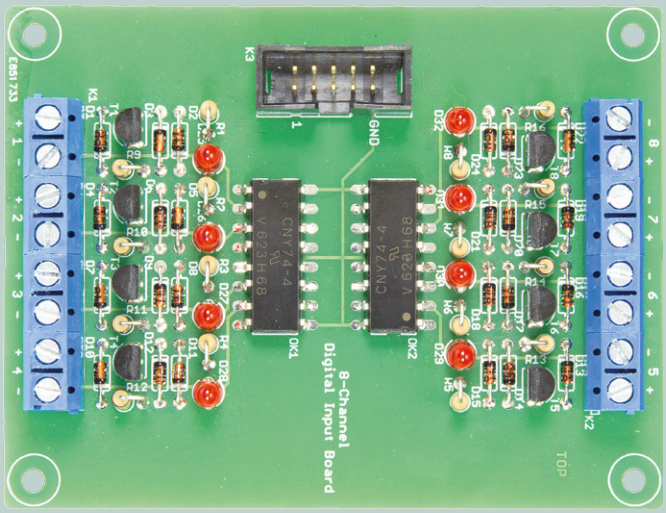
COMPONENT LIST FOR DIGITAL INPUT BOARD

resistors
R1–R8 = 220Ω
R9–R16 = 10kΩ

Semiconductors
D1–D24 = 1N4148
D25–D32 = LED, 3mm, red
T1–T8 = BC547B
OK1,OK2 = CNY74-4

Miscellaneous
K1,K2 = 8-way PCB screw terminal block, 0.2" pitch
K3 = 10-pin boxheader





transistor will start conducting and the current will only be limited by the 220 Ω resistors and the leakage currents of the indicator and optocoupler LEDs. To avoid potential damage to the LEDs, particularly with relatively high input voltages, an additional 1N4148 diode is included to provide reverse-polarity protection.
In the prototype we used red LEDs as indicators, but you can also use other colors. Bear in mind that this will have some effect on the input voltage range, especially the minimum input voltage.
On the Swiss Pi side the optocouplers have open-collector outputs. This means that the GPIO pins must be configured as inputs with pull-up resistors. When a voltage is applied to an input on the extension board, the corresponding GPIO input is pulled low (inverse logic).

DC motor controller

The DC motor controller board (**Figure 3**) can drive two DC motors independently, with a maximum operating voltage of 40 V and a maximum current of 5 A. The output stage of this board consists of two TLE5206-2 MOSFET H-bridge ICs. These ICs are especially easy to control and are protected against overcurrent and shorted outputs. The drain-source resistance $RDS_{(on)}$ is 0.2 Ω, so the power dissipation remains within reasonable limits. A drawback of this IC is that the switching time ("source on" time) can be as much as 15 µs, which is fairly slow. However, that is not a significant problem because the PWM controller of the Swiss Pi has a maximum upper frequency of about 1.5 kHz. This also means that the drive frequency is audible on the DC motor, especially at low PWM values.

To prevent noise and harmonics on the motor supply voltage rails from reaching the Swiss Pi and the Raspberry Pi, galvanic isolation is provided between the H-bridges and the Swiss Pi. For the PWM signals this is done using type ADUM1200ARZ digital isolators. These devices have two channels and much wider bandwidth than optocouplers. Unfortunately, they are not available in a through-hole version, but the 8-pin SOIC package is relatively easy to solder by hand. Conventional optocouplers are used for the optional feedback of error flag signals from the H-bridges.

The dual motor controller boards can

COMPONENT LIST FOR DC MOTOR CONTROLLER

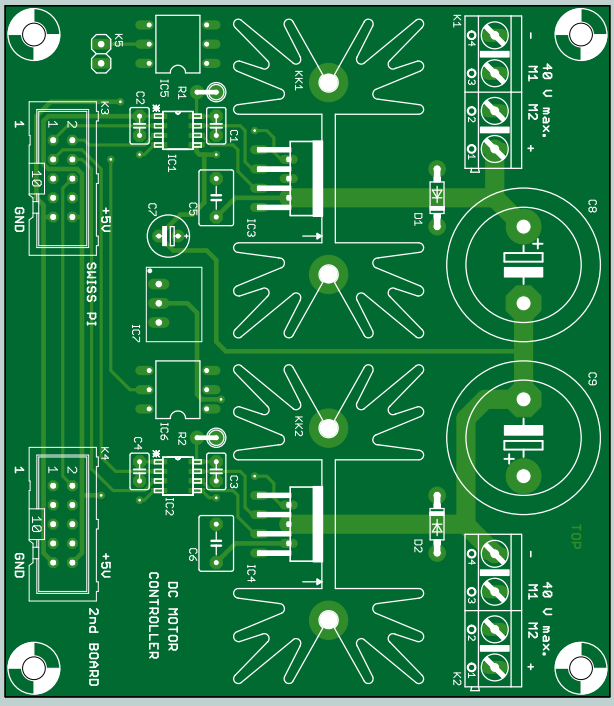
Resistors
R1,R2 = 330Ω

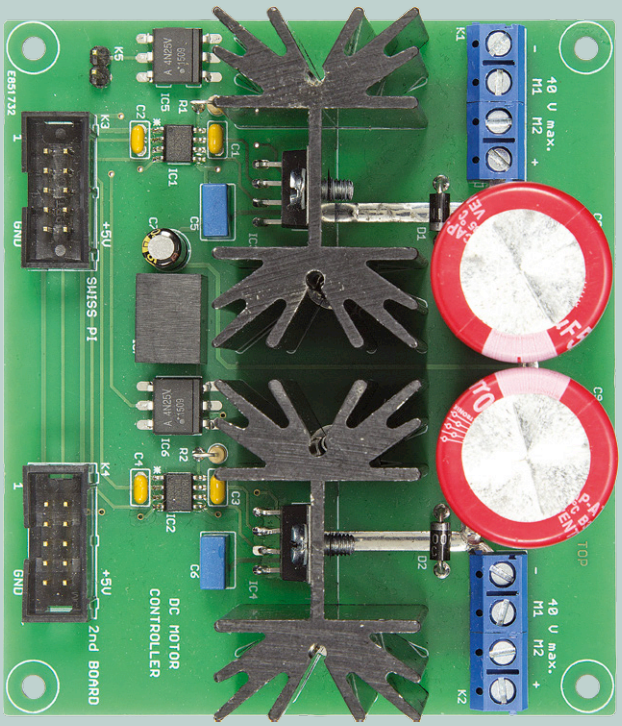
Capacitors
C1–C4 = 100nF
C5,C6 = 470nF/63V*
C7 = 22µF/50V*
C8,C9 = 4700µF/50V*

Semiconductors
D1,D2 = 1N4007
IC1,IC2 = ADUM1200ARZ
IC3,IC4 = TLE5206-2
IC5,IC6 = 4N25
IC7 = DC/DC converter SIP3, 5V/1A, 40Vin (e.g. Würth 173010542)*

Miscellaneous
K1,K2 = 4-way PCB screw terminal block, 0.2" pitch
K3,K4 = 10-pin boxheader
KK1,KK2 = heatsink, e.g. SK 129 38,1 STS from Fischer Elektronik (38.1 x 42 x 25 mm)

*These component values can be adjusted if the motor controller is used with voltages lower than 40 V. IC7 does not need to supply 1 A, but all available DC/DC converters with a relatively high input voltage rating (over 28 V) are 1 A types.





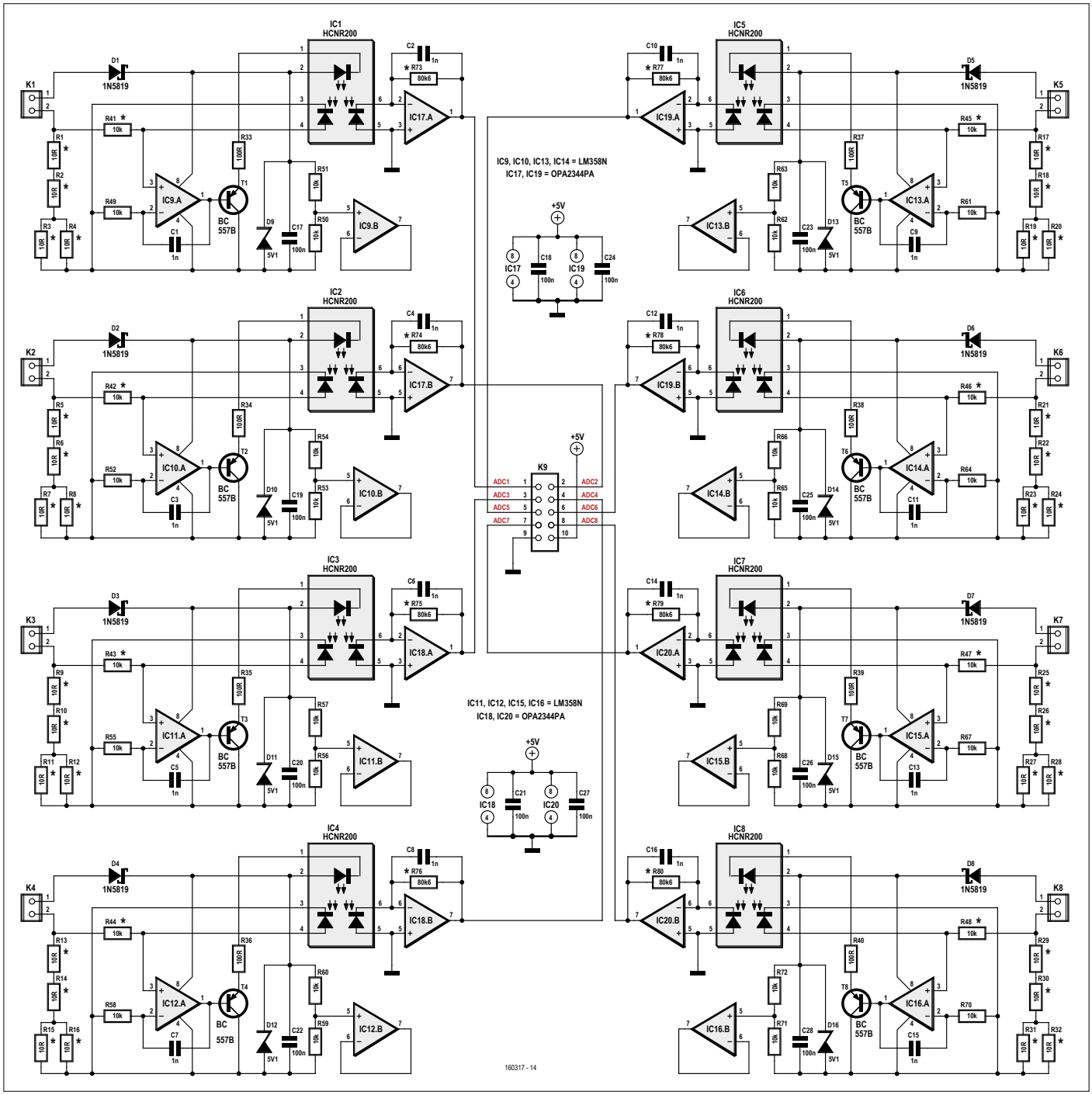


Figure 4. The schematic of the current loop interface board looks fairly large, but it actually consists of eight identical circuits.

be daisy-chained to enable a total of eight DC motors to be controlled with the Swiss Pi.

Finally, a DC/DC converter provides the 5 V supply voltage for the logic circuitry on the H-bridge side. If you want to use relatively high supply voltages, it is essential to choose a DC/DC converter that can withstand a 40 V input voltage.

Current loop interface

Current loop interfaces are still widely used in the industrial sector for reading sensors, despite the advent of digital alternatives. Current loops are simple and robust, and they have a long working range. Consequently, there are a lot of sensors available which use this technology.

The interface described here (see **Figure 4**) is built around HCNR200 linear optocouplers from Avago. The internal structure of these devices consists of two matched photodiodes and an LED. At the transmit end, one photodiode is used to provide feedback from the power emitted by the LED. The overall transmission characteristic is virtually linear because the current through the photodiode at the receive end is nearly the same as the current through the photodiode at the transmit end.

For this circuit we used a fairly direct copy of the typical application circuit on the Avago data sheet, with only a few minor changes. For instance, we replaced the LM158 opamps at the transmit end

COMPONENT LIST FOR CURRENT LOOP INTERFACE

Resistors

R1–R32 = 10Ω*
R33–R40 = 100Ω
R41–R48 = 10kΩ*
R49–R72 = 10kΩ
R73–R80 = 80.6kΩ*

Capacitors

C1–C16 = 1nF
C17–C28 = 100nF

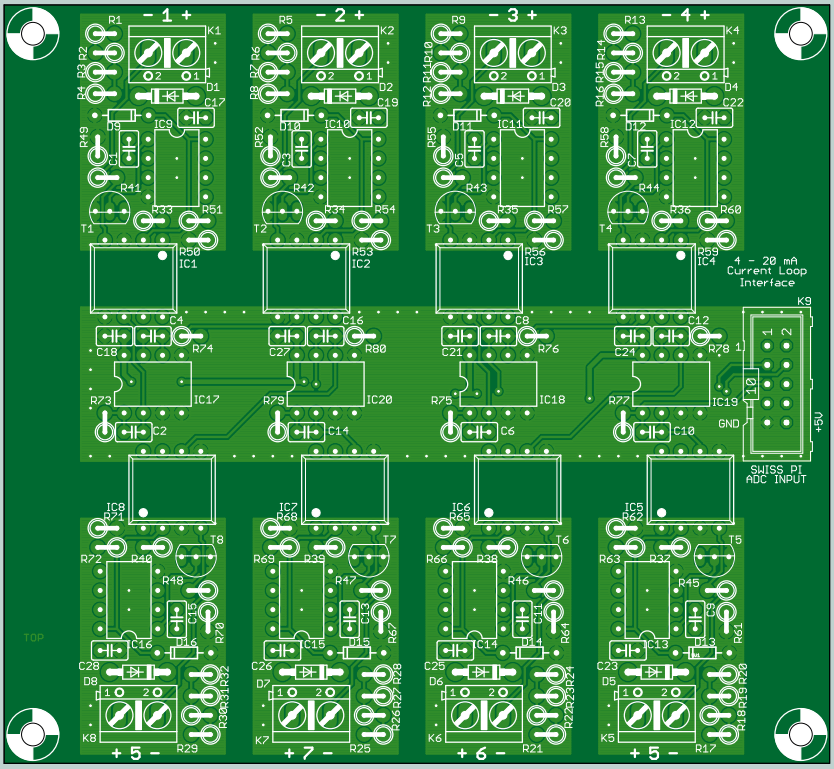
Semiconductors

D1–D8 = 1N5819
D9–D16 = Zener diode 5.1V, 400mW
IC1–IC8 = HCNR200
IC9–IC16 = LM358N
IC17–IC20 = OPA2344PA (LM6142BIN/NOPB)

Miscellaneous

K1–K8 = 2-way PCB screw terminal block, 0.2" pitch
K9 = 10-pin boxheader

*1% metal film or better



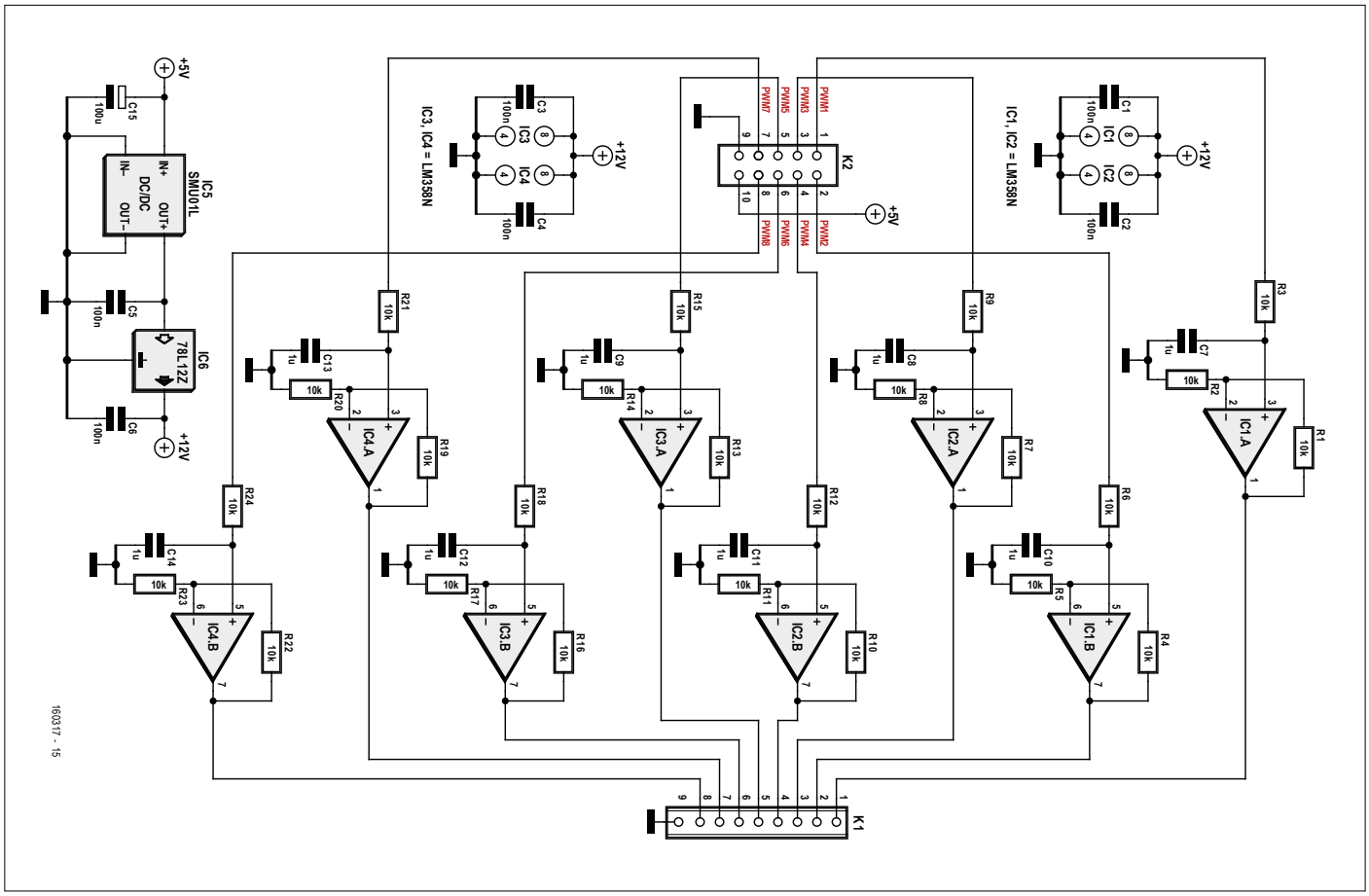
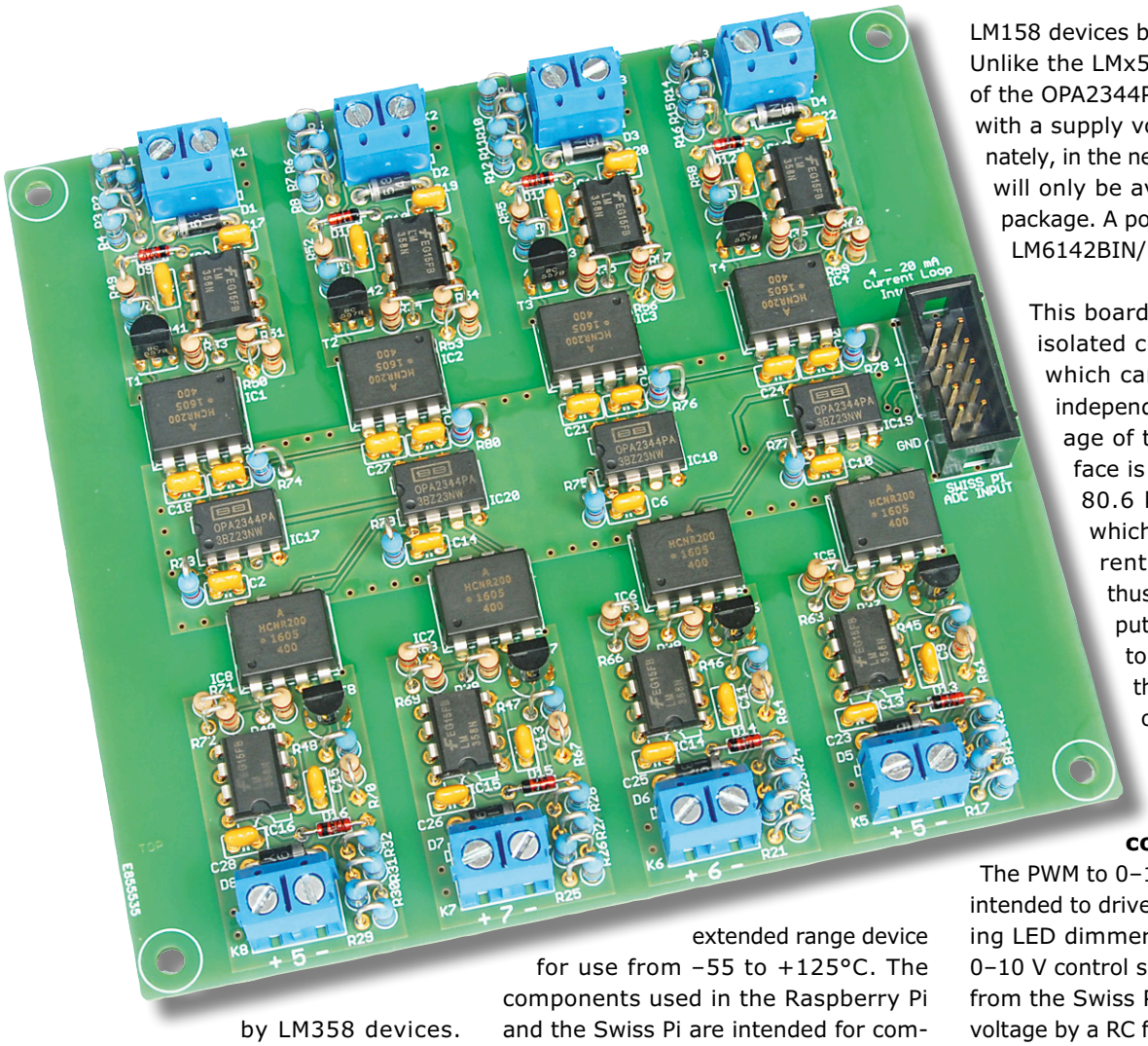


Figure 5. The PWM to voltage converter board generates 0 to 10 V voltage outputs from PWM signals.



by LM358 devices. They cost a lot less and are more readily available than the LM158. The main difference is in the rated temperature range of the opamps: the LM158 is an extended range device for use from –55 to +125°C. The components used in the Raspberry Pi and the Swiss Pi are intended for commercial use with a temperature range of 0 to 70°C. In that situation there is not much reason to use the LM158.

At the receive end we replaced the LM158 devices by OPA2344PA opamps. Unlike the LMx58 opamps, the output of the OPA2344PA can easily reach 4 V with a supply voltage of 5 V. Unfortunately, in the near future these opamps will only be available in the SOIC-8 package. A possible alternative is the LM6142BIN/NOPB.

This board provides galvanically isolated current loop interfaces which can be used completely independently. The output voltage of the current loop interface is equal to I_{in} × (25 Ω × 80.6 kΩ)/(10 kΩ + 25 Ω), which is I_{in} × 201. The current range of 4 to 20 mA thus corresponds to an output voltage range of 0.804 to 4.02 V, which is within the A/D conversion range of the Swiss Pi.

PWM to voltage converter

The PWM to 0–10 V converter board is intended to drive light dimmers (including LED dimmers), most of which use 0–10 V control signals. The PWM signal from the Swiss Pi is converted to a DC voltage by a RC filter, and this voltage is then buffered and doubled by the opamps (see **Figure 5**).

The current consumption of the opamps

COMPONENT LIST FOR PWM TO VOLTAGE CONVERTER

Resistors

R1–R24 = 10kΩ

Capacitors

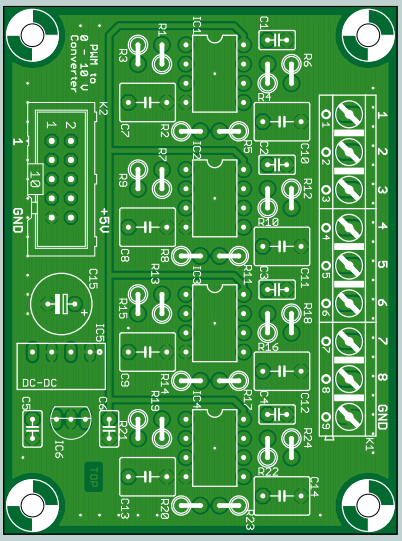
C1–C6 = 100nF
C7–C14 = 1µF
C15 = 100µF

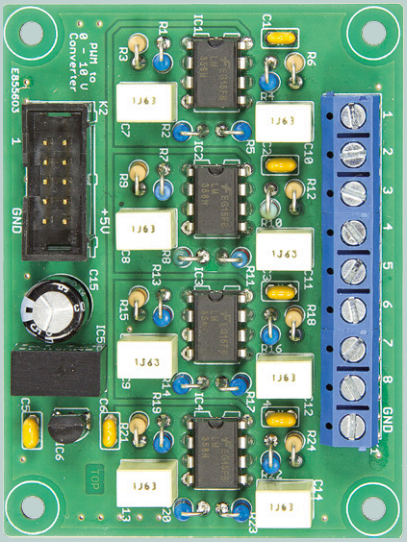
Semiconductors

IC1–IC4 = LM358N
IC5 = SMU01L-15
IC6 = 78L12Z

Miscellaneous

K1 = 9-way PCB screw terminal block, 0.2" pitch
K2 = 10-pin boxheader





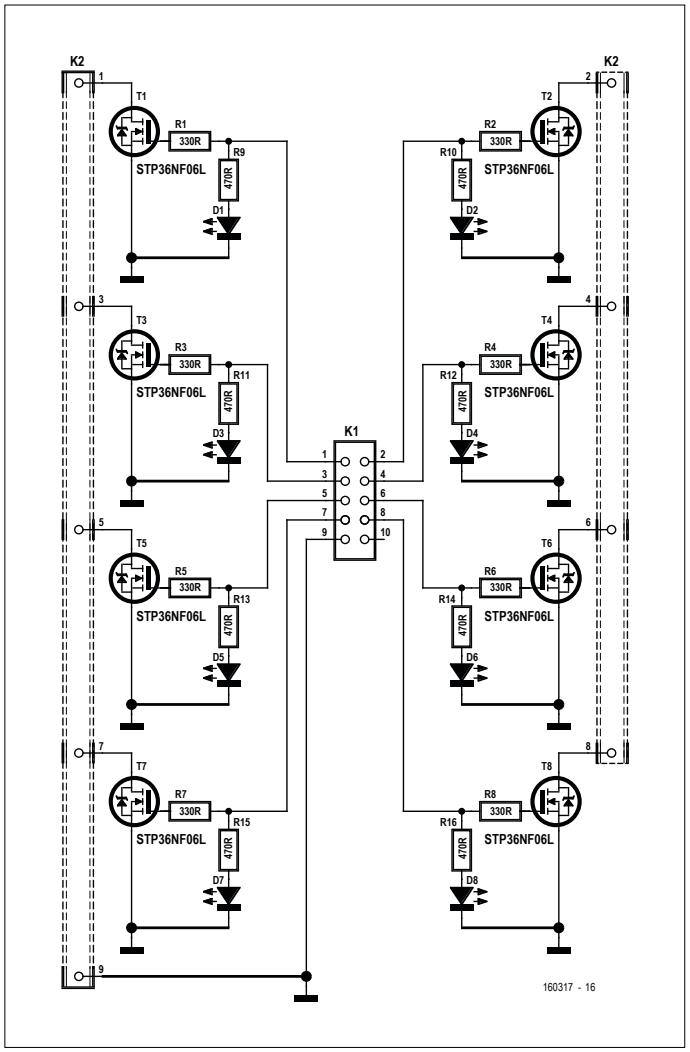


Figure 6. The MOSFET board is populated with STP36NF06L devices, which can handle 30 A and a maximum voltage of 60 V.

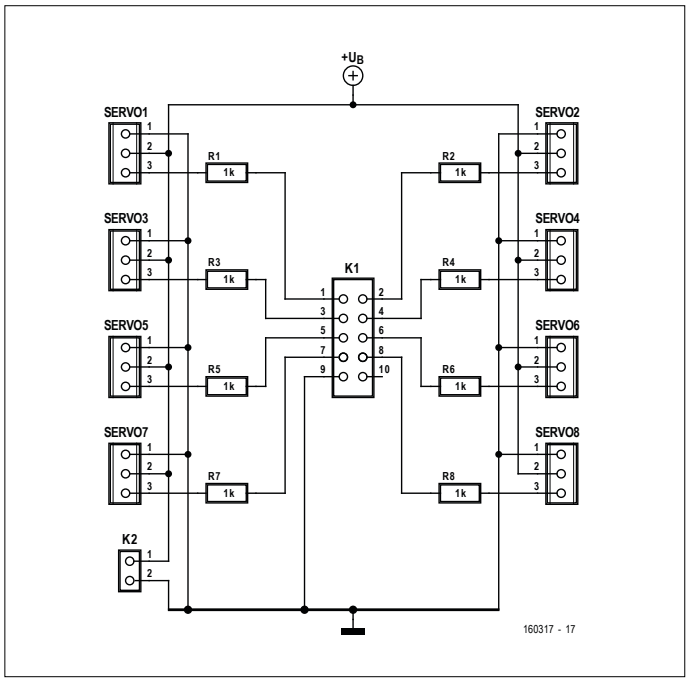


Figure 7. With this simple servo motor board you can control up to eight servo motors from the Swiss Pi.

COMPONENT LIST FOR MOSFET BOARD

Resistors

R1–R8 = 330Ω

R9–R16 = 470Ω

Semiconductors

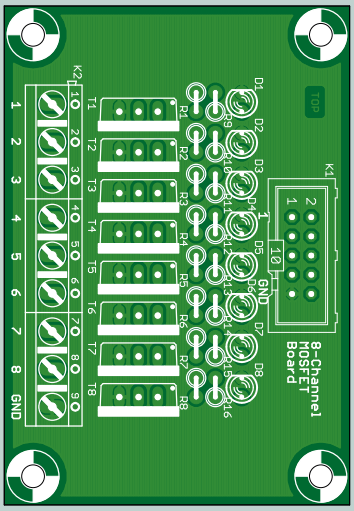
D1–D8 = LED, 3mm, red

T1–T8 = STP36NF06L

Miscellaneous

K1 = 10-pin boxheader

K2 = 9-way PCB screw terminal block, 0.2" pitch



is low and the load on the 0–10 V signals is usually low, so a simple DC/DC converter is used to provide the higher supply voltage for the opamps. It converts the 5 V supply voltage from the Swiss Pi to 15 V, which is then reduced to a regulated 12 V by a 78L12. That eliminates the need for an external power source.

You should bear in mind that the output voltage range may not be exactly 0 to10 V — in practice it extends from 0 V to twice the supply voltage of the PWM IC on the Swiss Pi board.

MOSFET board

The MOSFET board (**Figure 6**) can be used to switch DC loads. The normal GPIO lines of the Swiss Pi or the PWM channels can be used for this purpose. Driving RGB LED strips is a typical application.

COMPONENT LIST FOR SERVO MOTOR BOARD

Resistors

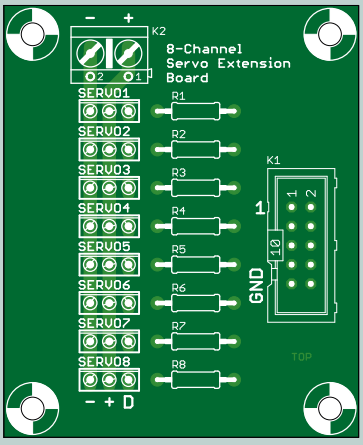
R1–R8 = 1kΩ

Miscellaneous

K1 = 10-pin boxheader

K2 = 2-way PCB screw terminal block, 0.2" pitch

K3–K10 = 3-pin header, 0.1" pitch



For the MOSFETs we chose type STP36NF06L devices. That is an N-channel MOSFET which is suitable for voltages up to 60 V with a rated current of 30 A ($R_{DS(on)}$ = 0.032 Ω at V_{GS} 10 V / 15 A). The STP36NF06L is a logic-level MOSFET, which means it can be driven fully on even with a gate voltage less than 5 V. With a V_{GS} of 5 V the on resistance $R_{DS(on)}$ is 0.045 Ω, which in practice means that you can switch about 5 A per channel without needing a heat sink for the MOSFET. Each channel also has an indicator LED.

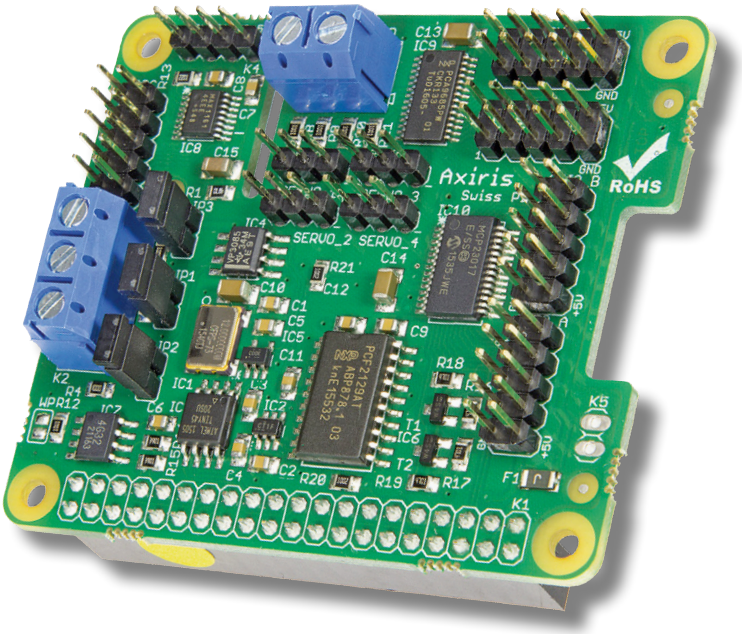
Servo motor extension

With the servo motor extension board (**Figure 7**) you can connect more servo motors to the Swiss Pi than the four supported directly. With two of these boards you can control a maximum of sixteen servo motors.

The resistors protect the Swiss Pi in the event that the three-pin plug of a servo motor is accidentally connected the wrong way.

In combination with the previous article and the many software examples described there [3], you now have an arsenal of tools to tackle a wide range of applications with the Swiss Pi / Raspberry Pi duo.

(160317)



Web Links

[1] www.elektormagazine.com/150584

[2] www.elektormagazine.com/160317

[3] www.elektormagazine.com/160237

IN THE STORE

→ **SKU 17713** Swiss Pi (150584-91))

→ **SKU 17631** Raspberry Pi 3

Advertisement

mouser.com

The Newest Products for Your Newest Designs®

The **widest selection** of the **newest products.**

Over **4 Million** products from over **600 manufacturers.**

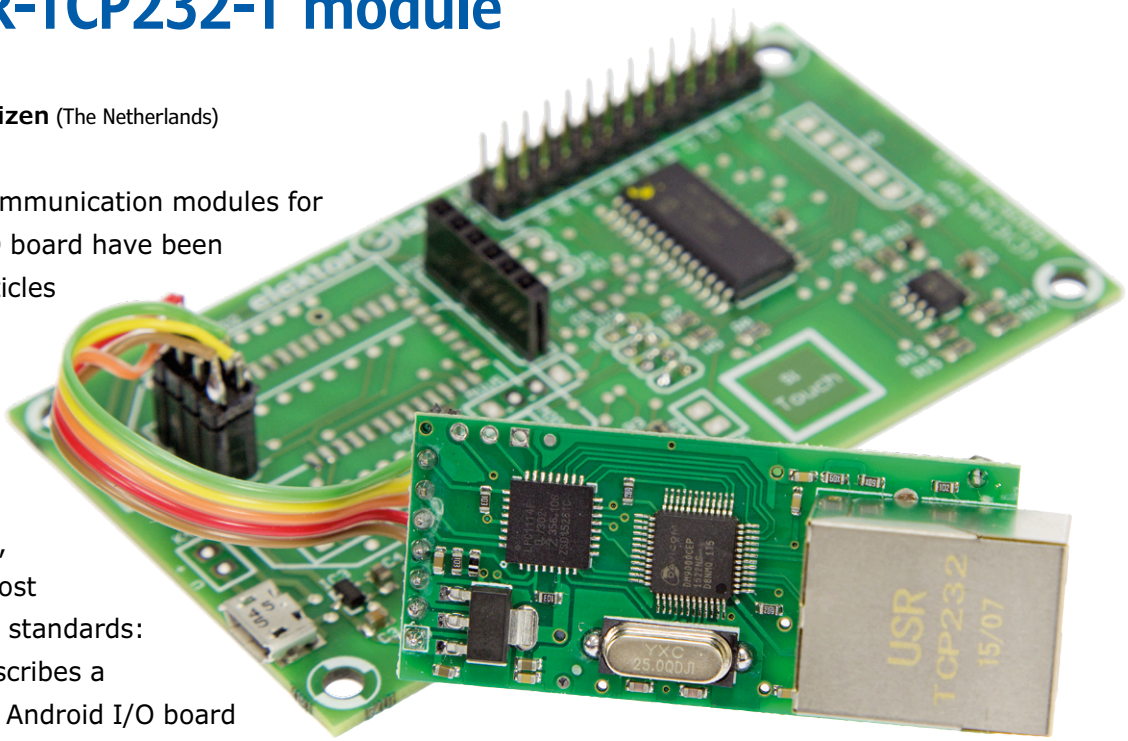
MOUSER ELECTRONICS

Authorised distributor of semiconductors and electronic components for design engineers.

Ethernet on the Android I/O Board

Using a USR-TCP232-T module

By **Elbert Jan van Veldhuizen** (The Netherlands)



A number of different communication modules for use with the Android I/O board have been introduced in various articles for the board. These modules have used Wi-Fi, Bluetooth or USB. However, there is one module for the Android I/O board that's missing, which uses one of the most popular communications standards: Ethernet. This article describes a module that enables the Android I/O board to be connected to a wired Ethernet network.

The USR-TCP232-T module [1] is one of the cheapest modules around that can convert data from an Ethernet connection into serial data, which can be processed by the Android I/O board. The advantage of an Ethernet module is that it doesn't use radio waves for the connection, which means that any interference in that area is avoided.

The USR-TCP232-T module can operate in one of four modes [2]. The main modes are a server mode and a client mode. In server mode the app has to create the connection with the USR-TCP232-T module. In client mode the module works the other way round: the module takes the initiative to create a connection to the app. For the Android I/O board we have to configure the USR-TCP232-T as a server: The standard software for the apps always takes the initiative to create the connection. In both the server mode and client mode you can choose between a UDP and TCP connection. With a UDP connection the data is transmitted without any checks taking place that the data has been received successfully. With a TCP connection there are checks to see that the data has been received (and in the correct order); if necessary, the data will be retransmitted. The standard software for the Android I/O board uses TCP. For the Android I/O board we therefore use the 'TCP server' mode for the USR-TCP232-T module.

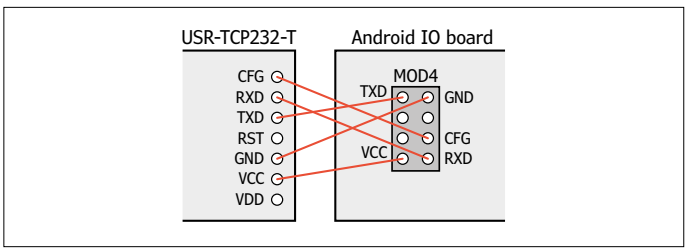


Figure 1. The wiring diagram for the two connectors on the cable linking the USR-TCP232-T module to the Android I/O board.

The USR-TCP232-T module doesn't support DHCP. With DHCP, the router allocates IP addresses to the devices on the IP network, such as PCs or mobile phones. With the USR-TCP232-T module we can only set up a fixed (static) IP address. This address should be within the IP range of the (private) network that the module is connected to.

If we want to connect a USR-TCP232-T module to the Android I/O board, we must first take care of two things. First of all, we need to make a connection cable to connect the module

to the Android I/O board. We then have to set up the serial connection and the IP address of the USR-TCP232-T module.

Connecting to the Android I/O board

There is no single connector on the Android I/O board that exactly matches the one on the USR-TCP232-T module. However, all connectors do have the links we need: the supply voltage (3.3 V and GND) and the serial data (TX and RX). The USR-TCP232-T module also has a configuration pin. This is only required when we want to configure the module via the serial port. Since we're going to configure the module via the Ethernet network, this pin isn't really required.

As we saw earlier, we can use any of the connectors on the board. The author decided to use the connector for the ESP8266 (MOD4). There are two ways in which to make the connector: via a ribbon cable or using a (double-sided) experimenter's board with the connectors for the USR-TCP232-T module and the MOD4-connector mounted on it. **Figure 1** shows the wiring diagram. Note that the two pins either side of the Ethernet connector on the USR-TCP232-T module are not electrically connected to any of the parts on the module; they are there to provide mechanical stability to the module.

Setting up the USR-TCP232-T

To set up the USR-TCP232-T, we can download a Windows program from the manufacturer's website [3]. This program sends a special UDP packet containing all parameters to the module in order to change its settings [2].

Below is a comprehensive set of instructions on how to set up the USR-TCP232-T (also see **Figure 2**). But first you must determine which addresses you can use in your network and what its subnet mask is. On a Windows PC you type the command 'ipconfig' in a 'DOS box' (this is opened by typing 'cmd' from the start menu). You will then see the IP address of the PC and the subnet mask. Choose a higher address within the subnet; as an example, when the subnet mask is 255.255.255.0 you could use .240 as the last number for the IP address. This address won't be in use, unless you have hundreds of devices connected in your subnet. Once the router has spotted that this address is in use, it won't allocate this address to other devices. The DHCP server in the router can also be configured so that it won't dynamically allocate all addresses. The author has configured his router so that the DHCP server will only allocate addresses from .10 to .100. The other addresses can then be statically configured in the peripheral devices.

You can use any port number you like. 'Logical' choices are '23' (the port for Telnet), '2000' (the port used by default by the RN-171) or '20108' (the default port of the USR-TCP232-T module).

In the following example the subnet mask is '255.255.255.0' and the module gets the IP address of 192.168.178.120 with a port at 20108. The gateway (router) has the address 192.168.178.1. The MAC address is different for every module, so the MAC address used here is just an example. Follow these steps:

- Connect the USR-TCP232-T with an Ethernet cable to the network to which the PC with the configuration program is also connected.

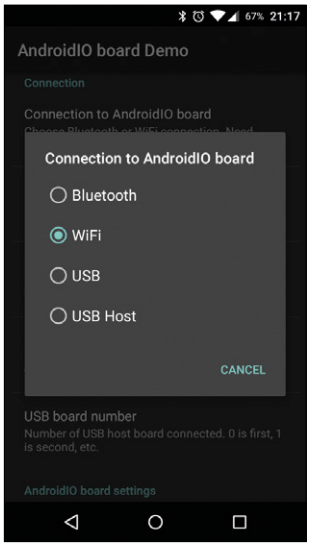


Figure 3. In de demo-app 'Android IO board Demo' you have to select Wi-Fi for the setting 'Connection to Android I/O board'. In this case it really means 'IP network'.

- Connect the power to the module. Don't use the configuration pin on the module (don't connect it to anything at all).
- Start the configuration program on the PC (USR-TCP232-T24 V5.1.0.1.exe).
- Click on 'Search in LAN'. The program will then find the module on the network.
- The module will appear in the 'Device list in the net'. Select the module by clicking on it.
- Select the following settings:
  - Module work mode: TCP server
  - Module IP: 192.168.178.120
  - Subnet mask: 255.255.255.0
  - Default gateway: 192.168.178.1
  - Baud Rate (bps): 9600
  - Parity/Data/Stop: NONE, 8, 1
  - Module port: 20108
  - 'Destination IP' and 'Destination port' are grey and can't be filled in.
- Click on 'Set selected item via LAN'
- The following then appears in the Logs: "The param of Device which MAC is 00C18B5C42E1 set OK, You can search for new setting later."

The module is now ready for use.

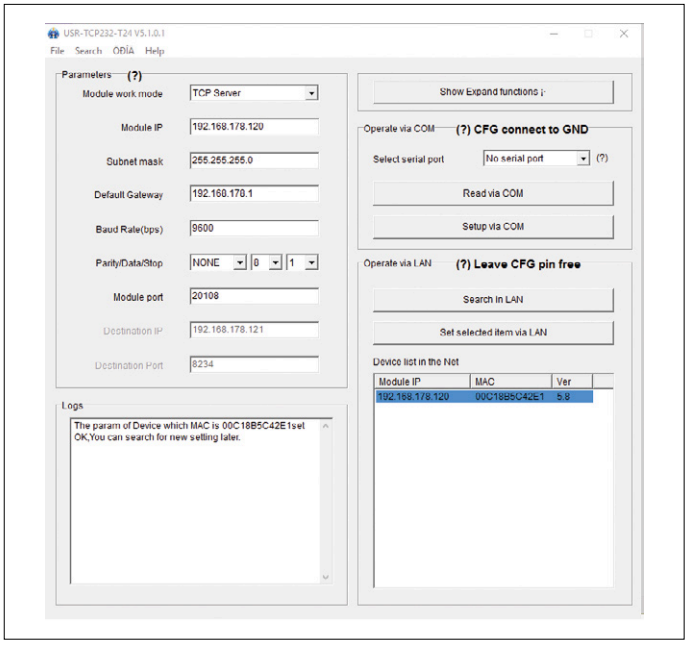


Figure 2. Screen dump of the 'USR-TCP232-T24 V5.1.0.1.exe' program, with the settings shown that are used to configure the module.

Setting up the app

To get the app on an Android phone to connect to the USR-TCP232-T module, we have to select WiFi (in the 'Android I/O board Demo' app, for example) for the setting in 'Connection to Android I/O board' (**Figure 3**). For the 'WiFi address' and 'WiFi port' you should type in the IP address and the port number of the USR-TCP232-T module. In this case, 'WiFi' should be considered to be 'IP network'.

Controlling several Android I/O boards simultaneously

It is possible to control several Android I/O boards from a single app. This can also include a mix of different communications methods: IP (Wi-Fi and Ethernet), Bluetooth and USB (see **Figure 4**). There isn't really a limit to the number of Android I/O boards you could control! With IP networks in particular, it's possible to have the Android I/O boards very far apart from each other. Despite this, they can still be controlled and interrogated by a single app.

When multiple Android I/O boards are controlled, a separate object from the IOFunction class has to be created for each board. We can also create an array of objects, if that is easier in your code. For example:

```
IOBoardFunctions[] IOBoard = new IOBoardFunctions[4];
for (int i=0; i<4; i++) { IOBoard[i] = new
      IOBoardFunctions(); }
```

The choice of object determines which Android I/O board will be used for the initializing of the Android I/O board, making a connection or the transmission of data. For example:

```
IOBoard[2].initiate(2, hnd, "192.168.178.120", 23, 1,
      this);
```

This ensures that the object IOBoard[2] will be linked to the Android I/O board with the specified IP address and port number. The data sent back from all the different Android I/O boards can be processed by a single handler. The first parameter passed with the .initiate (in this example it is '2'), acts as a channel number. It is included with all data in the '.arg1' field of the Message object [4]. This way the app will know exactly from which Android I/O board the data comes from. The figure below shows this graphically.

There is an accompanying demo app (including source code) with this article, which simultaneously reads the temperature sensors of four Android I/O boards. From the menu you can select the communications method for each of the Android I/O boards independently. The app then reads the temperature sensors once per second.

In the source code you can see exactly how multiple Android I/O boards can be controlled. This code and the apk file can be downloaded from [5].

(150804)

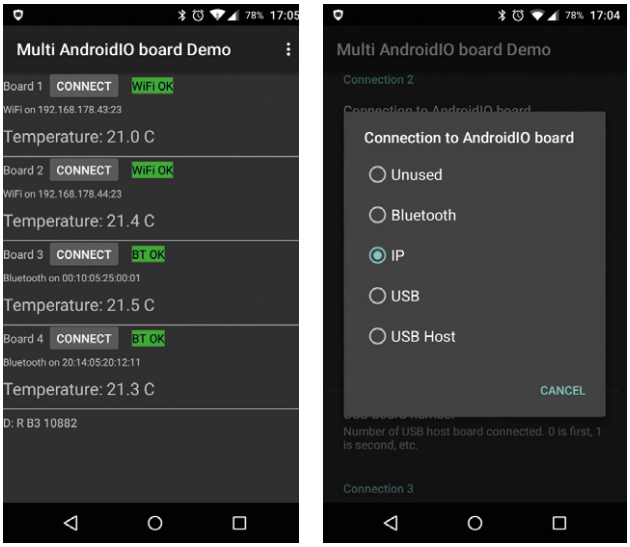


Figure 4. In the 'Multi AndroidIO board Demo' you can see that four different boards are connected, where the temperature of each one is shown. For each board you can select a different connection method.

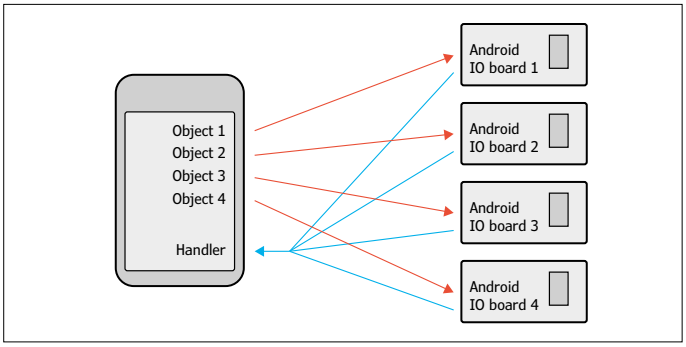


Figure 5. All the data sent back by the Android I/O is processed by a single handler.

New version of IOBoardFunctions

With the demo app is a new version of IOBoardFunctions, the class containing all the libraries for controlling the Android I/O board from Android. There are several improvements in this version:

All IOBoardFunctions are now in a single file, instead of five files. This makes it easier to use the IOBoardFunctions in your own project.

You can now change the IP address without having to close the app. In the app you can even change the communications method, from IP to USB or Bluetooth, for example.

The number of different '.initiate' functions has been streamlined. There is now a single function for apps that offers all of the communications methods, as well as separate functions for each of the communications methods (Bluetooth, IP, USB Accessory and USB Host).

A detailed explanation can be found at the start of the file IOBoardFunctions.java.

Web Links

[1] Product page: www.usriot.com/Product/20

[2] Manual for the USR-TCP232-T: www.usriot.com/download/T24//USR-TCP232-T24-EN%20V3.2.5.pdf

[3] Configuration software: www.usriot.com/download/software/USR-TCP232-T24V5.1.1.20.rar

[4] Description of the Message class: http://developer.android.com/reference/android/os/Message.html

[5] www.elektormagazine.com/150804

reichelt elektronik

Technology connects.

- More than 45 years of experience
- 24-hour shipping
- More than 70,000 products



Foto: Foto- und Bilderwerk

V-TAC® INNOVATIVE LED LIGHTING

PRICE HIT

VT-7126

0.50 (€ 0,58)

A+

LED BULB, GU 10

- Output: 3 watts
- Luminous efficiency: 210 lm
- Light colour: warm white
- Life: 20,000 h
- Angle of radiation: 110°
- CRI (Ra): 80

REPLACE LIGHTS NOW AND SAVE!

LED STRIP

- Output: 18 watts
- Angle of radiation: 120°
- Length: 5 m
- Width: 8 mm
- Flexible, easy installation
- CRI (Ra): 80



A

each **3.90**

VT-2016	3.90 (€ 4,50)	warm white
VT-2041	3.90 (€ 4,50)	neutral white
VT-2005	3.90 (€ 4,50)	cold white

LED BULB, E 27

These LED bulbs can be used to replace nearly every energy-saving bulb and conventional light bulb in an E27 socket. So you can start saving energy right away.

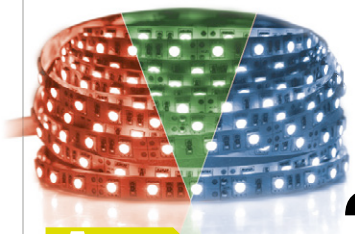
- Output: 10 watts
- Luminous efficiency: 806 lm
- Light colour: warm white
- Life: 20,000 h
- CRI (Ra): 80



A+

VT-4209

1.23 (€ 1,42)



LED STRIP

- Output: 18 watts
- Angle of radiation: 120°
- Length: 5 m
- Width: 8 mm
- Flexible, easy installation

A

from **3.90**

VT-2015	3.90 (€ 4,50)	red
VT-2011	3.90 (€ 4,50)	green
VT-2013	3.90 (€ 4,50)	blue
VT-2124	5.79 (€ 6,68)	RGB

MORE LED LIGHT TECHNOLOGY

The articles in this ad and many others are available in our online shop:
http://**rch.lt/zW**

Daily prices! Price as of: 31. 1. 2017

Prices in £ plus statutory VAT, plus shipping costs · reichelt elektronik, Elektronikring 1, 26452 Sande (Germany)

Onlineshop languages:

PAYMENT METHODS: VISA MasterCard PayPal

SHOP CONVENIENTLY ONLINE!

www.reichelt.co.uk

ORDER HOTLINE: +49 (0)4422 955-360

HOMELAB PROJECT

Build Your Own Theremin

Using JFETs instead of vacuum tubes

Although the theremin was invented nearly 100 years ago, the instrument still has enthusiastic fans. It is played entirely without physical contact, and almost everyone will have heard one at some point. The original design from 1920, by Lev Termen, alias Léon Theremin, used vacuum tubes, but JFETs can equally well be employed. In this article we look at the development of an experimental design for a theremin that you can build yourself.

Von **Burkhard Kainka** (D)



The principle of operation of the theremin is straightforward and is based on the mixing of two RF signals to produce a beat frequency. One RF oscillator, called the 'pitch oscillator', has its frequency tuned by variations in the capacitance to the thereminist's hand thus adjusting the pitch of the note being played, while the other operates at a fixed frequency. The outputs of the two oscillators are taken to a demodulator which works on the same principle as an AM radio. Overall the circuit is very similar to a receiver for unmodulated Morse signals. For simplicity, then, our first experiment will use a ready-made radio receiver.

First experiments using the Elektor SDR shield

For our first experiments we will only build a single oscillator, whose frequency we can pull to a sufficient extent using the capacitance of a hand. To set this up we will use the Elektor SDR shield connected to an Arduino Uno running the standalone VFO software, along with an LCD panel on an Elektor extension shield. An arrangement like this has been described previously in Elektor for use at shortwave frequencies [1]. Here we use the shield as a direct-conversion mixer and we connect its output to an active loudspeaker, thus replacing the second oscillator and the mixer in the original theremin. It is easy to set the frequency of the beat oscillator very precisely (see **Figure 1**) and all we need now is the pitch oscillator. That will give use our first simple theremin, albeit without a volume control: an interesting application of the SDR shield.

The FET oscillator circuit shown in **Figure 2** operates at 420 kHz; moving your hand near to it will pull its frequency by up to 5 kHz. The antenna for the SDR is a length of wire simply placed close to the oscillator. The signal is so strong that only a weak coupling is required to the receiver. Instead of connecting a PC to the stereo output, we wire it directly to an

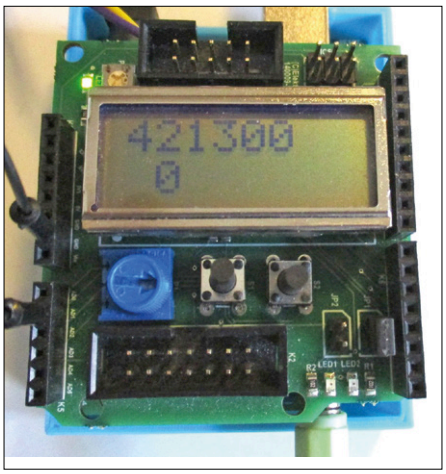


Figure 1. The receiver tuned to 421.3 kHz.

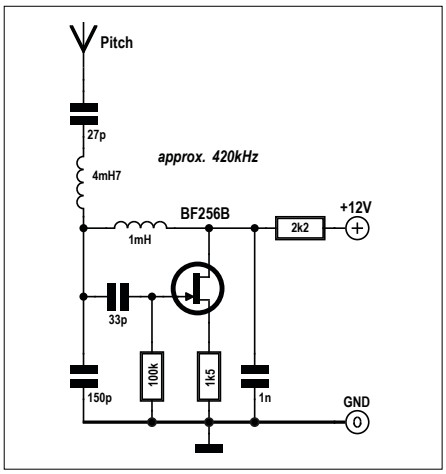


Figure 2. A simple pitch oscillator.

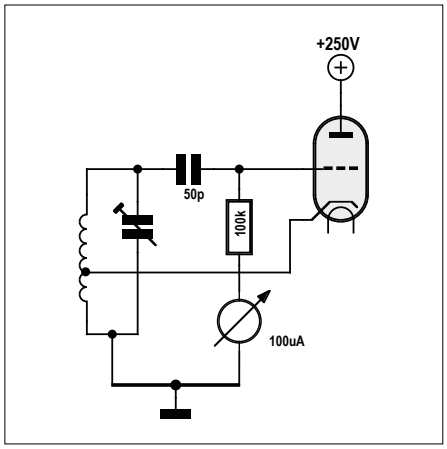


Figure 3. A grid dip meter using a triode.

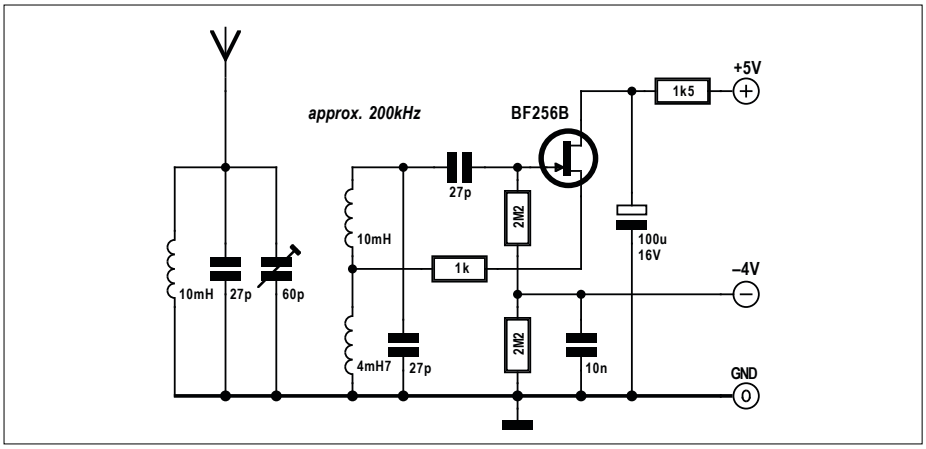


Figure 4. Modified FET dip meter.

active loudspeaker. The shield operates as a simple direct-conversion mixer without the need for any SDR software. The standalone software for the SDR shield and the extension shield allows the tuning to be adjusted easily and precisely. First it is necessary to use the buttons on the extension shield to locate the frequency of the oscillator and find the point where the beat frequency is zero. If you now move your hand close to the pitch oscillator its frequency will go down and the beat frequency will rise. And now it is just a question of a little practice, and you'll be able to play your first tune.

The connection to the antenna includes a 4.7-mH loading coil to increase its effective 'electrical length'. The result of this is to ensure that the very small extra capacitance introduced by the proximity of the hand, around one picofarad, can cause a sufficiently large change in frequency. A 1-mH inductor would require a capacitance of 143 pF for resonance at 420 kHz. With 144 pF of capacitance, the resonant frequency is 1.5 kHz lower. The load inductor considerably increases the size of this effect: you can visualize it as being equivalent to having a much longer antenna in the proximity of a much larger hand. The trick of using a loading coil was employed in the original theremin design and in variations on it ever since.

Putting it slightly more scientifically, a series resonant circuit has a low reactance near to its resonant frequency, which below this frequency corresponds to a large capacitance. The coil we have chosen here, in conjunction with an ordinary whip antenna, gives a measured resonant frequency of 600 kHz. This is not too far from the operating frequency of 420 kHz, which means that the capacitance of the hand has a noticeable effect. The result is that the sensitivity of the circuit to the capacitance of the hand is very good, with a useful working range of up to 30 cm. Because the oscillator frequency is low, stability is entirely adequate, and with a weak coupling to the SDR shield a pleasant sound is produced.

Testing the volume antenna

This simple theremin oscillator still has a shortcoming: there is no way to alter the volume of its output. The original design had two antennas rather than one, and the second antenna was operated by the player's left hand. As the hand approached the antenna the sound became quieter or even completely silent, making for a truly playable instrument.

The first idea that might come to mind is to pass one of the two oscillator outputs through a resonant circuit whose frequency is pulled by the volume antenna. Unfortunately, that will have the undesirable consequence of also pulling the pitch oscillator, resulting in an interdependency between pitch and volume.

A glance at older theremin circuits reveals an alternative approach, i.e. using a third oscillator that operates in similar fashion to a grid dip meter. A grid dip meter (**Figure 3**) is used to determine the resonant frequency of a circuit. A resonant circuit in the proximity of an oscillator coil draws energy from the oscillator circuit, reducing the amplitude of oscillation, which can be measured as a change in grid current. At the point of resonance the current 'dips', that is, falls to a very low value.

In the theremin the negative grid voltage of the oscillator is used to control the gain of a variable-mu tube in the audio signal path. This avoids potential problems of instability, as the dip frequency and the audio range lie far from the frequency of the beat frequency oscillators. In a tube-based circuit the amplitude of oscillations can be more than 10 V, and with a high value (around 1 MΩ) grid leak resistor the grid voltage might be around −10 V. If a second resonant circuit is weakly coupled in, then the grid voltage might dip to 1 V: this would correspond to maximum volume. If the resonant circuit is now detuned or damped, the negative grid voltage will rise significantly, and at −10 V the audio path will be completely blocked.

Our first attempt at a FET-based grid dip meter uses relatively large inductances (see **Figure 4**). The circuit does indeed oscillate with a large amplitude, and the output voltage is −4 V. The resonance dip effect does occur, but it is too weak: the voltage only changes by about 0.5 V. There is also a coupling effect, which also occurs with regular dip meters: the amplitude changes suddenly on one edge of the dip.

Tesla coupling

Looking at older theremin circuits, we can see that they also contain a kind of loading coil in the antenna

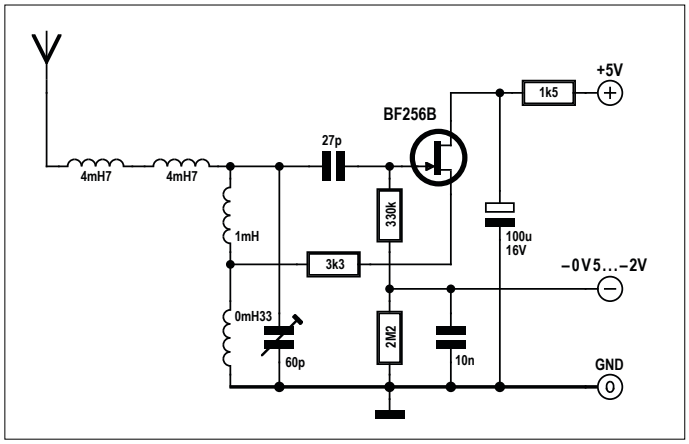


Figure 5. Oscillator with an antenna coil.

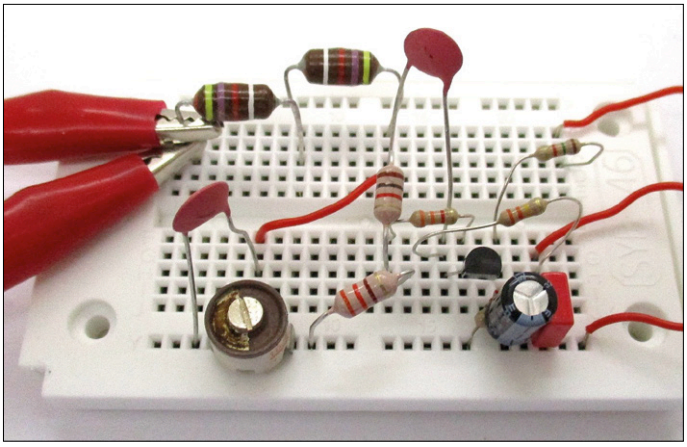


Figure 6. Prototype construction of the volume oscillator.

connection. On closer inspection this turns out to be a Tesla transformer: the coil has a self-resonance at the operating frequency and has a very large inductance to capacitance ratio. It is also strongly coupled to the oscillator, and in this case the oscillator (**Figure 5**) is tuned by the Tesla coil, whose resonant frequency will be pulled significantly even with tiny changes in capacitance.

When the circuit is in resonance the RF voltage on the antenna will be multiplied significantly. The secondary circuit draws a considerable amount of energy from the oscillator, which reduces its amplitude. The grid dip effect comes into play at voltages of up to −0.5 V, but with greater detuning or damping, for example if the antenna is touched directly, this effect disappears and the grid voltage falls by up to 2 V.

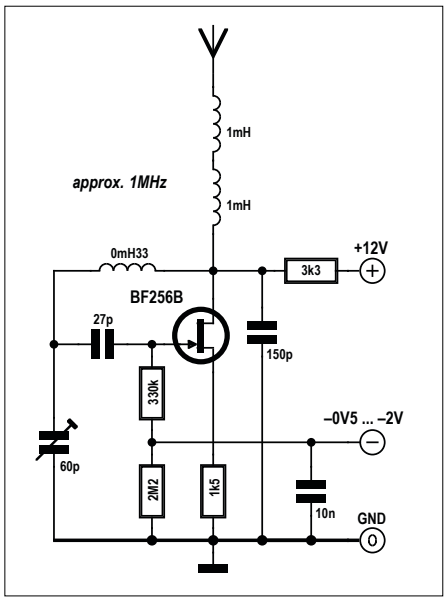


Figure 7. Alternative oscillator circuit.

The resonant frequency is easy to set, there is no longer any coupling effect and the control voltage range is large enough. For experimental purposes this oscillator can be constructed on a breadboard (**Figure 6**). A loop of cable terminated in crocodile clips can be used as an antenna.

Simplified oscillator circuit

The modified oscillator circuit shown in **Figure 7** avoids the need for a tapped inductor in the resonant circuit. Instead, the resonant circuit works using a capacitive voltage divider. By changing the ratios of the values of the individual capacitors it is possible to adjust the impedances at the gate and the drain of the FET. In the example shown the oscillator runs at around 1 MHz, which is also where the self-resonances of the two loading coils lie. As in the previous experiment the amplitude of the oscillator increases when the upper circuit is detuned or damped. The gate voltage then falls to −2 V.

The complete theremin

The complete circuit (**Figure 8**) includes the second oscillator, the mixer and a circuit for the volume antenna. The second oscillator is stabilized at 470 kHz by a ceramic resonator, and so it must be possible to set the adjustable oscillator to this frequency. A trimmer capacitor is provided for tuning.

The free-running oscillators now use 1 mH inductors. For resonance at 470 kHz a capacitance of 115 pF is needed; the capacitance of the antenna was measured at 5 pF, rising to 6 pF with a hand placed a few centimeters away. Calculation shows that this tiny change, just one picofarad, gives rise to a frequency change of 2 kHz, and only a small space near to the antenna is available to the player. The effect of the 4.7-mH loading coil is to increase significantly the change in capacitance seen at the lower end of the coil.

The dip oscillator also now uses a 1-mH inductor. The self-resonance of the loading coil for the volume antenna was found to be at 600 kHz, and so the free-running oscillator must adjusted to approximately this frequency, which requires a total capacitance of 70 pF. At 600 kHz the antenna circuit draws so much energy from the oscillator that the control voltage plunges to −2.5 V. If the antenna circuit is detuned by bringing a hand near to it the oscillator amplitude increases significantly, the control voltage reaching −3.5 V.

The 4.7 mH antenna coil requires a total capacitance of 26 pF for resonance at 600 kHz. The majority of this capacitance comes from the windings of the coil itself, only a small part being due to the antenna. Changing the capacitance by one picofarad by bringing a hand near to the antenna detunes the resonant frequency by 12 kHz, which is enough to generate the required dip.

The final element is the variable-gain amplifier. Its gain is controlled by its gate voltage: at −3.5 V the FET is fully off and there is no output signal, and with −2.5 V at the gate the circuit already has a considerable gain. In this common-gate configuration the signal is applied to the source connection of the FET and the output is coupled from the drain. The dip oscillator has to be adjusted so that it generates a suitable

range of control voltages as you move your hand near the antenna, including turning off the amplifier completely. The final stage is an audio lowpass filter to minimize the amount of residual RF at the output.

The circuit can be constructed on the copper side of an uncoated printed circuit board (**Figure 9**). This has the advantage of providing a good ground connection and decoupling the individual stages. Wherever a ground is needed, the shortest possible path can be used. This construction method, used along with small pieces of prototyping board (see **Figure 10**), can also be employed for more complex and higher-frequency RF projects.

An interesting lesson from this project is that JFET-based oscillators are very stable in frequency, and insensitive to variations in supply voltage.

In all these experiments the circuit was earthed, as it was powered from an earthed bench supply. However, when the circuit was finished and mounted in its box (**Figure 11**) and the power switched over to batteries, a problem immediately reared its head: there was a strong interaction between pitch and volume. When the instrument was earthed, the problem immediately went away. It is evident that without an earth connection the two hand capacitances are effectively in series and so will inevitably affect one another. But with the earth connection in place the instrument is very playable: you quickly get used to using your right hand to control pitch and your left hand to control volume. ⏮

(160153)

Web Link

[1] www.elektormagazine.com/160165

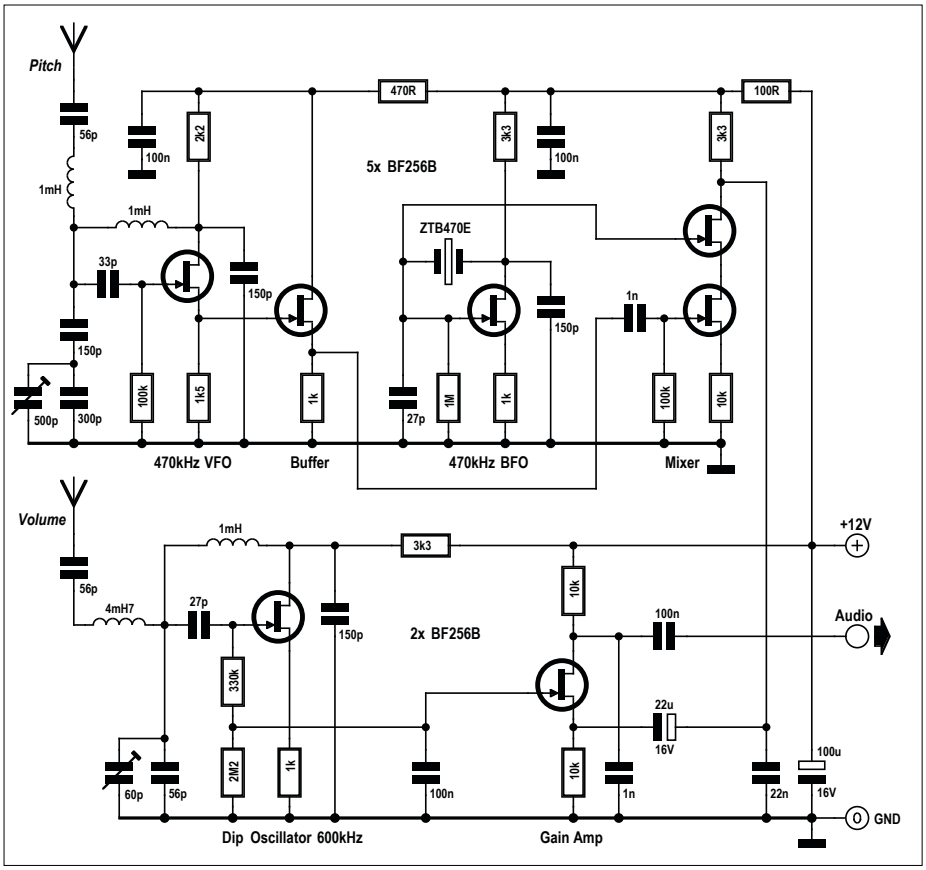


Figure 8. The complete JFET-based theremin.

The circuit can be built on an uncoated piece of printed circuit board.

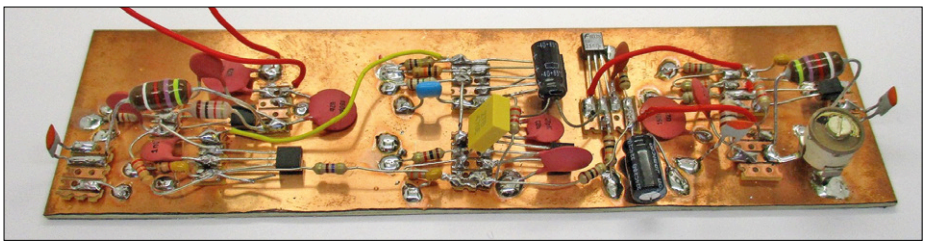


Figure 9. All the stages on one circuit board.

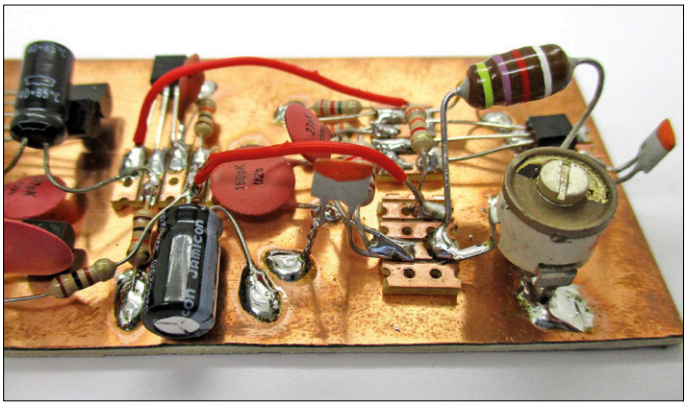


Figure 10. Close-up of the volume oscillator.

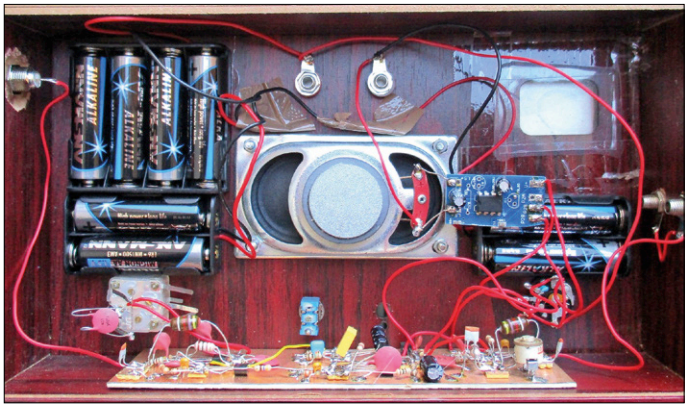
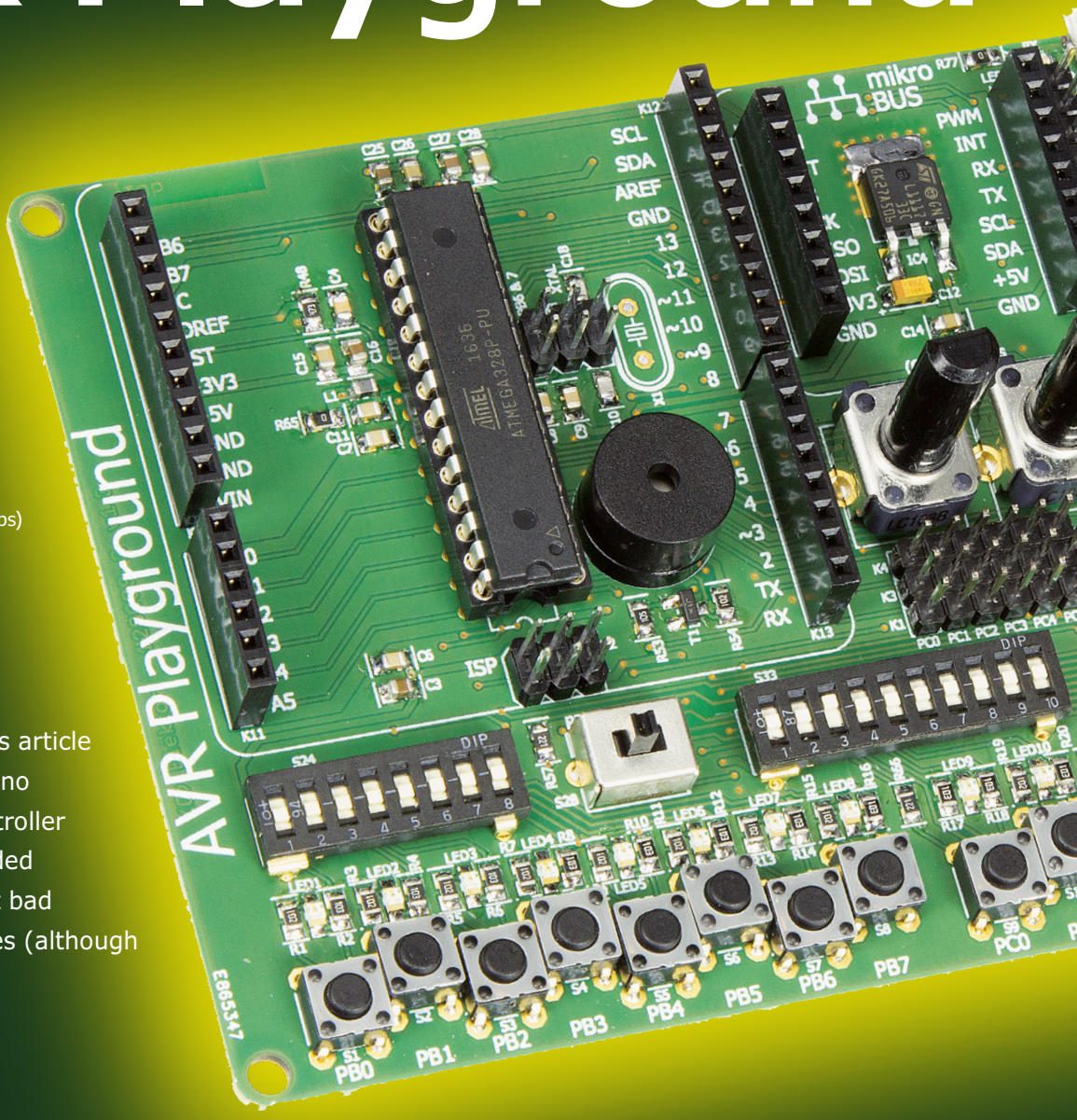


Figure 11. The instrument in its enclosure.

eLABS PROJECT

AVR Playground

Improves the way to do Arduino



By **Clemens Valens** (Elektor Labs)

The board presented in this article is a hybrid of an Arduino Uno and a traditional microcontroller development board, intended for 'doing Arduino' without bad connections and loose wires (although it doesn't disallow it).

Before the rise of Arduino, microcontroller development boards had on-board peripherals like pushbuttons and LEDs, a display, one or more potentiometers for analog signals, extension connectors, etc. and, of course, a decent power supply. The goal of these boards was to provide an easy way to start learning the microcontroller without having to solder or add other components. The AVR Playground – a playful reference to the Arduino forum known as the Arduino Playground – was designed with this in mind, and extended with things from Arduino that we have learned to appreciate.

The board can roughly be divided into four horizontal zones (**Figure 1**), with, from top to bottom, the following functions:

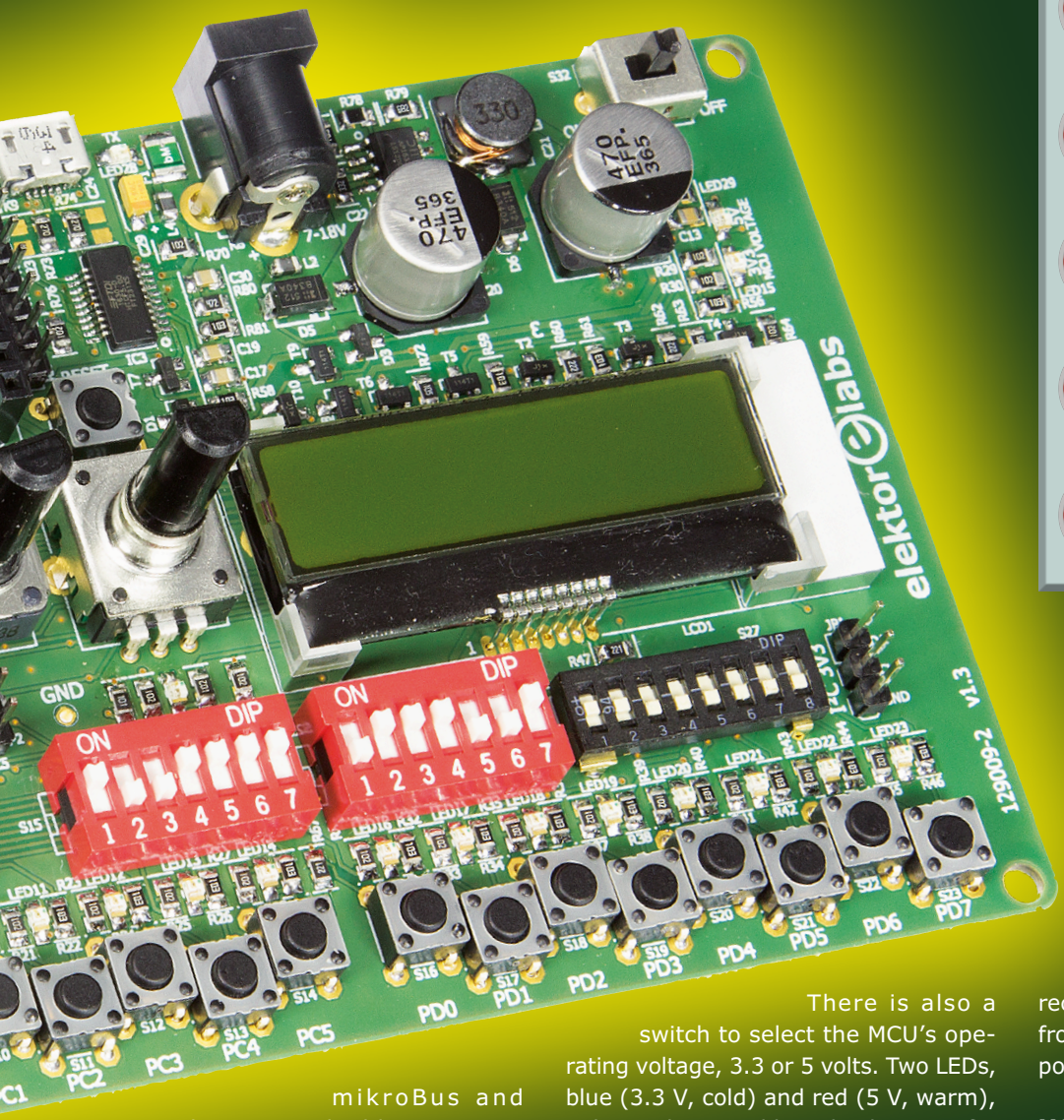
1. extension connectors, USB-to-serial converter and power supply;
2. user interface peripherals like a buzzer, rotary controls and a display;
3. configuration switches;
4. pushbuttons and LEDs.

If you want to see the schematics, the PCB or the component list of the AVR Playground, please refer to [1].

Extension connectors

The top-left corner of the board accommodates the microcontroller and Arduino Uno compatible extension connectors. This part of the board behaves exactly like an Arduino Uno.

Next to it sits a mikroBus expansion slot. This extension standard, developed by the Serbian company MikroElektronika, is gaining popularity thanks to the availability of hundreds of cheap extension boards, ranging from GPS receivers to humidity sensors and from LED arrays right up to phone modems. Together, the



mikroBus and Arduino Uno shield connectors give the AVR Playground user access to a huge library of extension boards.

PROJECT INFORMATION

Microcontrolers

Arduino | ATmega328 | Programming

entry level
➡ intermediate level
expert level

4 hours approx.

SMD Soldering

€75 / $80 / £65 approx.

Pushbuttons, LEDs & DIP switches

The bottom part of the board, zone 4, is occupied by pushbuttons and LEDs connected to every microcontroller port that can be used. The pushbuttons allow applying a logical level to a port, while the LEDs provide visual feedback. The DIP configuration switches in zone 3 of the board determine:

- if a GPIO port is pulled up or down or not at all;
- if pressing a pushbutton provides a logic low or high;
- if the LEDs are connected or not and if user interface peripherals from zone 2 are connected or not.

There is also a switch to select the MCU's operating voltage, 3.3 or 5 volts. Two LEDs, blue (3.3 V, cold) and red (5 V, warm), indicate the actual board voltage. Refer to **Tables 1 to 5** for details.

Power supply

The power supply was designed with quality and robustness in mind. Protected against short circuits and high temperatures, it is able to deliver 5 V at 1 A without complaining. There is one condition, though: you must power it from an external power source like a DC adapter (wall wart) capable of providing about 7 V minimum (and, of course, enough current). The USB port may also be used as a power source but this will limit the available current to 500 mA in order to protect your computer. The 5 V drives a beefy 3.3-V low-dropout voltage regulator so that in 3.3-V mode too, enough power is available for your experiments. An on/off switch can cut the power to the rest of the board, enabling safe hardware reconfiguration without disconnecting it from the computer and losing the serial port connection.

USB-to-serial converter

The USB-to-serial converter not only acts as a 5-V power source for the board, it is also the programming interface for Arduino sketches and, of course, a USB-compatible serial port for user applications. It can be disconnected entirely from the microcontroller, to free up GPIO pins for instance or to use it on other ports.

Human interface devices

In zone 2, i.e. the middle of the board, we find typical user interface devices like a buzzer, two analog controls in the shape of potentiometers, a rotary encoder and an alphanumerical display. Together with the pushbuttons and LEDs they allow the creation of human-friendly applications without soldering. With an Arduino Uno, a few prototyping boards and a spaghetti of connecting wires it is possible to achieve the same goals, but

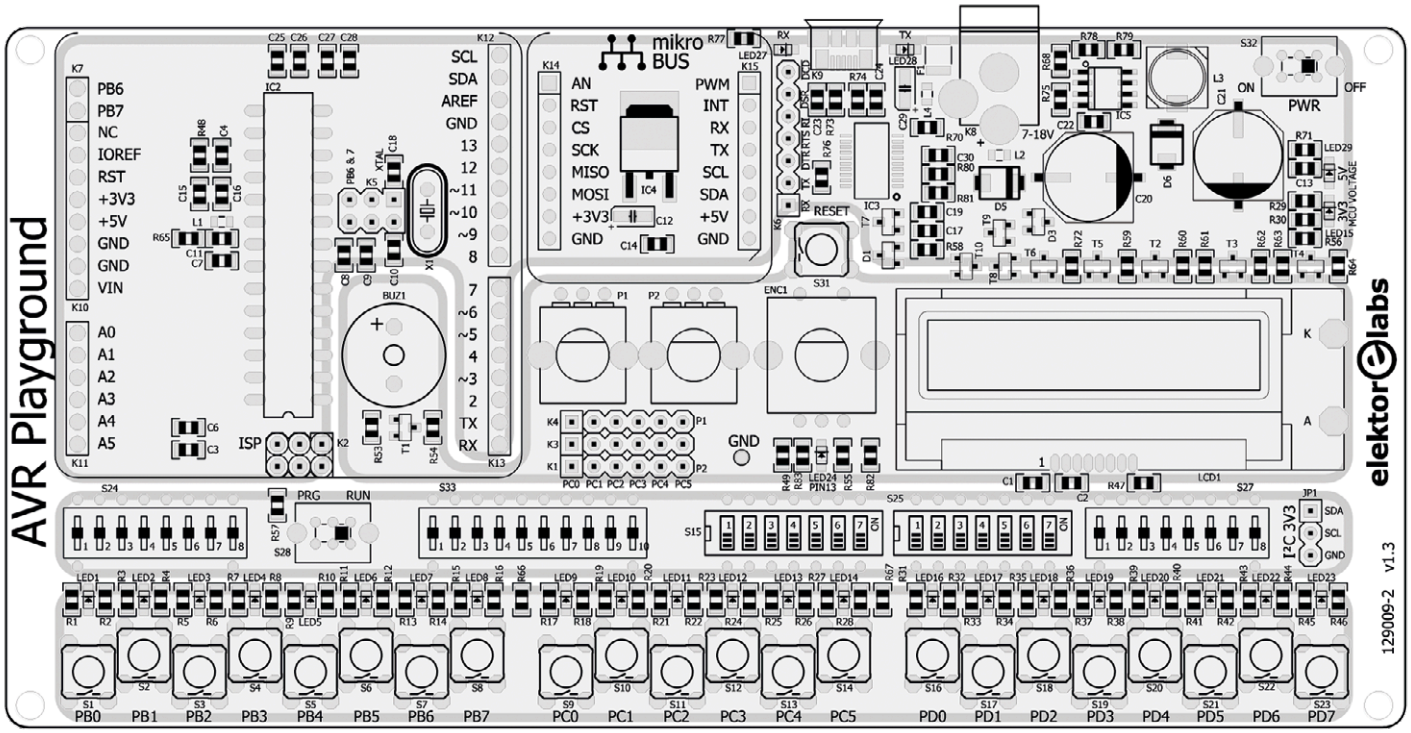


Figure 1. The AVR Playground can be divided into four functional zones.

things are much more comfortable and reliable when using a tool like the AVR Playground.

LCD

The display, a small LC one with backlight, has an I²C interface instead of the frequently seen 4-bit semi-parallel or 8-bit parallel interface. The evident advantage is that it only requires two wires for connecting it to the MCU. The inconvenience is the need for a special driver, but we took due care of that in the Boards Package (see below).

Table 1. The functions of DIP switch S15.

S15	Function	Off	On
1	Buzzer	Disconnected	Connected to PB1
2	LEDs Port B	Disconnected	Connected to GND
3	LEDs Port C	Disconnected	Connected to GND
4	LEDs Port D	Disconnected	Connected to GND
5	USB-to-serial RXD	Disconnected	Connected to PD1
6	USB-to-serial TXD	Disconnected	Connected to PD0
7	USB-to-serial DTR	Disconnected	Connected to Reset

Table 2. The functions of 7-way DIP switch S25.

S25	Function	Off	On
1	MCU voltage	5 V	3.3 V
2	'Arduino LED'	Disconnected	Connected to PB5
3	LCD SDA	Disconnected	Connected to PC4
4	LCD SCL	Disconnected	Connected to PC5
5	Not used		
6	LCD Backlight	On (if S25-7 in On position)	Connected to PD7
7	LCD Backlight	Off	Controllable

Table 3. The functions of DIP switch S24.

S24	Port	Down	Middle	Up
1	PB0	Pulled down	Not pulled	Pulled up
2	PB1	Pulled down	Not pulled	Pulled up
3	PB2	Pulled down	Not pulled	Pulled up
4	PB3	Pulled down	Not pulled	Pulled up
5	PB4	Pulled down	Not pulled	Pulled up
6	PB5	Pulled down	Not pulled	Pulled up
7	PB6	Pulled down	Not pulled	Pulled up
8	PB7	Pulled down	Not pulled	Pulled up

Rotary controls

Two potentiometers provide analog signals to the microcontroller. By positioning a jumper, they can be connected to any of the six analog inputs of the MCU without the risk of both being connected to the same input at the same time. The rotary encoder is in reality the same thing as (up to) three pushbuttons hence it's effectively connected in parallel to the pushbuttons on PD3, PD4 and PD5.

Arduino LED

The LED connected to PB5 (Pin 13) on the Arduino Uno is available on the AVR Playground too; it is located below the rotary encoder. This LED is used in many sketches, reason why it is present on this board.

Clock frequency & reset issues

The preferred clock oscillator for the AVR Playground is the MCU's internal 8-MHz RC oscillator. This ensures that the MCU will always work within its specifications, no matter if the MCU voltage is 5 V or

3.3 V. Because the quartz crystal is disconnected by default, ports PB6 and PB7 are available for user applications. If a crystal is required, it can be soldered on the board and connected to the MCU by moving two jumpers.
In normal operation port PC6 functions as the reset input of the MCU. It is possible to disconnect it from its reset function by programing the MCU's RSTDSBL fuse. However, that's not recommended as it will disable MCU programming over the serial port **and** through the ISP connector. The only way to reprogram the MCU is to remove it from the board and land it in a so-called parallel programmer. Because disabling the reset input is incompatible with the objectives of the AVR Playground, PC6 is not treated like the other port pins and there is no LED connected to it (there is, however, a pushbutton for it: Reset).

Table 4. The functions of DIP switch S33.

S33	Port	Down	Middle	Up
1	Port B pushbutton level	Low	Disconnected	High
2	Port C pushbutton level	Low	Disconnected	High
3	Port D pushbutton level	Low	Disconnected	High
4	PC0	Pulled down	Not pulled	Pulled up
5	PC1	Pulled down	Not pulled	Pulled up
6	PC2	Pulled down	Not pulled	Pulled up
7	PC3	Pulled down	Not pulled	Pulled up
8	PC4	Pulled down	Not pulled	Pulled up
9	PC5	Pulled down	Not pulled	Pulled up
10	Not used			

Table 5. The functions of DIP switch S27.

S27	Port	Down	Middle	Up
1	PD0	Pulled down	Not pulled	Pulled up
2	PD1	Pulled down	Not pulled	Pulled up
3	PD2	Pulled down	Not pulled	Pulled up
4	PD3	Pulled down	Not pulled	Pulled up
5	PD4	Pulled down	Not pulled	Pulled up
6	PD5	Pulled down	Not pulled	Pulled up
7	PD6	Pulled down	Not pulled	Pulled up
8	PD7	Pulled down	Not pulled	Pulled up

Installing the AVR Playground

Although it is possible to use the AVR Playground without installing any software, we would not recommend it. Because the board sports features not supported by the standard Arduino IDE, we prepared some libraries that make using the on-board peripherals easier. Having that said, if, for some reason, adding a board to the Arduino IDE is unwanted, know that the standard board 'Arduino Pro or Pro Mini' with as processor (from the 'Tools → Processor' menu) the 'ATmega328 (3.3 V, 8 MHz)' can be used instead. The voltage is not important, what counts is the frequency having to match that of the MCU's clock oscillator.

Adding a new board to the IDE is not difficult and there is even a special tool for this: the Boards Manager, accessible at the top of the 'Tools → Boards' menu. The Boards Manager allows installing, updating and removing third-party boards. For this to work, the manufacturer of such a board must provide a Boards Package telling the IDE what to download and use for the board.

The procedure to install the AVR Playground's Boards Package is quite simple but requires an Internet connection. It starts from the 'File' menu by opening the 'Preferences' dialog of the Arduino IDE (see **Figure 2**, Arduino version 1.6.13 or newer from arduino.cc; do not use 1.7.x from arduino.org). Copy the URL below or, even better, read the QR code from **Figure 3**:

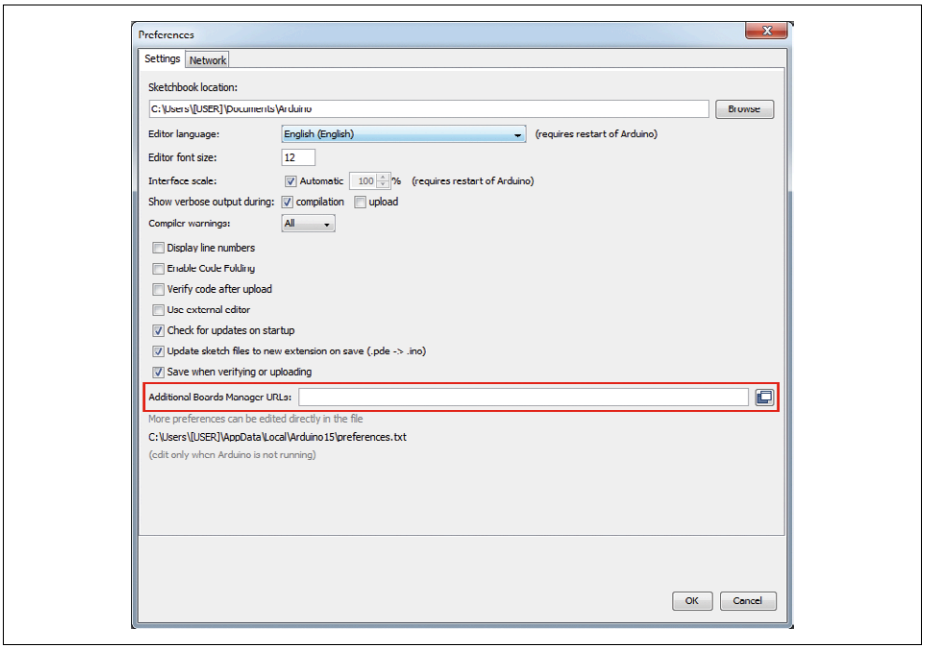


Figure 2. This is where you enter the url to access the AVR Playground Boards Package.

```
https://raw.githubusercontent.
    com/ElektorLabs/arduino/
    master/package_elektor_
    boards_index.json
```

Figure 3. Avoid typing mistakes by reading this QR code with your webcam, then copy-paste the URL into the 'Additional Boards Manager URLs' box of the 'Preferences' dialog.

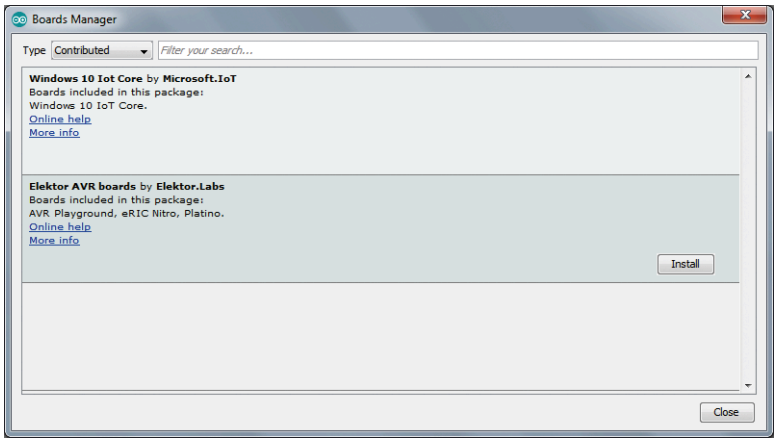


Figure 4. Once the IDE has found the AVR Playground Boards Package it will allow you to install it.

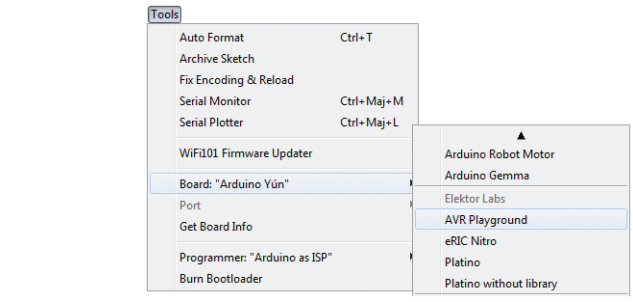


Figure 5. After installing the Boards Package, scroll through the Boards list and select the AVR Playground.

(one line, no spaces, beware of typing errors) into the 'Additional Boards Manager URLs' box of the 'Preferences' dialogue. Close it when done.

Open the Boards Manager ('Tools → Boards'). In the upper left corner of the window that opens select 'Contributed', look for the AVR Playground in the list that appears, click on it and then click the 'Install' button (**Figure 4**). The IDE will download the required files and copy them to right location. When done, close the window. Now the AVR Playground will be listed somewhere in the 'Boards' menu, under the header 'Elektor Labs' (**Figure 5**). Of course you should connect the AVR Playground to your computer before you can select its 'Port'.

Enter game programming

We developed a fun application using many of the board's options, it's a simple game called 'Simon Says' inspired by a famous game from the early years of the microcontroller. The original game came as a round black plastic box with four large, backlit, colored buttons: red, blue, green and yellow. The computer plays a random luminous sequence where every light is accompanied by a musical note. When done the player is invited to play the same sequence. If the player fails, the game restarts. If the player succeeds the sequence is extended by one randomly chosen color/note. The game is still rather popular among programmers and you can easily find a version of the game for your smartphone or to play online.

The AVR Playground has everything needed to create and play the game: LEDs, pushbuttons, a buzzer and, of course, a microcontroller. What's more, the game can be extended by adding the LCD to show things like instructions, the high score, and some other statistics. There are no colored lights but bits of colored plastic film or paper can help here.

In what follows, fragments of the demo program are highlighted, the complete source code can be downloaded freely from [1].

Table 6. The ports, their pins and their functions in the game. Ten GPIO pins remain unused, showing that a microcontroller with fewer pins might suffice, making the final design cheaper.

Port B	Function	Port C	Function	Port D	Function
PB0	Not used	PC0	Not used	PD0	RXD
PB1	Buzzer	PC1	Not used	PD1	TXD
PB2	Not used	PC2	Not used	PD2	LED0
PB3	Not used	PC3	Not used	PD3	LED1
PB4	Not used	PC4	LCD SDA	PD4	LED2
PB5	Not used	PC5	LCD SCL	PD5	LED3
PB6	Not used			PD6	LED4
PB7	Not used			PD7	LED5

Port mapping

The first step is to decide which ports will be used for what function (**Table 6**). This also greatly determines the setting of the configuration DIP switches (see **Tables 7 to 11**). The settings chosen will switch the LCD's backlight on, and, because the level of the pushbuttons on port D is set to logic high, an LED will automatically light when the corresponding button is pressed. To make this free visual feedback work properly, the pull-down resistors on these pushbuttons need to be activated.

Controlling the LEDs...

... is slightly more complicated than usual because the pushbuttons are connected to the same pins. Also, to improve flexibility, a lookup table is added allowing the use of other LEDs simply by modifying the table. **(Listing 1)**

Reading the pushbuttons...

... is based on the same technique as lighting the LEDs, except for an input now being read instead of an output being driven. Due to the way the DIP switches are set, a pressed button will produce a logic high.

Table 7. Settings of S15 according to our plans.

S15	Function	Position
1	Buzzer	On
2	LEDs Port B	Off
3	LEDs Port C	Off
4	LEDs Port D	On
5	USB-to-serial RXD	On
6	USB-to-serial TXD	On
7	USB-to-serial DTR	On

Table 8. Settings of S25 for our game.

S25	Function	Position
1	MCU voltage	Off
2	'Arduino LED'	On
3	LCD I²C SDA	On
4	LCD I²C SCL	On
5	Not used	
6	LCD Backlight	Off
7	LCD Backlight	On

Table 9. S24 controls Port B, it's not used in our game.

S24	Port	Position
1	PB0	Middle
2	PB1	Middle
3	PB2	Middle
4	PB3	Middle
5	PB4	Middle
6	PB5	Middle
7	PB6	Middle
8	PB7	Middle

Table 10. S33 controls Port C and the pushbutton's active levels.

S33	Function	Position
1	Port B pushbutton level	Middle
2	Port C pushbutton level	Middle
3	Port D pushbutton level	High
4	PC0	Middle
5	PC1	Middle
6	PC2	Middle
7	PC3	Middle
8	PC4	Middle
9	PC5	Middle
10	Not used	

Table 11. S27 controls Port D.

S27	Port	Position
1	PD0	Middle
2	PD1	Middle
3	PD2	Down
4	PD3	Down
5	PD4	Down
6	PD5	Down
7	PD6	Down
8	PD7	Down

A problem with pushbuttons is that they are mechanical devices in essence, exhibiting milliseconds or more of time lag for the contact to settle, hence it is necessary to 'debounce' the buttons. An easy way to do this when there are no particular (timing) constraints as in our simple game, is to scan the buttons a second time after having waited a short while. Only if a button is read twice as being pressed, the program decides it's a valid button press. By scanning periodically at a high enough rate (10 Hz or so) the chances of missing a key press are minimal. A logic AND ('&') of two scan results will remove inconsistent reads. **(Listing 2)**

Random numbers

The game needs a sequence of random numbers (corresponding to the LED numbers) that's long enough to make most players fail before reaching the end. The sequence can be stored in a table. The size of the table is important. If it is too small, it is too easy for the player to outplay the computer; if it is too large, the player gets discouraged and will give up. The table can be filled at the start of a game.

The function `rand` is available for producing random values in the range 0 to `RAND_MAX` (which corresponds to 32,767 in Arduino). A quick and dirty way to bring the output in the wanted range is by doing a modulo division ('%').

Listing 1.

```
#define LED_BUTTON_TABLE_SIZE  (4)
uint8_t led_button_table[LED_BUTTON_TABLE_SIZE] = { 4, 5, 6, 7 };

void led_set(uint8_t nr, uint8_t value)
{
  pinMode(led_button_table[nr],OUTPUT);
  digitalWrite(led_button_table[nr],value);
}
```

Listing 2.

```
uint8_t button_read_all_debounced(void)
{
  uint8_t buttons = button_read_all();
  delay(10); // Wait for any bouncing to stop.
  buttons &= button_read_all(); // AND the second scan.
  return buttons;
}
```

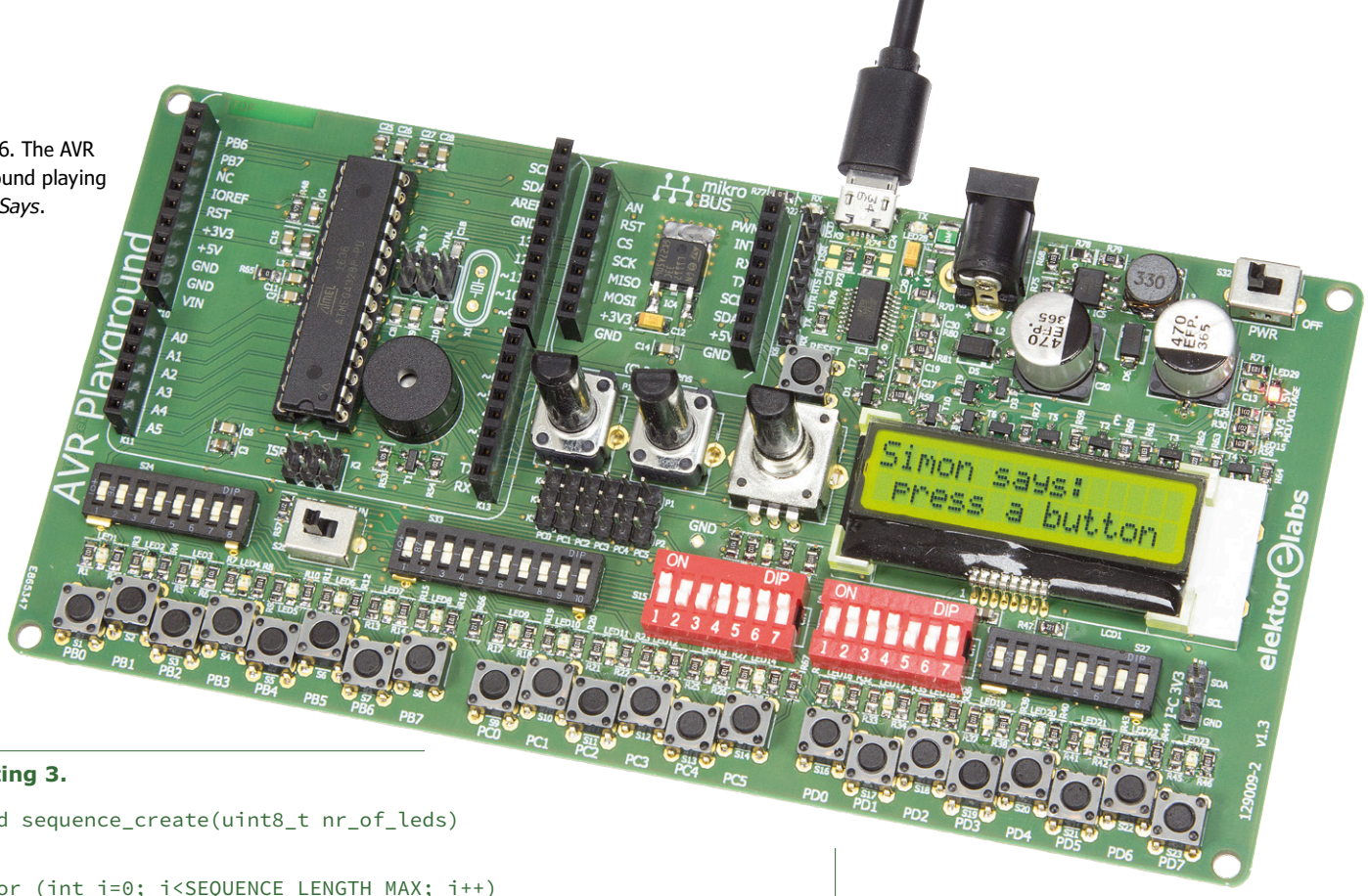


Figure 6. The AVR Playground playing *Simon Says*.

Listing 3.

```
void sequence_create(uint8_t nr_of_leds)
{
  for (int i=0; i<SEQUENCE_LENGTH_MAX; i++)
  {
    sequence[i] = rand()/(RAND_MAX/nr_of_leds + 1);
  }
}
```

Listing 4.

```
void setup(void)
{
  while (button_pressed()==false);
  srand(millis());
  sequence_create(LED_BUTTON_TABLE_SIZE);
  game_round = 1;
}
```

Listing 5.

```
const uint8_t buzzer = 9;
const uint16_t button_sound[LED_BUTTON_TABLE_SIZE] =
{ 554, 659, 880, 1319 }; // C#5, E5, A5, E6 in Hz

#define SEQUENCE_SPEED  (400) /* ms */

void sequence_play(uint8_t len)
{
  for (int i=0; i<len; i++)
  {
    tone(buzzer,button_sound[sequence[i]],SEQUENCE_SPEED/2);
    led_set(sequence[i],true);
    delay(SEQUENCE_SPEED);
    led_set(sequence[i],false);
    delay(SEQUENCE_SPEED/4);
  }
}
```

However, from a mathematical point of view this approach is questionable because the distribution of the output may no longer be uniform and independent due to the random number generator's implementation. **Listing 3** shows a better technique to produce random numbers in a small range.

Random numbers in programming are problematic because programming is deterministic by definition. For the function `rand` to work properly it must be initialized with a random number (the seed), and a chicken-and-egg problem is the result. A trick often encountered is to use the time as the seed, because time, as you know, never stops and thus makes an excellent seed… except in microcontroller systems where time is reset to zero at every (re)start.

For our game time can be used to seed the random number generator if we measure the time it takes for the player to press the first button. Measured in milliseconds the result will hardly ever be the same. **(Listing 4)**

Play a sequence, add sound

Every game round, the sequence is a bit longer and must be played to the user starting from the beginning. If the sequence is played too fast, the player may not be able to memorize it; if it is too slow, he or she may get bored.

To enhance Player Experience we add a musical note to every LED, made audible by the buzzer (connected to pin 9, PB1). **(Listing 5)**

The table `button_sound` holds the frequencies in hertz of the notes to play. The constant `SEQUENCE_SPEED` (in milliseconds) determines the replay speed of the sequence. The function `sequence_play` takes as its argument the length of the sequence to play.

Processing user input

Once the sequence has been played, it is time for the player to repeat it. As soon as the player makes a mistake the program can stop reading pushbuttons. There is one little complication to take care of: avoiding that a very long key press gets interpreted as two or more presses. **(Listing 6)**

Play the game

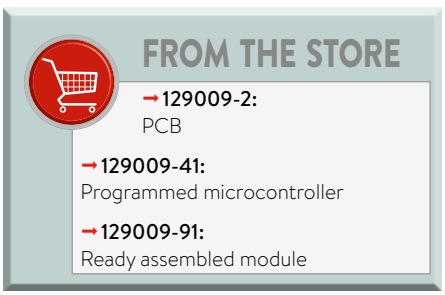
All that remains to do is to add the function `loop` to glue together the functions `sequence_play` and `sequence_read_buttons` and to keep track of the sequence length and the state of the game. **(Listing 7)**

The bodies of the `if-else` statements have been left largely empty here because it is up to you what to put inside. Consider playing a tune depending on the state of the game, a 'Well Done' tune, a 'You Win' tune and a 'Game Over' tune, most probably accompanied by flashing LEDs. Let your creativity run wild and have fun designing special effects.

(160316)

Web Link

[1] www.elektormagazine.com/160316



Listing 6.

```
bool sequence_read_buttons(uint8_t len)
{
  for (int i=0; i<len; i++)
  {
    uint8_t nr, buttons;

    // Wait for a button press.
    do
    {
      buttons = button_read_all_debounced();
    }
    while (buttons==0);

    // Check that we have a valid button press.
    nr = button_as_number(buttons);
    if (nr!=BUTTON_PRESS_INVALID && nr<LED_BUTTON_TABLE_SIZE)
    {
      // Play sound & wait until it finishes.
      tone(buzzer,button_sound[nr], SEQUENCE_SPEED/2);
      delay(SEQUENCE_SPEED/2);
      // Wait until the player releases the button.
      while (button_pressed()==true);
      // Fail on wrong button.
      if (sequence[i]!=nr) return false;
    }
    else return false;
  }
  return true;
}
```

Listing 7.

```
void loop(void)
{
  sequence_play(game_round);
  if (sequence_read_buttons(game_round)==true)
  {
    if (game_round<SEQUENCE_LENGTH_MAX)
    {
      // Next round, play 'well done' tune?
      game_round++;
    }
    else
    {
      // Player wins, play 'you win' tune?
      game_start();
    }
  }
  else
  {
    // Game over, play 'game over' tune?
    game_start();
  }
}
```



Debugging and Decoding Digital Communication

using the SmartScope

By **Riemer Grootjans** (LabNation)

Even though the LabNation SmartScope is mainly known as a full-featured oscilloscope, its rear port allows you to hook up eight probes to measure digital signals. This article explains how to use the SmartScope as a logic analyzer to validate your digital circuits, and to automate the decoding of digital protocols for you.

Most of today's circuits have a digital section; it's either a purely digital circuit, or the digital core is used to control an analog circuit or capture its outputs. When a digital circuit isn't performing as it should, you want to see at which moments your digital lines are High and when they're Low; you want to see at which moments exactly they change state.

The most basic usage of a logic analyzer is to study an event and its effect: attach one wire to the signal you want to monitor and a second wire to a trigger. Each time the trigger occurs, the logic analyzer captures the behavior of both signals over a certain timespan, allowing you to analyze this at your own pace. For most cases, digital communication is all about multiple lines which need to respect a certain mutual timing. The SmartScope offers eight digital inputs and includes quite some functionality to measure the exact time between various pulse transitions in your communication streams.
And finally, logic analyzers are also used to monitor serial communication for passing messages between chips. Decoders help you to convert those streams of transitions into meaningful commands and values.

Key specifications

When looking at the SmartScope acting as a logic analyzer, there are quite a few specifications to be aware of.

- **Number of inputs:** even though two inputs would be the bare minimum (to wait for a trigger and watch its effect on a signal), more is better in this case, as it allows you to combine the bits of multiple inputs into numbers. Sometimes you'll want to trigger on a number instead of a single signal. The SmartScope has eight digital inputs

on its rear port, on which you can either connect the 0.1" Dupont wires which ship with each SmartScope, or LabNation's dedicated Logic Analyzer cable.

- **Input voltage range and protection:** the logic analyzer stage of the SmartScope can capture digital High levels between 1.8 V and 5.0 V. Newer chips typically use lower voltages, to reduce the power consumption and hence the EM radiation caused from charging and discharging the tracks and pins.
- Each input of the SmartScope contains protection diodes in both directions, making sure the circuitry will survive in case of unexpected behavior or voltage/ESD spikes. Just make sure you do not apply a continuous voltage level above 5.5 V.
- **Sampling rate:** the SmartScope samples your signals at 100 MS/s. This allows you to visualize pulses as short as 10 ns. Translated to duty cycle, this allows you to measure the duty cycle of a 500-kHz wave with a resolution of 1%.
- **Sample memory/RAM:** small sample memories (<10 K) are sufficient if you want to visualize simple cause-and-effect situations. However, when you want to capture the communication between digital chips, you need much more sample memory. For this reason, the SmartScope is equipped with on-board RAM which can capture 4 M samples for each channel. This allows you to capture a complete transmission and then take your time to zoom in on any part of the transmission, decoding and verifying every edge of it. This brings us to the next point…
- **Protocol decoders:** checking each and every edge of a digital communication can be very cumbersome and frustrating. While it is possible to manually translate short sequences, the counting of edges is bound to cause headaches soon.
  To make this easier, the SmartScope software comes with built-in decoders for the most common protocols, including 1wire, UART/RS232, I²C, I²S and SPI. The decoder system of the SmartScope is very flexible, so decoders can also be built to display additional information such as the total number of edges, or the time between edges. You can even create your own decoder [1].
  For the sake of completion, protocol decoding is handled in software on almost all USB-based logic analyzers. Consequently, data is captured over a defined timespan, transferred to the PC, and this section gets decoded in software. Even though you can search through decoded data, there is no robust way to trigger on it. This would require the decoded data to be available in hardware, and the logic analyzers that do support this typically cost a multiple of a full SmartScope.

Getting started with the Logic Analyzer

This section explains how to get started using the Logic Analyzer of the SmartScope, and how to achieve the best results with it.

Connect properly to your hardware: just as with an oscilloscope, a logic analyzer needs to have a reference of the 0 V level on the device to measure. Therefore, you will need to make at least one connection between Ground of the SmartScope and Ground of the device you want to test. After that, connect the inputs of the SmartScope to the signals in your circuit of which you want to capture the data.

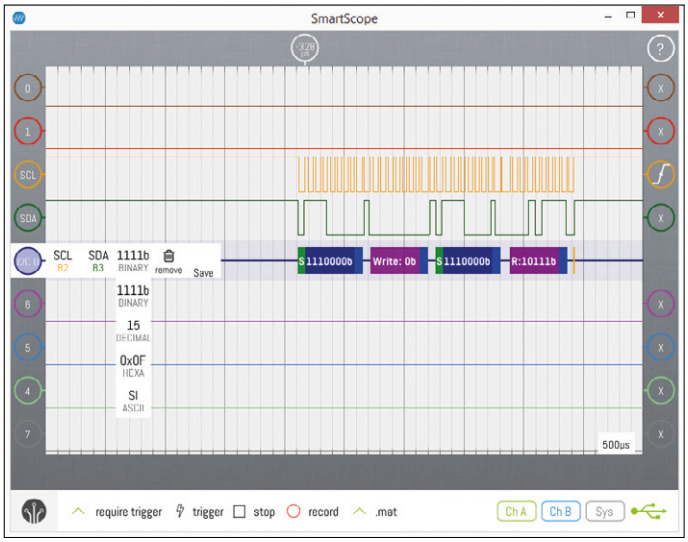


Figure 1. A protocol decoder at work.

Set up the software: after connecting the SmartScope to your pc/tablet/phone, start up the SmartScope app and follow *Menu → Digital mode*. By default, the eight digital channels will pop up. In case you didn't connect all channels and you want to free up some space, tap on the indicator of an unused channel and select 'Hide'. The channel will be removed and its indicator is parked at the bottom-right side of the screen (**Figure 5**). Simply tap it to restore that channel.

Configure the triggering: when you have connected the wires to a running circuit, you are likely to see lots of signals coming through in a random fashion. By setting up a trigger condition, you can make the SmartScope wait for a certain combination of the digital signals to occur, before it is captured and displayed. This is achieved by changing the symbol inside the trigger indicator on the right of each signal. In **Figure 1** for example, the SmartScope will wait for a rising edge to occur on D2.

The trigger position can also be adjusted to left or right, allowing you to focus on the part of interest. Note that the SmartScope has a very wide range for this trigger position, meaning you are at liberty to set the trigger position outside the boundary of your screen, or even outside the boundary of your RAM. Notice that the trigger location inside the RAM is indicated by the red vertical line in the panorama at the top of the screen. Finally, there's the triggering mode. In 'Require trigger' mode, the SmartScope will wait for a trigger condition to happen and only then display the captured sequence on the screen. 'Normal trigger' mode behaves identically, but if no trigger condition was detected for one second, the SmartScope will simply capture sequences at will, and display them on the screen. 'Single trigger' mode waits for a trigger condition to occur, and subsequently sends that acquisition to the screen in its finest detail (i.e. the entire RAM content is transferred at that time).

Accessing basic information

With an acquisition frozen on screen, there is a lot of information you can access immediately, as illustrated in **Figure 2**.

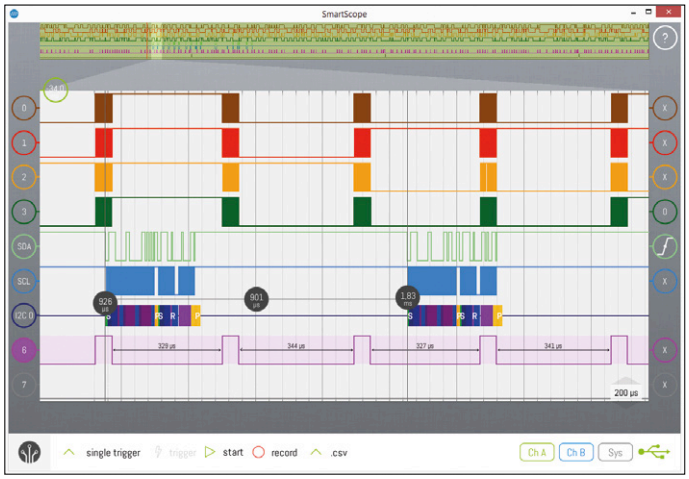


Figure 2. Interface to a 5-MP RGB image sensor. The bottom lines are the HSync and VSync, the top four lines contain actual pixel data. D4 and D5 are the Data and Clock lines of the I²C channel, with a decoder added to decode their data.



Figure 3. Decoded I²C message.

- **Interval timing:** for the currently selected wave, the time interval between all edges is displayed. Hover your mouse over any wave to display its interval information as well.
- **Time cursors:** drag in a time cursor from the side of the graph onto the graph. By adding a second cursor, a delta-cursor will appear which displays the timespan between both cursors. When you tap that delta-cursor, the timespan will be replaced by the corresponding frequency.
- **Edge decoders:** In case you need to know the exact number of edges displayed between two locations, follow *Menu → Add decoder → Edge counter*. Select the channel you're interested in and the decoder will count the edges for you. Adding the 'Edge intervals' decoder instead will show you the time between each interval, down to 10 ns accuracy.

Using the digital decoders

In a lot of systems, digital chips communicate with each other over some low-speed channel to pass on configuration data and/or low-speed measurement data.

Such a channel is also present in Figure 2, where the controller is passing a message towards the RGB imager using the I²C protocol. Each protocol is well defined; in case of I²C the message starts with a byte containing the address of the target chip and whether the master wants to read or write to it. Each bit of the SDA line should be interpreted at the moment there's a rising edge on the SCL line.

Based on this information, you can start off decoding the data you see on the screen, noting down the value of the SDA line at each rising edge of the SCL line. Or you can ask the SmartScope app to do this job for you: simply follow to *Menu → Add decoder → I2C Decoder*. The decoder will be added to the screen, and the software tries to detect which wire is SDA and which is SCL. The image of **Figure 3** shows a more complex situation with multiple lines connected at low and high frequencies, so we need to give the software a hand. With the decoder menu opened by default, hit the SCL button and select D4. Next, hit the SDA button and select D5. Finally hit the indicator itself to remove the menu, and you will see that the decoder managed to decode the entire communication for you.

Changing the type of output of the decoder

Sometimes you wish to know the decoded values as simple decimal numbers, while at other times you need them in hexadecimal notation. You can specify this by tapping on the indicator of the decoder, tap on the radix button and then select the type of output you want (Figure 1). In this way you can translate, say, UART communication straight into ASCII.

Creating your own decoder

The decoder subsystem of the SmartScope app is very flexible. You can even code your own decoder in C#, compile it and place it in the same folder as the app [1]. This allows you to create a decoder for your own protocol, or you can even stack decoders. By stacking them, you can for example create a decoder which takes the output of the I²C decoder and translates the register addresses to register names.

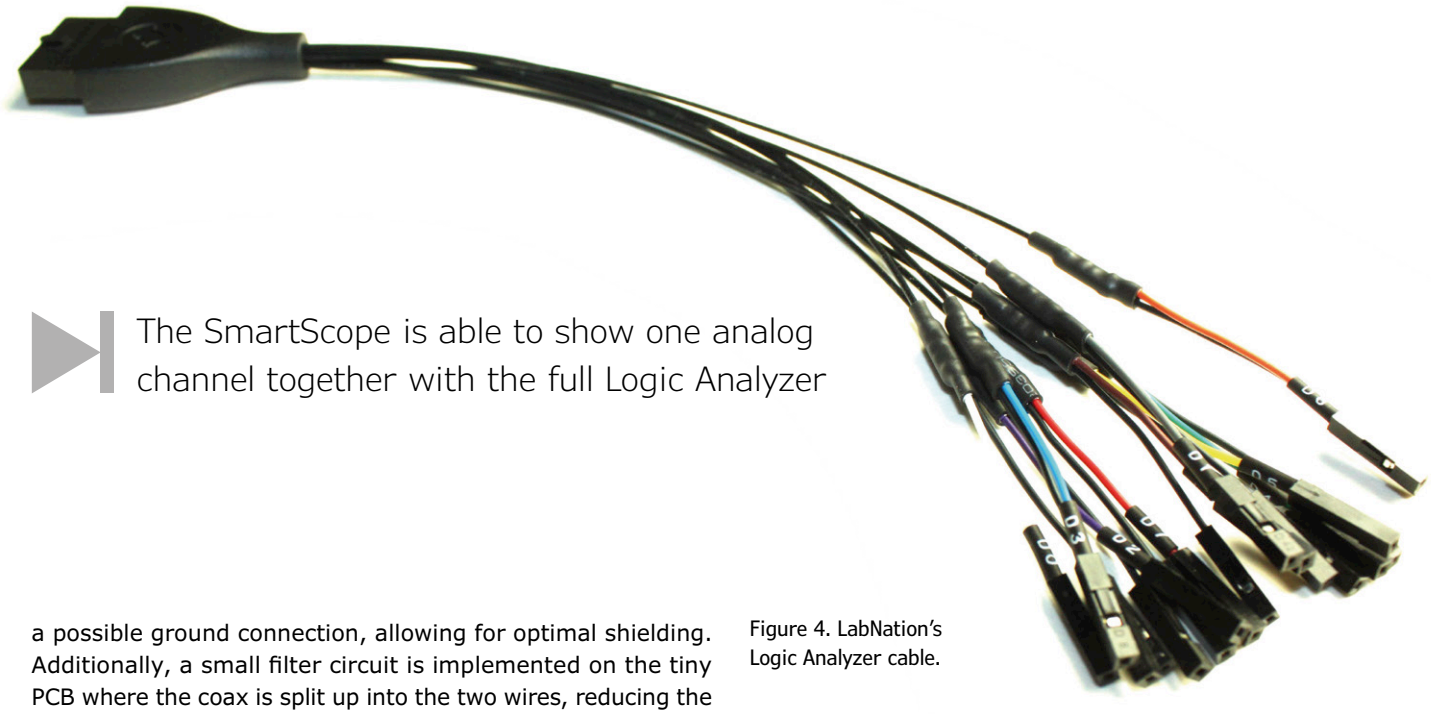
Achieving optimal results

Some fiddling with the wires may be needed to get correct measurements, especially on signals with high-frequency signals which are actively driven Low and High. Because the wires between the SmartScope and the device you want to measure are unshielded, crosstalk between the wires can occur. As such, a rising edge on one wire may cause a very short pulse on a neighboring wire. Since both electrical and magnetic influences exist, when using unshielded wires the best approach is to use short, well-spaced wires that are kept close to a ground wire though.

Additionally, the included wires act as a small capacitance which you add to the signal you're monitoring and this can influence the signal itself up to a point that communication starts failing, just because you've hooked the wire to the signal.

LabNation Logic Analyzer cable assembly

In view of the above issues, LabNation has developed a dedicated Logic Analyzer cable (**Figure 4**) to connect to the AUX port on the rear side of the SmartScope. For each digital input, the cable provides shielding by using a grounded coaxial wire. At the side of the signal to measure, each coax line is split up into a short wire for the signal itself and another wire for



> The SmartScope is able to show one analog channel together with the full Logic Analyzer

Figure 4. LabNation's Logic Analyzer cable.

a possible ground connection, allowing for optimal shielding. Additionally, a small filter circuit is implemented on the tiny PCB where the coax is split up into the two wires, reducing the influence of the cable on the signal itself to a bare minimum.

Decoding strategy

Among the practical applications where a Logic Analyzer stands out, is when you want to capture a lengthy communication stream, decode this and save the result to a file. Here is the easiest way to do this.

Start off with a decent connection between the SmartScope and the device to test. Preferably using the Logic Analyzer cable, or by making sure signal wires are separated and Ground is nearest to the highest-frequency wire. Make sure there's a perfect Ground connection between both boards.

Set the triggering condition, typically a rising or falling edge on one of the signals. Make sure signals that don't matter for the trigger are set to 'X', otherwise the trigger condition might never occur.

Next, arrange your Panorama (the RAM view at the top of the screen) such that it can fit the full communication you want to analyze. Typically you'll want to move the trigger position (the vertical red line) to the far left of the panorama. If available, the optimum way to do this is by running the 'scope in Auto triggering mode and generate a couple of sample communications.

Finally, select Single trigger mode and wait for the next communication. As soon as the trigger fires, the full content of the RAM will be transferred to your host, allowing you to browse it at your own pace, and add decoders as you see fit.

If you need such a long Panorama that your sample rate drops below 100 MS/s, follow *Menu → System → Acquisition depth → 4MS*. This kluge allows you to use the full depth of the RAM, which will obviously take longer to transfer to your host.

Mixed mode to the rescue!

In some cases you view perfect signals on your screen, but your chips are not reacting to them. Possibly, your communication is using 3.3-V logic while your chip is expecting 5-V signals. Although this yields perfect signals in the logic analyzer, they are 'invisible' to the 5-V chip. Or the other way around, where everything works fine but you see unexpected spikes in your logic analyzer screen.

That's when you want to have both a logic analyzer and an oscilloscope. And that's exactly what the SmartScope is. Follow *Menu → Mixed mode*, which gives you one analog channel in addition to the full Logic Analyzer.

Both analog and digital channels are measured synchronously, so when you attach your analog probe to one of the digital probes you can see both the analog shape of the signal and the digital result as shown in **Figure 5**. Notice that the portion shown in the large graph is less than 1% of the data contained in the RAM, providing plenty of detail for both the analog and digital waveforms. ⏮

(160282)

Web Link

[1] http://wiki.lab-nation.com/index.php/Creating_your_own_Protocol_Decoder



Figure 5. Mixed mode, with Channel A attached to SDA.

HomeLab Helicopter

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

Open counter of the Third Kind



Maker Store Berlin,
Danziger Str. 22, 10435 Berlin

In December 2016 a rather surprising event took place in Berlin. It was neither artistic nor sports- nor food-related — although beverages and pizza were being served during the event — but the opening of a store. And not just 'a store', but the world's first Arduino Store. As far as I know, and where I have lived, electronics shops of the window & cash register type only go out of business, and they have been doing so for some 35 years now. Virtual electronics shops are flourishing, popping up all over the Internet, but opening an electronics store with a counter and paying someone to stand behind it is, well, nothing short of a new — or should I say revolutionary? — concept. In the new Arduino Store you can buy Arduino products (duh), but not just that, as it doubles as a maker space where workshops are being organized. Is this the beginning of a new trend? Will electronics shops soon flower on every street corner like in the old days? Or will this store go down the same road as its predecessors?



Tips & Tricks

Q. How can you tell if you're working with AC or DC power?

A. *If it's AC, your teeth chatter when you hold the conductors. If it's DC, they just clamp together.*

Semiconductor jungle: eat or be eaten



Big fish eat Little Fish,
courtesy of www.catherineswenson.com

If you are an electronics industry watcher or a mere newspaper reader, you will have noticed the wave of mergers and takeovers that have been going on over these past few years. Intel acquired FPGA manufacturer Altera. NXP took over Freescale only to be reeled in itself a few months later by Qualcomm. Microchip gobbled up its competitor Atmel; ON Semiconductor bought Fairchild Semiconductor, and Analog Devices ate Linear Technology. Component distributor Avnet took over colleague Farnell who just had sold Cadsoft (Eagle) to Autodesk (AutoCAD); Mentor Graphics merged with Siemens, and, right after the Brexit, ARM was bought by the Japanese Softbank. Did you know that Amazon has entered this dance too? They own Annapurna Labs from Israel. These are only the high-profile mergers and acquisitions worth several hundreds of millions up to tens of billions of euros/dollars/pounds, but many smaller takeovers have taken place too. Small companies merge to form larger entities, but when they become too big they start an attempt to lose weight. NXP was split off from Philips, Freescale used to be part of Motorola; HP decided to cut itself in two, and at the end of 2016 Samsung announced a possible split too. It is like watching stars and planets colliding and breaking apart again. Is there a black hole hidden somewhere in this universe trying to suck up the whole industry?

Color	1st band	2nd band	3rd band (multiplier)	4th band (tolerance)	Temperature Coefficient
Black	0	0	$x10^0$		
Brown	1	1	$x10^1$	±1%(F)	100 ppm
Red	2	2	$x10^2$	±2%(G)	50 ppm
Orange	3	3	$x10^3$		15ppm
Yellow	4	4	$x10^4$		25ppm
Green	5	5	$x10^5$	±0.5%(D)	
Blue	6	6	$x10^6$	±0.25%(C)	
Violet	7	7	$x10^7$	±0.1%(B)	
Gray	8	8	$x10^8$	±0.05%(A)	
White	9	9	$x10^9$		
Gold			$x10^{-1}$	±5%(J)	
Silver			$x10^{-2}$	±10%(K)	
None				±20%(M)	

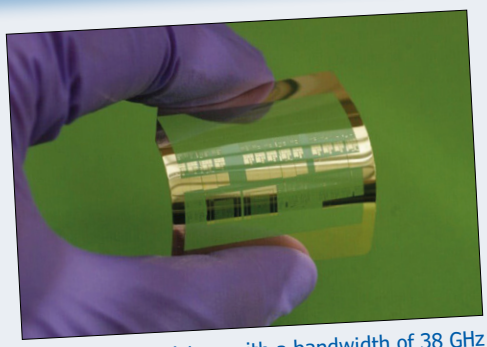
RS-279?

Most Elektor readers know RS-232, many know RS-485, some know RS-422, but who knows RS-279? You! This may come as a surprise, but, since 1963, RS-279 is the official EIA standard for the resistor color coding that you unwittingly memorized over the years. Devised almost a century ago it became an IEC standard in 1952 that also covers capacitors. SMD parts don't use color coding, but letter and digit coding. The current international standard defining marking codes for resistors and capacitors is IEC 60062:2016.

The PCB printer — yet another useless machine?

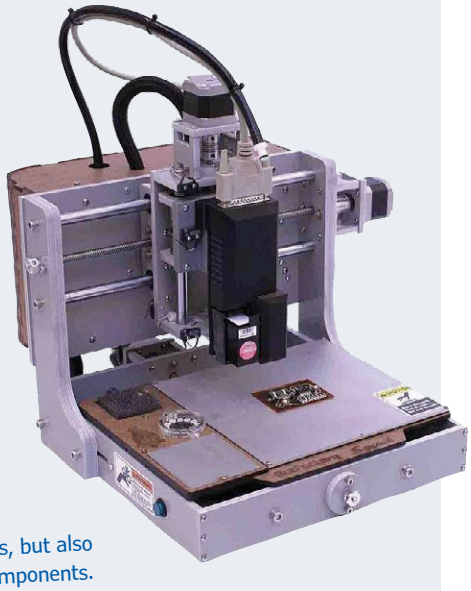


The Voltera desktop PCB printer & assembler combines design and functionality.



These tiny transistors with a bandwidth of 38 GHz were engraved and printed on a flexible substrate (photo courtesy of Jung-Hun Seo).

Did you know that electronics would not be where it is today without a technique developed some 500,000 years ago? I don't mean the invention of silicon which dates back even further, a few billion years or so, but engraving, "the practice of incising a design onto a hard, usually flat surface by cutting grooves into it". (Thank you, Wikipedia.) Printed circuit boards, or PCBs, can be made using various engraving techniques, be it chemical, mechanical or optical, and integrated circuits are engraved on silicon wafers. Printing, a more than 2,000 year old invention, has also turned out pretty important for the electronics industry. First of all, because without it you would not have been able to read this fine piece of prose, but secondly because PCB manufacturing relies on it. The 'P' in PCB is currently extra hot thanks to the development of printers capable of producing PCBs without engraving, simply by squirting conductive ink on a hard, usually flat surface. Printers that print electronic circuits instead of circuit boards are being developed too. Work is in progress to print a simple 4-bit microcontroller directly on cardboard. The goal is to create intelligent packaging, or smart clothes, or some other clever doohickey. Today people wonder why on earth you'd want a 4-bit processor, but in 1971, when Intel launched their 4004 4-bit powerhouse, it was a revolution that kept fighter planes up in the air. What would you prefer to have in your homelab? A printer that prints printed circuit boards that you must populate yourself, or one that spits out fully functional circuits? Maybe we should just skip the PCB printer altogether and concentrate on the electronics printer?



The Squink by BotFactory not only prints PCBs, but also places and solders components.



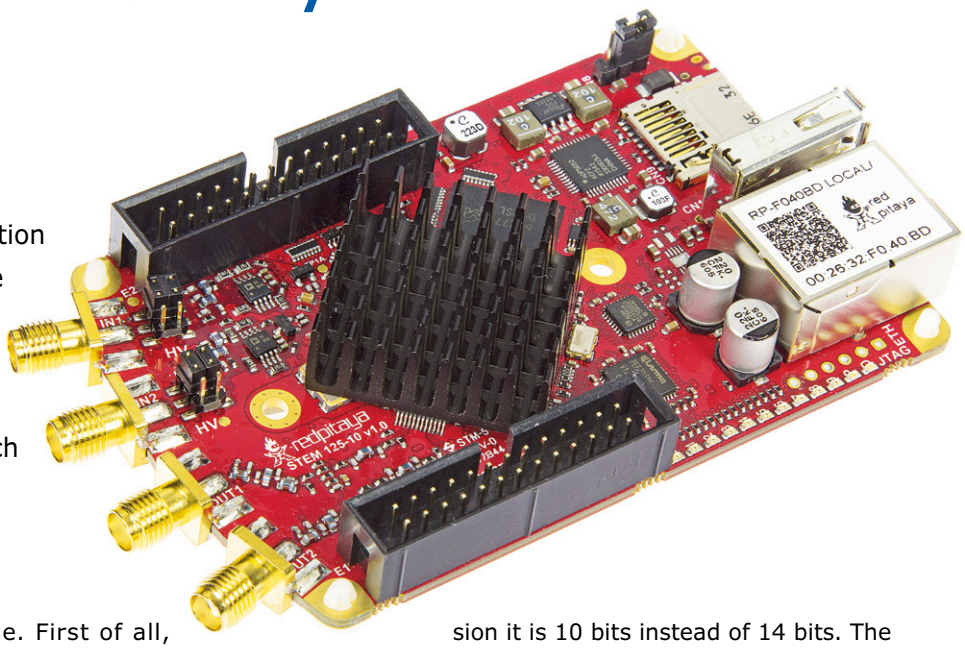
Must-have homelab tool

Guitar plectrum or pick to pry open maker-unfriendly gadgets without damaging them (too much). Keep losing them? Get yourself a plectrum punch.

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

Review:
STEMlab125-10
The little brother of the original Red Pitaya

By **Harry Baggen** (Elektor Labs)



The well-known Red Pitaya instrumentation and development board is now available under a new name in two different versions: a 10-bit version and a 14-bit version. Here we report on our initial experience with the 10-bit version, which goes by the name STEMlab125-10.

Red Pitaya has become a familiar name in the electronics world. This credit-card sized red board is a powerful instrumentation platform with a built-in signal generator as well as a versatile FPGA development system for your own applications. We have already published several articles in Elektor about applications for the Red Pitaya, along with a separate book [1].

Now the engineers at Red Pitaya have updated and expanded their product line. First of all, the name has been changed to STEMlab (STEM is an acronym of "science, technology, engineering and mathematics" and is often used in the educational sector to designate the scientific and engineering disciplines). The original Red Pitaya has been renamed STEMlab 125-14. It also has a little brother now: the STEMlab 125-10. They differ in the resolution of the A/D and D/A converters on the board. With the junior version it is 10 bits instead of 14 bits. The STEMlab 125-10 also has less RAM than the -14 version, but for many applications that is not important.

Fewer bits

I spent some time playing with a prototype of the STEMlab 125-10 as well as a series production board, and I hardly noticed that it is a junior version of the Red Pitaya. That's because both versions work with the same apps and the mea-

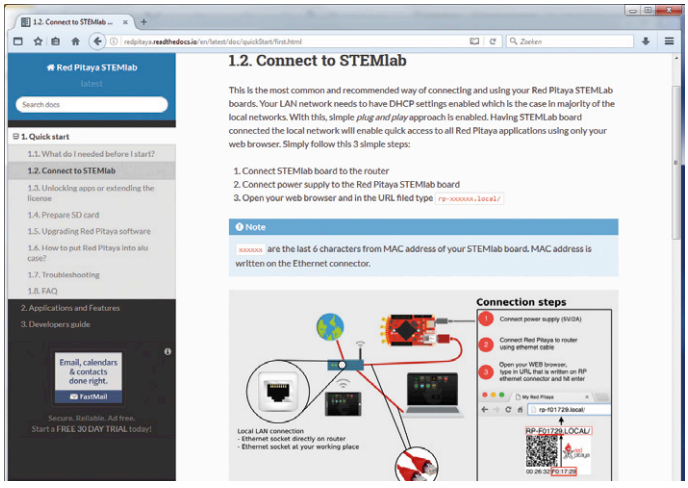


Figure 1. The documentation on the Red Pitaya website.

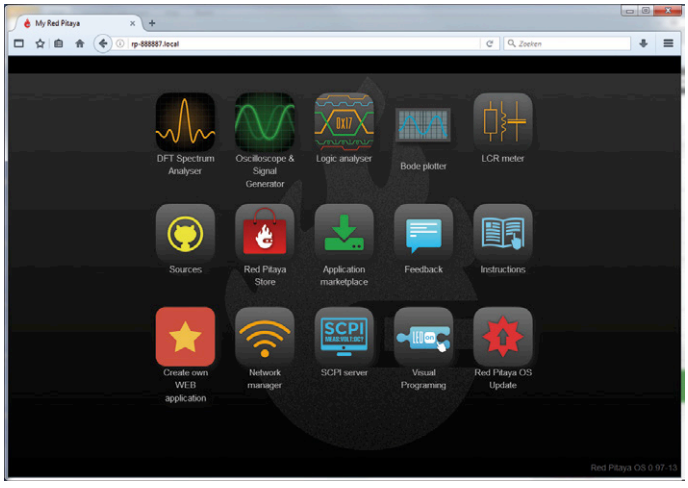


Figure 2. The updated user interface with the various app icon tiles.

sured signals look the same. The most obvious physical difference is that the -10 version has fewer connectors.

Although 10 bits may seem like a lot less than 14 bits with the top-end version, you should bear in mind that most stand-alone and USB oscilloscopes in this price class (and a bit higher) work with 8-bit converters. From that perspective, 10-bit resolution gives you four times as much detail. For oscilloscope functions, 10-bit resolution is enough to produce very detailed signal waveforms on the screen. The main target group for 14-bit resolution is people who write their own apps and want to make very precise measurements.

A big advantage of the new STEMlab devices is that now you get a matching AC adapter and a pre-programmed microSD card together with the board. That makes it a lot easier to get started than with the original version, where you first had to download a software image file and copy it to an SD card. The most frequently used apps are also included on the memory card: oscilloscope, signal generator, logic analyzer, Bode analyzer and LCR meter. That means you don't have to download them separately. By the way, the apps for the oscilloscope, signal generator and spectrum analyzer are the previous Pro versions, while the other apps have a different layout and user interface.

Quick start

If you are using a STEMlab board for the first time, it's a good idea to start by reading the documentation (**Figure 1**), which can be accessed via the Help button on the website [2]. This documentation is a lot clearer than the previous version, and after the first few pages you already have the device up and running. By the way, it's a good idea to connect the board directly to your router through an Ethernet cable. Identifying the board in the network is now much easier than it used to be – you only have to enter a few digits of the MAC address in a command on the browser. That brings up the user interface with a set of app tiles (**Figure 2**). There are icons for the pre-installed apps and several other icons for a number of new functions, including a network manager and function for updating the OS from the browser, and of course there is an icon for the store where you can download more apps.

If you want to work with a Wi-Fi link, you can use the network manager to register the board in the Wi-Fi network and enter the password, after which you see the IP address where you can access it on the network.

As far as I could see, nothing has changed in the existing apps. For a description of the oscilloscope, signal generator and spectrum analyzer functions, see our article "New Apps for the Red Pitaya" [3]. For these apps it does not matter whether the connected board is a 10-bit or 14-bit STEMlab version; the visible results on the monitor are the same in both cases. Extensive descriptions of the apps are provided in the documentation.

Conclusion

The STEMlab 125-10 is a nice little brother for the original Red Pitaya, and for most applications it is just as good as its bigger brother. Thanks to the preprogrammed SD card, you can get started right away and you already have the most frequently apps on board. A variety of packages with the STEMlab 125-10 or the STEMlab 125-14 are available, ranging from a board without accessories (except an AC adapter and SC card, which always come as standard) to expanded kits The accessories can also be bought separately, so you can enhance an existing board with whatever you need: an enclosure, an LCR module, a set of probes or something else.
For most electronics hobbyists and professionals who do not already have a Red Pitaya, the Diagnostic Kit with the STEMlab125-10 is a very attractive package. Along with the STEMlab 125-10, it includes an AC adapter, a preprogrammed microSD card, two oscilloscope probes, a Wi-Fi dongle, two SMA/BNC connectors, two 50-ohm terminators, two SMA tee adapters, nine probe clips with a connecting cable for the logic analyzer, an Ethernet cable and a transparent plastic enclosure. That puts a very complete measuring station on your bench.

(160366)

Web Links

[1] www.elektor.com/red-pitaya-for-test-and-measurement

[2] www.redpitaya.com

[3] www.elektormagazine.coml/150595

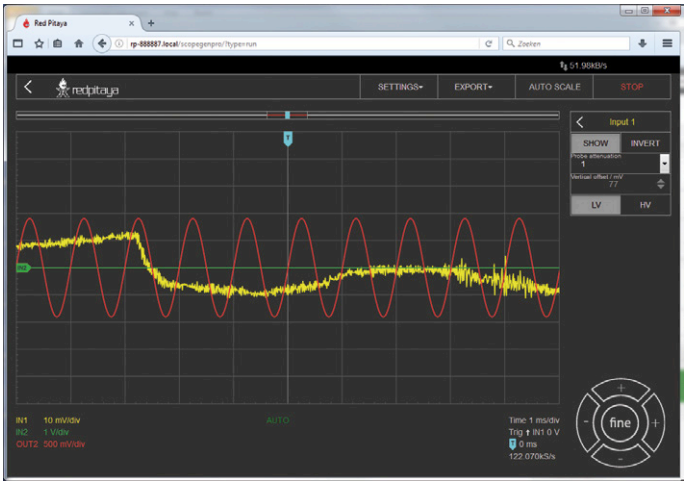


Figure 3. The oscilloscope app included with the board is the same as the previous Pro version.

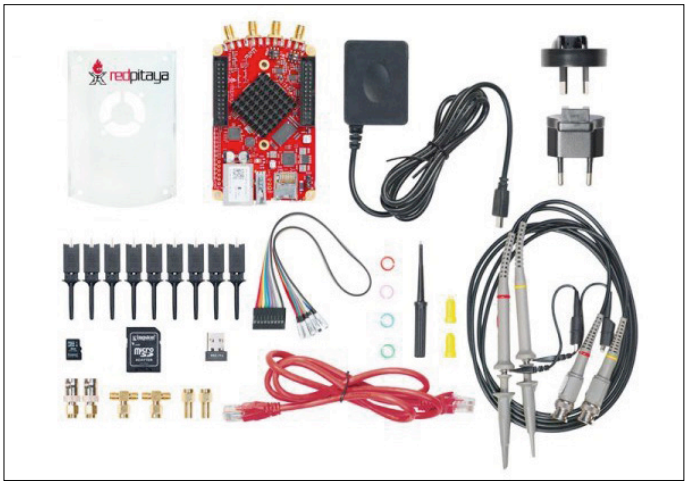
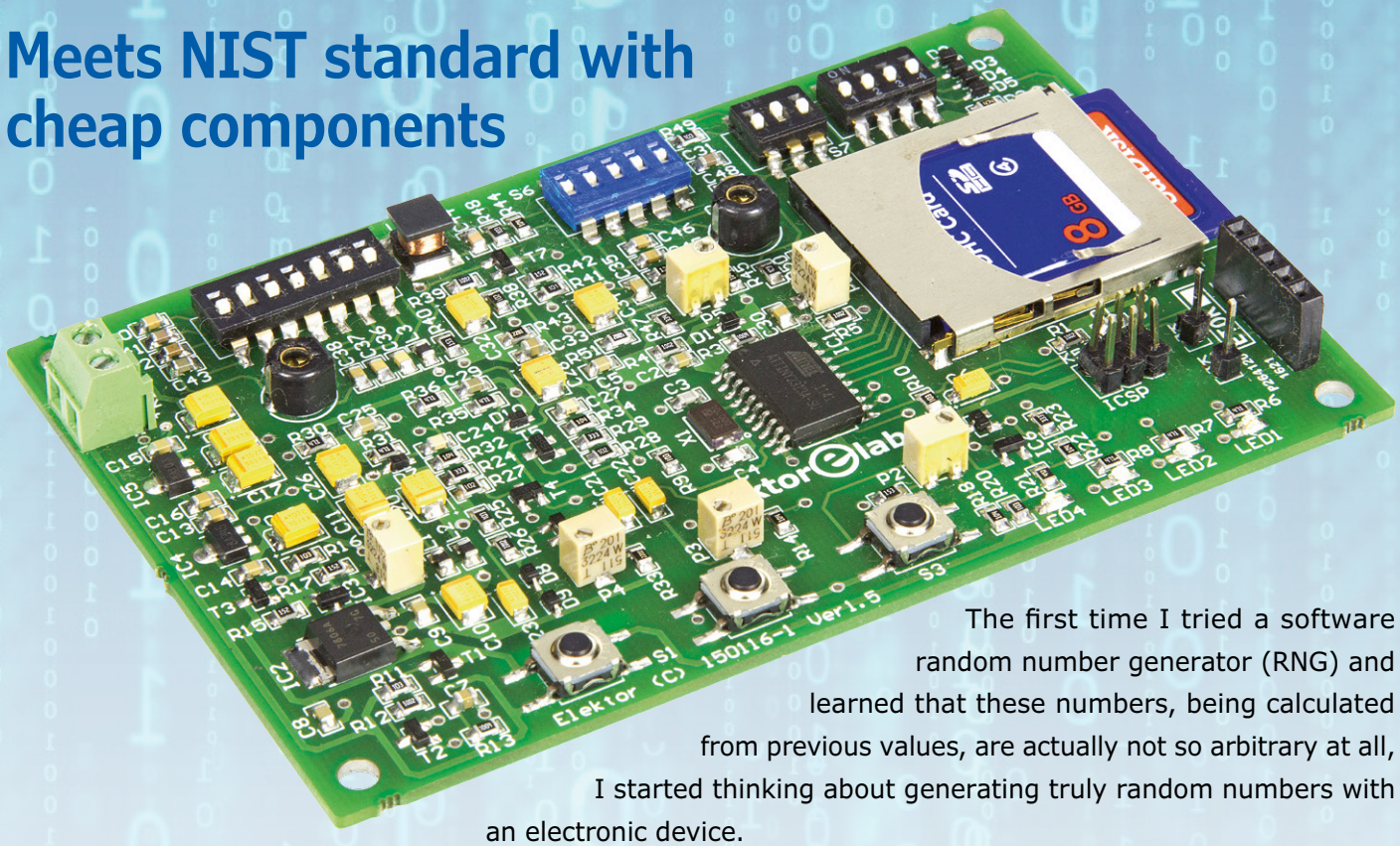


Figure 4. The Diagnostic Kit puts a very complete measuring station on your bench.

eLABS PROJECT

Truly Random-Number Generator

Meets NIST standard with cheap components



The first time I tried a software random number generator (RNG) and learned that these numbers, being calculated from previous values, are actually not so arbitrary at all, I started thinking about generating truly random numbers with an electronic device.

PROJECT INFORMATION

Computers& Internet

security | cryptography | gaming

entry level
intermediate level
➡ expert level

4 hours approx.

SMD Soldering
AVR programmer
adjustable power supply

€150 / $155 / £125 approx.

By **Luka Matić** (Croatia)

Later, when I learned the basics of cryptography, I discovered more reasons for a good Random Number Generator (RNG) to be a useful thing to have. Secure data encryption, for instance, is almost impossible without one. Gaming and gambling applications also require top-class RNGs [2], and—for the mathematically inclined—did you know that you can estimate the number π by using random numbers? Just search the Internet for 'Buffon' and 'Pi' to find out how. High-quality random signal generators are available commercially but tend to cost a lot of money, so I decided to build my own out of cheap and easily available components.

The circuit

For what follows, please refer to the schematic shown in **Figure 1**.

Because noise is random by nature, a noise source forms the basis of the RNG. The randomness of different types of electronic noise (shot, thermal, flicker, popcorn, avalanche, etc.) are well known. My RNG uses avalanche noise — which can be obtained from zener diodes — chosen because of its wide bandwidth and much higher amplitude compared to other noise sources. Two 12-V zener diodes (D7 and D10) are used to generate two noise signals. A differential amplifier built around T4, T5 and T6 amplifies the difference of both signals, to improve the chances that the noise signal is substantially random. This amplifier also removes common-mode signals due to interference and disturbances that may affect both diodes.

The next stage (T7) amplifies the noise signal further, up to 0.5 to 1 Vpp, which is enough for the analog comparator integrated in the ATtiny2313 micro. It also

serves as an adjustable corrective filter that allows the RNG to meet the quality criteria for random signals set by standardization organizations worldwide (see **inset**).

The output of the analog comparator is sampled at a rate of around 800 kHz. Consequently, certain patterns in the random bit stream may be correlated to subharmonics of the sample rate present in the noise signal. Bit patterns 0x00 and 0xFF correspond to a subharmonic of 50 kHz; patterns 0x0F and 0xF0 correspond to 100 kHz; patterns 0x33 and 0xCC to 200 kHz; and patterns 0x55 and 0xAA to 400 kHz. Too many or too few occurrences of a certain bit pattern indicate that the frequency response of the corrective filter must be adjusted to decrease or increase the gain at the 'offending' frequency. This can be done

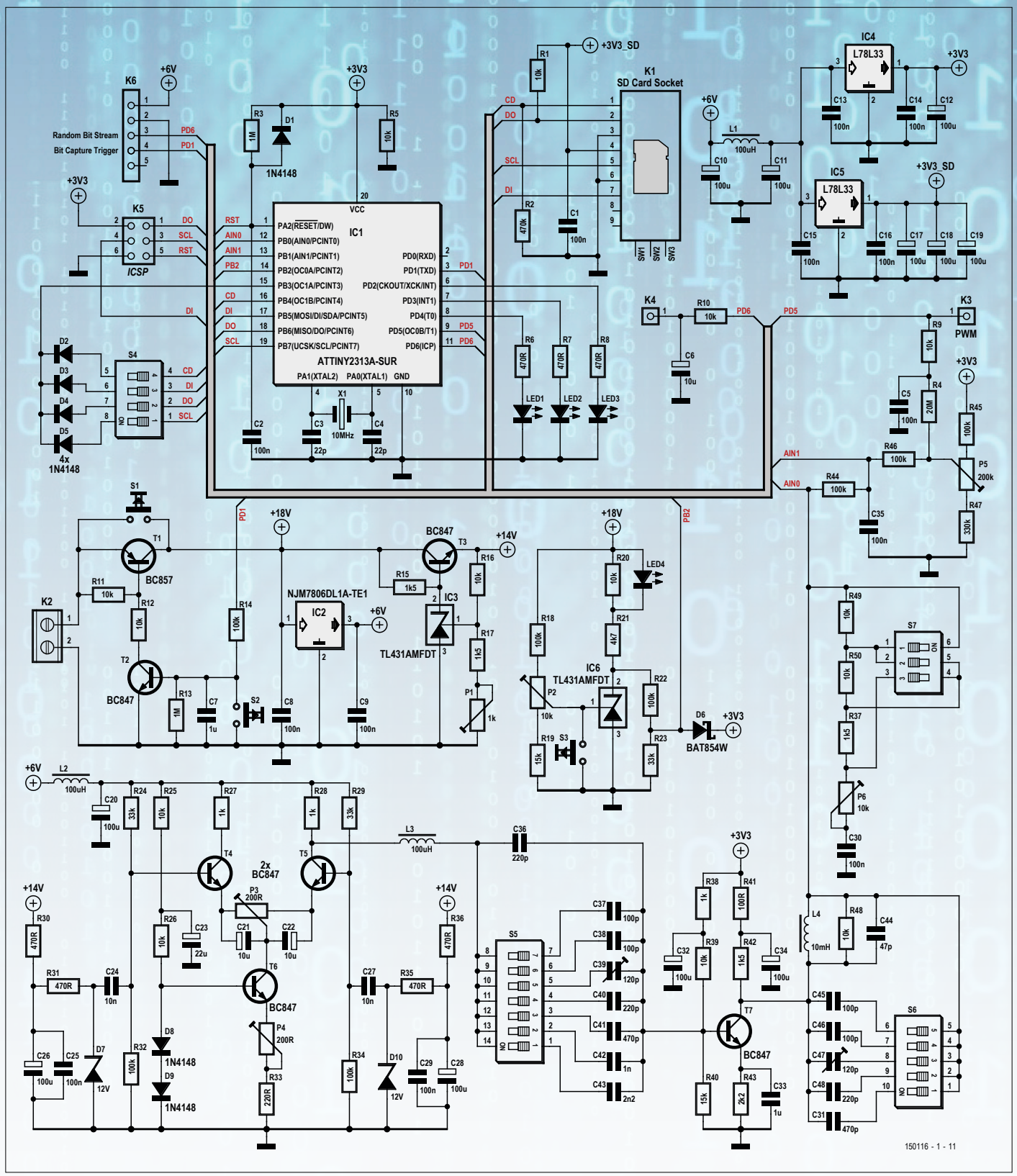


Figure 1. The Random Number Generator or RNG cleverly combines analog and digital electronics to produce a truly random bit stream.

Table 1. Add the values for every switch of S4 in the ON position plus 16 MB to determine the file size. With all switches set to OFF, the file size is 16 MB.

Switch	File size
S4-1	2048 MB
S4-2	1024 MB
S4-3	512 MB
S4-4	256 MB

with DIP switches S5, S6 and S7 and by adjusting trimmers C39, C47 and P6. The sampled output of the analog comparator is recorded on the SD card in one, large file. The SD card must be formatted as FAT32 (see inset). Before the RNG is switched on, DIP switch S4 must be set to define the desired size of the file, ranging from 16 MB to 3.8 GB (see **Table 1**). After a successful recording a file named 'random.hex' is created on the SD card.

The stream of random bits is also output on PD6 so that its quality can be monitored in real time with the Bit Stream Analyzer (see below), a tool that measures certain bit-pattern-occurrence frequencies — very handy for adjusting the filter around T7. To allow synchronizing to this bit stream, PD1 produces a short

COMPONENT LIST RNG

Resistors

Default: 0805, 100mW, 5%
R1,R5,R9,R10,R11,R12,R16,R20,R25,R26,R39, R48,R49,R50 = 10 kΩ
R2 = 470kΩ
R3,R13 = 1MΩ
R4 = 20MΩ
R6,R7,R8,R30,R31,R35,R36 = 470Ω
R14,R18,R22,R32,R34,R44,R45,R46 = 100kΩ
R15,R17,R37,R42 = 1.5kΩ
R19,R40 = 15kΩ
R21 = 4.7kΩ
R23,R24,R29 = 33kΩ
R27,R28,R38 = 1kΩ
R33 = 220Ω
R41 = 100Ω
R43 = 2.2kΩ
R47 = 330kΩ
P1 = trimpot, 1 kΩ, 11-turn, SMD
P2,P6 = trimpot, 10kΩ, 11-turn, SMD
P3,P4 = trimpot, 200Ω, 11-turn, SMD
P5 = trimpot, 200kΩ, 11-turn, SMD

Capacitors

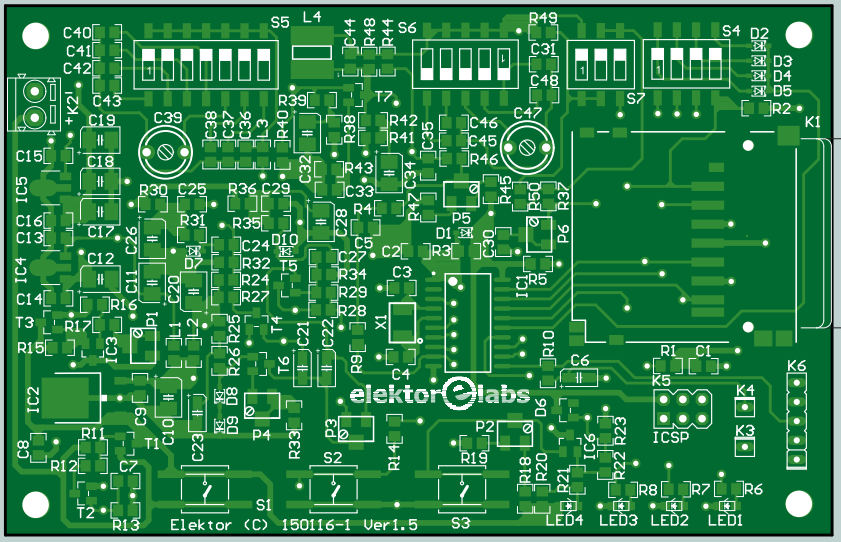
Default: 0805
C1,C2,C5,C8,C9,C13,C14,C15,C16,C25,C29, C30,C35 = 100nF
C3,C4 = 22pF
C6,C21,C22 = 10µF, 6.3V, case A
C7,C33 = 1 µF
C10,C11,C12,C17,C18,C19,C20,C26,C28,C32 ,C34 = 100µF, 10V, case B
C23 = 22µF, 10V, case A
C24,C27 = 10nF
C31,C41 = 470pF
C36,C40,C48 = 220pF
C37,C38,C45,C46 = 100pF
C39,C47 = 120pF adjustable capacitor (eBay is your friend)
C42 = 1nF
C43 = 2.2nF
C44 = 47pF

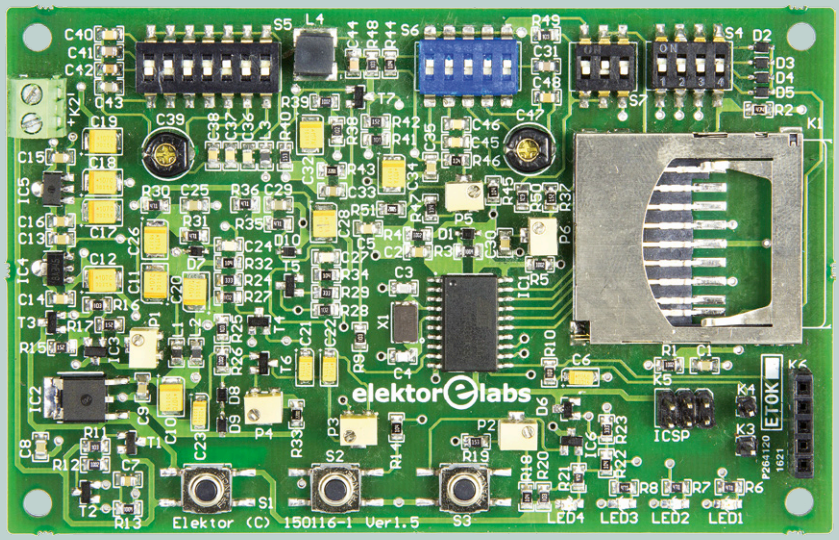
Inductors

L1,L2,L3 = 100µH, 0805
L4 = 10mH, 2220

Semiconductors

IC1 = ATtiny2313V-10SUR, programmed
IC2 = L7806CD2T-TR (or NJM7806DL1A-TE1)





IC3,IC6 = TL431AMFDT
IC4,IC5 = L78L33ACUTR
D1,D2,D3,D4,D5,D8,D9 = 1N4148WS
D6 = BAT854W
D7,D10 = BZX384-C12
T1 = BC857CW
T2,T3,T4,T5,T6,T7 = BC847BW
LED1,LED4 = green, 2x1.25 mm
LED2 = red, 2 x 1.25mm
LED3 = yellow, 2 x 1.25mm

Miscellaneous

X1 = 10 MHz quartz crystal, 18pF, 5.0x3.2mm

S1,S2,S3 = Tactile switch
S4 = DIP switch, 4-way
S5 = DIP switch, 7-way
S6 = DIP switch, 5-way
S7 = DIP switch, 3-way
K1 = SD card connector type SDBMF-00915B0T2
K2 = 2-way PCB screw terminal block, 3.5 mm pitch
K3,K4 = 1 pinheader pin
K5 = 6-pin pinheader (2x3), 0.1" pitch
K6 = 5-way pinheader, 0.1" pitch

(<1 µs), active-low pulse at the start of every bit. These pulses are too short to accidentally power off the RNG.

Power

To avoid introducing interference sources that may degrade the quality of the RNG output, the device is supposed to be powered from two 9-V batteries in series. The 18 volts we get from them are needed for the 12-V zener diodes D7 and D10. This type of diode is known to produce a good amount of noise at a reasonable voltage.

Transistors T1 and T2 along with output PD1 act as a power-on (and auto power-off) flip-flop. Pressing button S1 switches the circuit on, making the MCU come alive so that it can activate port PD1 and keep its own pants up.

The circuit around IC6 functions as a low-battery indicator, and informs the MCU when the voltage drops to about 15.4 V. When this happens, the MCU safely terminates writing the SD card before shutting down the circuit (by means of PD1) to prevent both data corruption and excessive draining of the batteries. A low-battery condition can be simulated with S3, which functions as a soft-stop button used to stop the recording at any time while keeping the data. S2 is a hard-stop button which turns off the power immediately. Pressing it while a recording is in progress is likely to result in a corrupted file on the SD card. Use it after adjusting trimmer P5 or when something goes wrong.

The LEDs

After pressing S1 to power on the RNG, and if the battery voltage is high enough, LED4 will turn on. Then LED3 comes on, followed by LED1 to show that the SD card was correctly initialized. LED1 then turns off, meaning that the recording of the random bits has started; it will turn on again when the file of the desired size has been written and from then on it will stay on as long as the battery voltage remains high enough. LED3 will flash repeatedly during recording: a short flash at the end of each cluster written, and a long flash when the FAT table is updated. LED2 indicates if any problems occurred during SD card communication.

Bit Stream Analyzer

Generating quality random data is not easy, which is why the RNG features several adjustable resistors and capacitors that need to be set properly. Even though it is possible to record a bit stream, analyze it, readjust the RNG, and repeat this process as often as necessary, doing so is not very practical. For this reason a special tool was developed: the Bit Stream Analyzer (BSA). With the BSA things become more convenient as it allows you to observe the tuning effects in real time. The BSA consists of an ATMega8 and a 2x16-character alphanumeric display (Figure 2) connected to K6 of the RNG. For its input it takes the random bit stream produced by the RNG on PD6

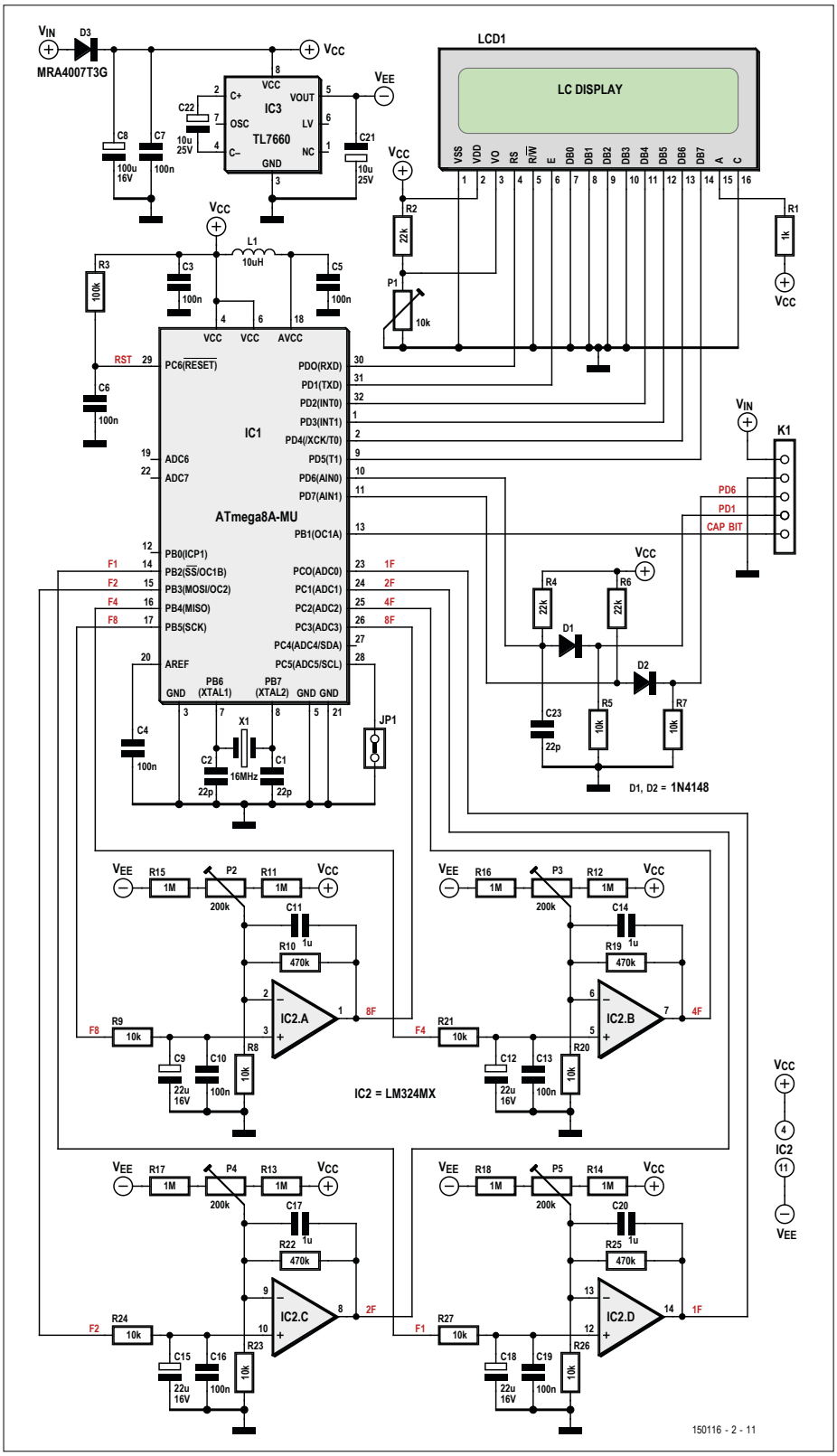


Figure 2. The Bit Stream Analyzer or BSA allows real time tuning of the RNG.

Even better...

The degree of randomness of the data produced by the RNG can be improved even further when two recorded sequences of the same length are combined by 'XOR-ing' them on a bit by bit basis. This is similar to encrypting one sequence with the other, also called One-Time Pad (OTP) encryption. The result is a sequence with a much flatter histogram (i.e. having a variation of just ±2-3%) that can pass certain NIST tests for much longer sequences. For instance, the 'Runs' test now passes smoothly for sequences longer than 500,000 bits, as opposed to 50,000 bits that can be achieved without post-processing.

and a bit synchronization signal on PD1. The BSA is a kind of spectrum analyzer tuned to the first four subharmonics of the RNG sample rate (800 kHz). It displays the amplitudes labelled as f1 (50 kHz), f2 (100 kHz), f4 (200 kHz) and f8 (400 kHz), and it displays the 0/1 ratio.

The MCU is helped by a kind of 'analog computer' (IC2) to determine the levels of the four subharmonics because the ATmega8 is not fast enough to do all the work itself. The MCU only handles the 0/1 ratio, and 'subcontracts' the subharmonics to the analog computer. The MCU outputs a short, constant-length pulse on PB2 to PB5 for each occurrence of a characteristic pattern 0x00, 0x0F, 0x33 or 0x55 and reads the processed analog value at PC0 to PC3.

The number in the upper right corner of the display is the 0/1 ratio which should be around 512. It is controlled by P5 of the RNG (not of the BSA) and the PWM auto-offset. The four subharmonics should have magnitudes in the same ballpark (i.e. within ±10%), the exact values do not matter much.

Before the BSA can be used, the DC offsets of the operational amplifiers must be balanced:

1. Fit jumper JP1 to connect PC5 to GND and then power up or reset the BSA. The MCU will enter the Zero-offset Adjustment Mode, indicated by the character 'Z' appearing on the LCD.
2. Adjust trimpots P2 to P5 to set the values of f1, f2, f4 and f8 to a value slightly above zero; between 10 and 20 is fine.
3. Remove jumper JP1, and reset (or power cycle) the BSA.

Tuning the RNG

Building a precision device with low-cost components usually results in one or more parameters that require adjusting manually. The RNG has several of them:

Trimpot P1

Adjust to obtain 14 V at the emitter of T3.

Trimpot P2

Determines the voltage that triggers the 'low battery' warning (LED4 turns off). If the RNG is powered by two 9-V NiMh batteries, the warning should trigger at 15.4 V (because such batteries contain seven 1.2-V NiMh cells, and when discharged to 1.1 V per cell this totals to 15.4 V). Connect an adjustable power supply on K2; vary its voltage between 15 V and 16 V. Adjust P2 to make LED4 turn off when the voltage at connector K2 drops to 15.4 V.

COMPONENT LIST BSA

Resistors

Default: 0805, 100mW, 5%
R1 = 1kΩ
R2,R4,R6 = 22kΩ
R3 = 100kΩ
R5,R7,R8,R9,R20,R21,R23,R24,R26,R27 = 10kΩ
R10,R19,R22,R25 = 470kΩ
R11,R12,R13,R14,R15,R16,R17,R18 = 1MΩ
P1 = 10kΩ, trimpot, SMD
P2,P3,P4,P5 = trimpot, 200kΩ

Capacitors

Default: 0805
C1, C2,C23 = 22pF
C3,C4,C5,C6,C7,C10,C13,C16,C19 = 100nF
C8 = 100µF, 16V, radial can SMD
C9,C12,C15,C18 = 22µF, 16V, radial can SMD
C11,C14,C17,C20 = 1µF
C21,C22 = 10µF, 25V, radial can SMD

Inductors

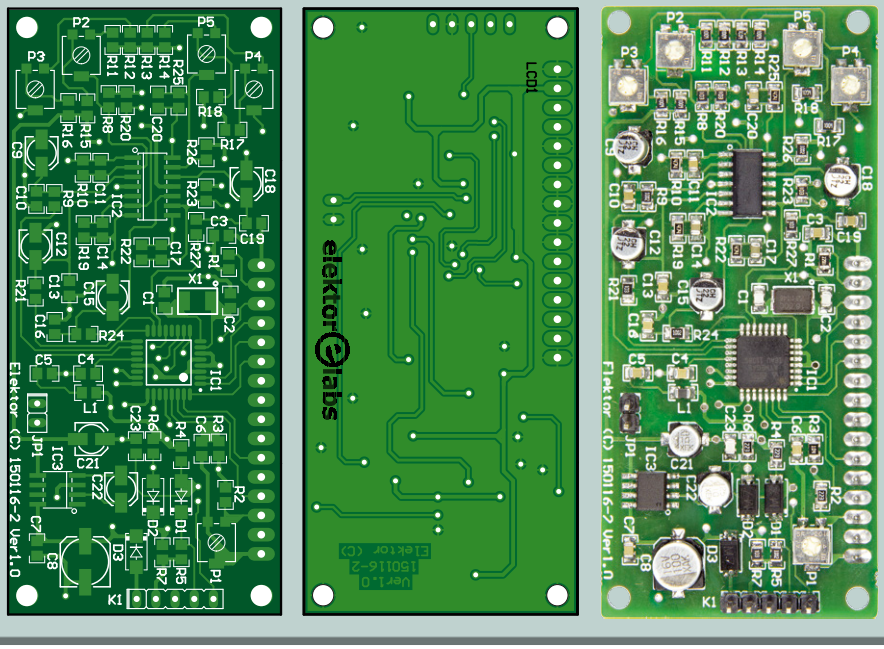
L1 = 10µH, 0805

Semiconductors

D1,D2 = 1N4148WS
D3 = MRA4007T3G
IC1 = ATmega8-16AU
IC2 = LM324MX
IC3 = TL7660CD

Miscellaneous

X1 = 16 MHz quartz crystal, 18pF, 5x3.2mm
JP1 = 2-pin pinheader, 0.1" pitch
Jumper, 0.1" pitch
K1 = 5-way pinheader socket, 0.1" pitch (solder side)
LCD1 = alphanumeric, 16 columns, 2 lines
16-pin pinheader (for LCD), 0.1" pitch
16-way pinheader socket (for LCD), 0.1" pitch



Trimpots P3 and P4

The collector currents of transistors T4 and T5 must be 1 mA. Adjust trimpots P3 (symmetry) and P4 (total current) to obtain a voltage drop of 1 V across both resistors R27 and R28.

Trimpot P5

1. Turn off the power (press S2, hard stop);
2. Hold S3 (soft stop), then press S1 (start) to enter Test Mode. LED3 turns on;
3. Connect an oscilloscope (or a DC voltmeter) to PD5 to monitor the PWM signal that compensates DC drift. Adjust P5 to make the PWM signal stabilize at around 50%;
4. The Test Mode runs for about 40 seconds, then the PWM value is written to the internal EEPROM of the MCU and LED1 is turned on. Restart the test mode if you need more time. If the PWM saturates at 0 or 100% LED2 turns on and the PWM value is not written into the EEPROM. If this happens, restart Test Mode.

S5, S6 and S7

Figure 3 shows the transfer function of the circuitry around T7. S5 together with its capacitors control the lower cut-off frequency. Use S6 to adjust the filter's resonant frequency as in:

$f_{res} = 1 / (2\pi \times \sqrt{(L4 \times C_{S6})})$ [Hz]

where S7 sets the maximum dip of:

$20 \times \log_{10}(R_{S7} / (R_{S7} + R48))$ [dB].

Auto-trimming

The last PWM value is always kept in internal EEPROM. In normal random data recording mode, the PWM duty cycle is adjusted constantly. At the end of a recording (at completion or a soft stop due to a low-battery condition) the current PWM duty-cycle value is written to EEPROM to be used the next time the RNG is powered on. It is therefore recommended to first record one small 16 MB file (and discard it) prior to recording a file of the desired size so that the PWM signal can be adjusted by the MCU. When done, set the desired size with DIP switch S4 and restart the RNG immediately. The random data will now have the highest quality because the DC offset will be properly adjusted from the very beginning.

Sanity check

After tuning, record a small 16 MB file and analyze it with a program like WINHEX to create a histogram of occurrences of all 256 possible 8-bit patterns. **Figure 4** shows a spiky, non-uniform histogram for data that's not random enough for use in serious cryptography even though its 0/1 bit ratio is correct. Tuning more carefully brings histograms similar to the one in **Figure 5** within reach.

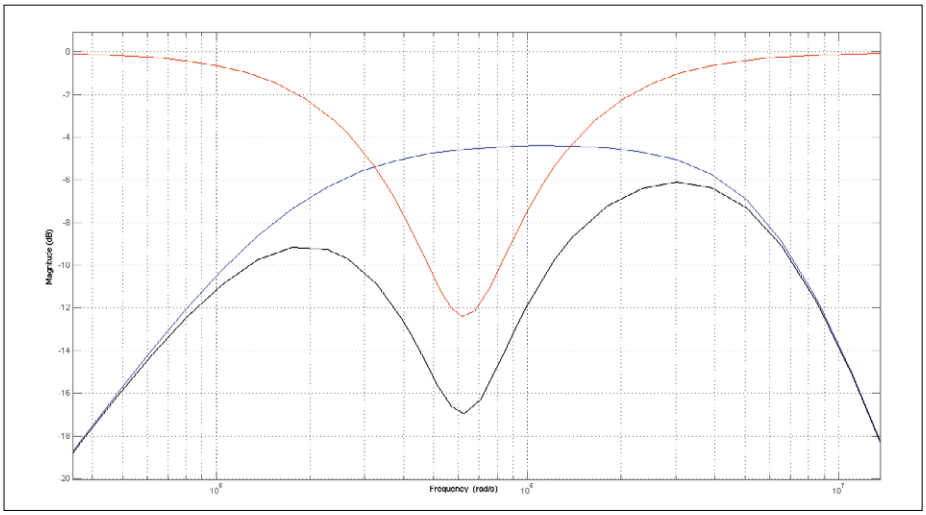


Figure 3. The simulated transfer function of the filter built around T7 (black curve). The blue curve is the transfer function of T7, defined by its bandwidth. The lower cutoff frequency is controlled by S5 and its capacitors C36 to C43. The red curve is the transfer function of the corrective filter L4-R48-CS6-RS7. For clarity, transistor amplification was left out, the real black and blue curves lie about 30 dB higher.

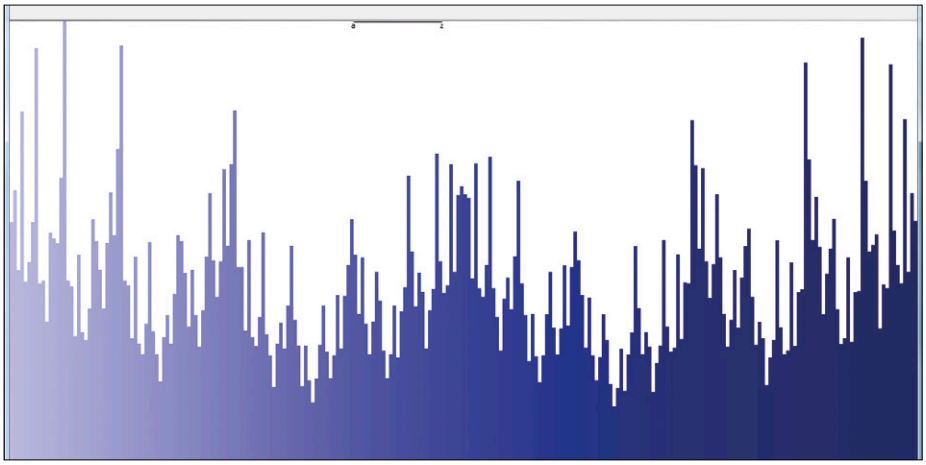


Figure 4. This histogram of the data generated by the RNG reveals a poorly tuned corrective filter. Looking only at the leftmost and rightmost bars (occurrences of respectively 0x00 and 0xFF) the zero-to-one ratio seems fine, however, because the corrective filter is not tuned properly, the occurrences of 0x0F and 0xF0 are almost three times higher than those of 0x55 and 0xAA.

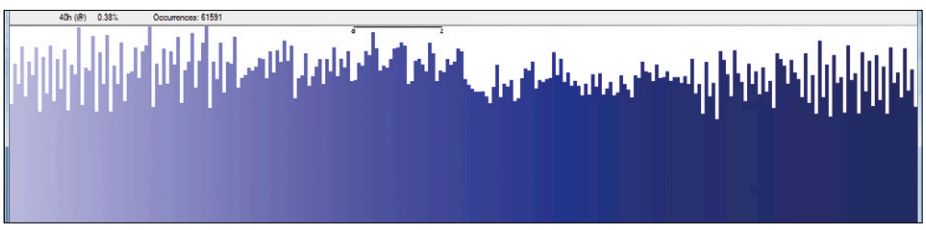


Figure 5. After proper tuning of the corrective filter the histogram is almost flat. Now you have premium-quality random data.

Some SD card subtleties

The (assembly) code for the ATtiny2313 only works with FAT32-formatted SDHC cards; it does not support FAT16 or SD cards that are not SDHC. It doesn't support SDHC cards bigger than 4 GB either, because its 32-bit pointers can address only up to 4 GB. This is not really a problem as FAT32 does not support files larger than 4 GB and writing such files would take too long anyway.
Since an all-singing, all-dancing FAT32 driver is sure consume too much program memory of our tiny MCU, only the most basic FAT32 operations needed to write one single file to the root directory, have been implemented. Because of this simplicity, and because of some safety measures to properly handle soft stops, the size of the final file may not exactly be the size advertised.
It is recommended to format the SD card to a cluster size of 8 KB for two reasons:

- The smaller the cluster size, the more often the FAT table will be updated (every two seconds with sectors of 1 KB). However, the signal to the Bit Stream Analyzer is stopped during FAT table updates, making the analyzer readings drop because its input capacitors discharge. If this happens too often, adjusting the RNG becomes difficult due to the fluctuating readings.
- The larger the cluster size, the longer the time between FAT table updates (more than 60 seconds for sectors of 32 KB). PWM auto-offset correction is applied at every FAT table update, and if the time between updates becomes too long, the auto-offset correction becomes ineffective.

Beware of EMI

Please keep in mind that without proper enclosures, high-precision instruments like the RNG are sensitive to EMI and RF radiation. It is therefore highly recommended to mount the RNG inside a metal (aluminum) enclosure with the circuit Ground properly connected to the enclosure.

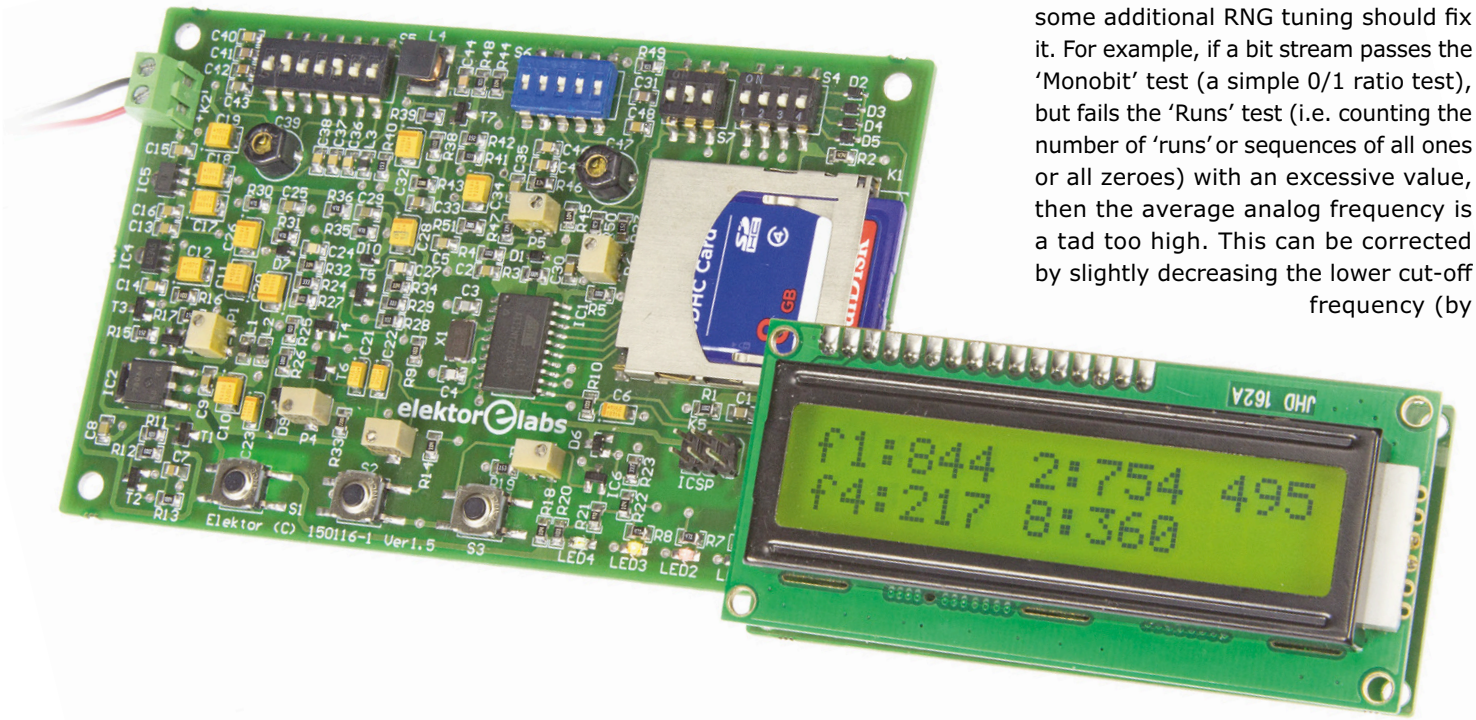
NIST testing

Standards to evaluate the quality of a random bit stream have been set by standardization institutes like the National Institute Standards and Technology (NIST). The NIST has defined 15 tests for randomness that boil down to three simple rules:

1. a truly random bit stream has an equal number of ones and zeros;
2. a truly random bit stream has equal frequencies of occurrences for different bit patterns;
3. no part of the bit stream is mathematically correlated to any other part of the stream.

When carefully tuned to produce histograms like the one in Figure 5, the RNG described in this article passes all 15 NIST tests with ease. The first criterion of randomness is met by carefully adjusting the DC bias with trimpot P5. The second criterion can be met by adjusting the frequency response of the corrective filter with its capacitor and resistor banks (DIP switches S5, S6 and S7) and adjusting trimmers C39, C47, and trimpot P6). Since the source of the random bit stream is random noise, the third criterion is met automatically.

Before diving into the NIST tests, you can try a simple test with a file compression utility like 7-Zip or similar. File compression algorithms try to reduce the redundancy or correlation of data to make it smaller. Since truly random data is completely uncorrelated, it cannot be compressed. The size of the compressed file will therefore be greater than the original file, where the increase in file size is due to file format related issues.
A Windows application which performs NIST testing, pseudo-random number generators (PRNGs) that produce good pseudo-random bit streams for comparison, and sample bit streams can be found at [2].
The NIST test application requires a text file as input, hence binary files must be converted first. The application is very fussy about the exact format of the input file—only the ASCII characters '0' (ASCII code 0x30), '1' (ASCII code 0x31), and space (ASCII code 0x20) are allowed. Every binary digit must be separated by a space; leading or trailing spaces are not allowed. One mistake and the program will reject the file.
If the bit stream fails a certain NIST test, it is always related to one of the three basic criteria listed above, and some additional RNG tuning should fix it. For example, if a bit stream passes the 'Monobit' test (a simple 0/1 ratio test), but fails the 'Runs' test (i.e. counting the number of 'runs' or sequences of all ones or all zeroes) with an excessive value, then the average analog frequency is a tad too high. This can be corrected by slightly decreasing the lower cut-off frequency (by



increasing the capacitor value selected with S5).
With a little tweaking of the trimmers it is not difficult to make the RNG pass all 15 (say, fifteen) NIST tests with much longer bit streams than the minimal required lengths.

Conclusion

Built with cheap, readily available components, the Random Number Generator (RNG) presented in this article is good enough to meet the standards for truly random data set by official international standardization organizations like the NIST. It shows that the DIY approach still pays off—in the true spirit of Elektor Magazine and Labs! This RNG is more than an experimental toy, it is a precision instrument. On the other hand, keep in mind that encryption schemes using it (or that use even better RNGs) can, in theory, be broken with a quantum computer, so neither Elektor nor the author can, and will, assume any responsibility for the privacy of your data—you have been warned.

(150116)



The Author

Luka Matić graduated from the FER Zagreb (Croatia) where he obtained a master's degree in automation engineering. His work experience includes automation of industrial plants, onshore as well as offshore, and of remotely operated submersible vehicles (ROSVs).
As a freelance engineer he has gained a lot of experience with all sorts of electronic devices and apparatus. Luka is not only an author for Elektor Magazine and Labs, he also co-authored a scientific article on advanced friction compensation for electrically driven positioning systems.
He can be contacted at **luka.matic@gmail.com**

Web Links

[1] http://csrc.nist.gov/groups/ST/toolkit/rng/documentation_software.html

[2] www.elektormagazine.com/labs/random-number-generator-150116

[3] www.elektormagazine.com/150116

FROM THE STORE

- 150116-1 PCB
- 150116-2 PCB
- 150116-41 Programmed microcontroller
- 150116-42 Programmed microcontroller

Advertisement

USB Add USB to your next project. It's easier than you might think!

DLP-USB1232H: USB 2.0 UART/FIFO

HIGH-SPEED 480Mb/s



- Multipurpose: **7** interfaces
- Royalty-free, robust USB drivers
- No in-depth knowledge of USB required
- Standard 18-pin DIP interface; 0.6x1.26-inch footprint

DLP-IO8-G
8-Channel Data Acquisition



Only $29.95!

- 8 I/Os: Digital I/O
  Analog In
  Temperature
- USB Port Powered
- Single-Byte Commands

DLP-IOR4
4-Channel Relay Cable

DLP-TH1b
Temp/Humidity Cable

DLP-RFID1
HF RFID Reader/Writer

DLP-FPGA
USB-to-Xilinx FPGA Module

DLP Design

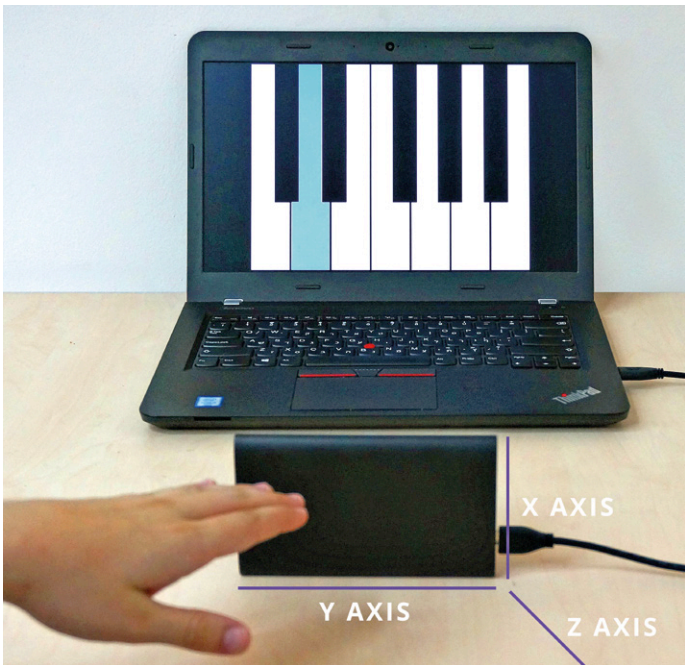
www.dlpdesign.com



Walabot – a 3D Sensor for your Smartphone

Also sees through walls

By **Naftali Chayat,** Vayyar Imaging Ltd. (Israel)



Imagine a sensor that creates a 3D picture of its surroundings without invading on your privacy, that can look below the surface as well as detect minute movements. Walabot can do all that — in a sleek mobile phone-sized form factor.

Even better, it can attach to a mobile phone and act as a portable scanning or monitoring device. Walabot uses radio waves to query its surroundings, be it through air or a material such as a plaster board. Here we demonstrate the capabilities of this novel sensor through two "handwaving user interface" applications: an "air piano" and a "page flipper". But first, a short background to the operation of the device.

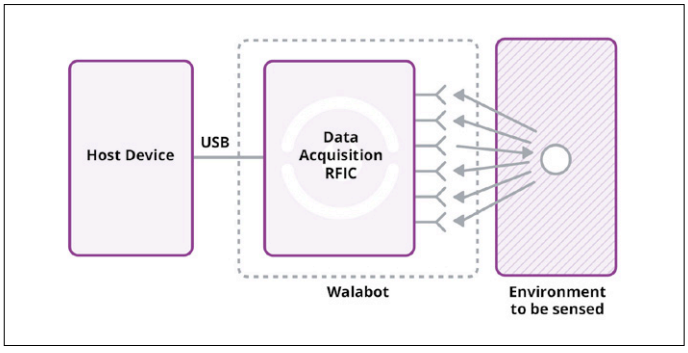


Figure 1. How the Walabot works.

The Walabot sensor

Walabot comes as a bare board or with a case, for integrating into your project. All you need is a USB cable to connect it to a hosting device — A PC, a phone or a tablet.

Walabot is based on the VYYR2401 multichannel radio transceiver chip developed by Vayyar Imaging Ltd. The chip attaches to multiple antennas, which "illuminate" the area in front of the board, and collect the reflected radio waves, see **Figures 1** and **2**. The information is passed to the host device through a USB controller, onto a software driver that collects and processes the data to extract the image data.

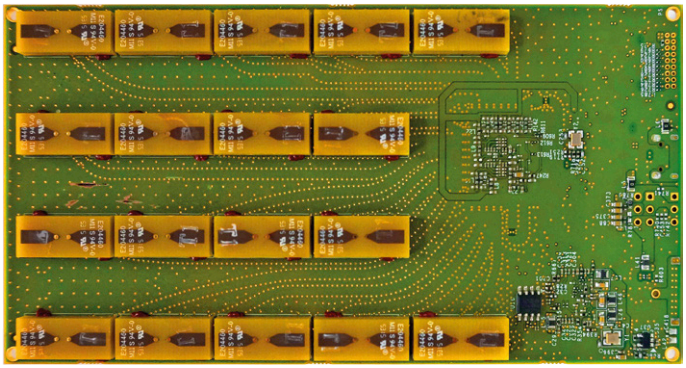


Figure 2. The antenna array allows the Walabot to "see".

The VYYR2401 chip scans through the frequency range of interest in fast progression, so as to not disturb any particular frequency. Very low power is used, below 0.1 milliWatt. The frequency range depends on the model – the US model uses a 3.3 to 10.3 GHz range, while the European model uses a 6.3 to 8.3 GHz range. Fourier theory is then used to convert the frequency responses into time domain – the larger the delay, the farther the object.

The driver software is responsible for the image formation (**Figure 3**). For each antenna pair, the distance to each candidate object's location can be calculated. By aligning the signals in

time for each candidate location and combining the signals, the return from that location is estimated.
Walabot's programming API allows providing data about the surroundings of it in a variety of formats — 3D map, range-azimuth map, or coordinates of the detected object. The API supports three main modes of operation: a short range sensor (such as detecting a pipe in a wall), a regular sensor mode (e.g. detecting a person in a room), and a raw data mode (in which you may process the signals in our own way). An MTI (moving target indicator) filter can be activated to focus on the changes (moving objects) and disregard the static returns (the background). In the example projects we are using the regular sensor mode with Cartesian coordinate output, where we report the location of the user's hand, to interface with the computer. The software API hides from the user all the math wizardry needed to extract the images, and lets the user focus on the application. It supports functions such as detecting object's location, or focusing on moving objects.

Example #1: Piano

So, what does it take to make a piano? First, we need to define the gesture language to relate it to the Walabot's coordinates. When Walabot is oriented as in the head illustration, the X coordinate relates to the "up-down" movements (used for pressing keys), the Y coordinate relates to "right-left" movements (i.e. which key to press), and Z relates to the "in-out" movements. Note that it is important to have the Walabot standing on its wide side with the cable on the right as shown, otherwise it's like playing a piano which is upside down!

By repeatedly invoking the scan operation, the X-Y-Z location of the hand is reported, and now we need to take action. First, we want to provide visual feedback to the user, so that the user can see which key he's pointing at (the Y-Z parameters), and whether he's "pressing" a key (X parameter). Then, of course, whenever a key is pressed, we want to play a sound! Here is the pseudo code:

```
Initialize Walabot
Loop
- Measure X,Y,Z
- Translate Y-Z to on-screen location
- - if position is on-screen, show cursor
- - if position close enough to one of the keys,
    highlight the key
- - - if X is below threshold (hand low enough)
- - - - change key graphics to a pressed key
- - - - play a sound
End loop
Close Walabot
```

The sample code and a video showing it in action can be found at [1].

Example #2: Page flipper

Another example of an interface based on hand gestures is flipping pages by swiping your hand in front of the Walabot, to the left or to the right. In this case, only the history of the Y coordinate plays role, to determine whether the hand moves left or right.

You can see it in action at reference [3]. The pseudo code reads something like:

```
Initialize Walabot
Loop
- Measure X,Y,Z
- (process to detect movement and direction of the
  movement)
- if swipe left detected, inject "PgDown" key
- if swipe right detected, inject "PgUp" key
End loop
Close walabot
```

The sample code and a video showing it in action can be found at [2].

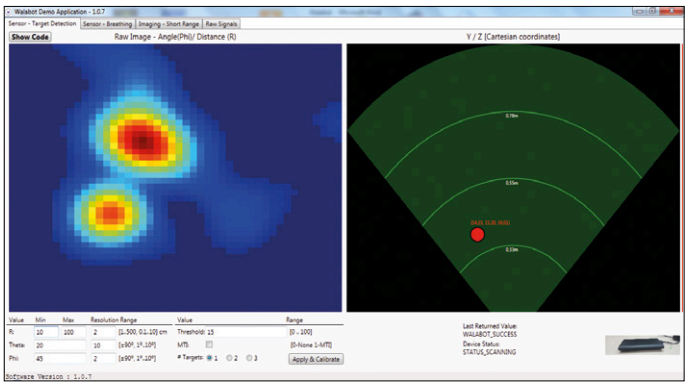


Figure 3. The driver interprets and shows the data.

Whereto next?

The Walabot sensor opens an endless variety of applications in your projects — it can enhance your robot to prevent collisions, it can monitor your sleep and warn you in case of apnea, it can detect a pipe in your wall so that you do not drill into it by chance. You can look at videos showing these uses at [3]. We're sure you'll come with your great ideas. So, how to start? Note that Walabot comes in several flavors, see [4], and the projects described here require the "Maker" or the "Pro" model. Note which regulatory flavor (FCC, CE or other) you need, according to your location.

The API description is at [5]. You will see examples of numerous applications, including application code at the [6].
So, what would **you** do with a Walabot? It's your call! ⏮

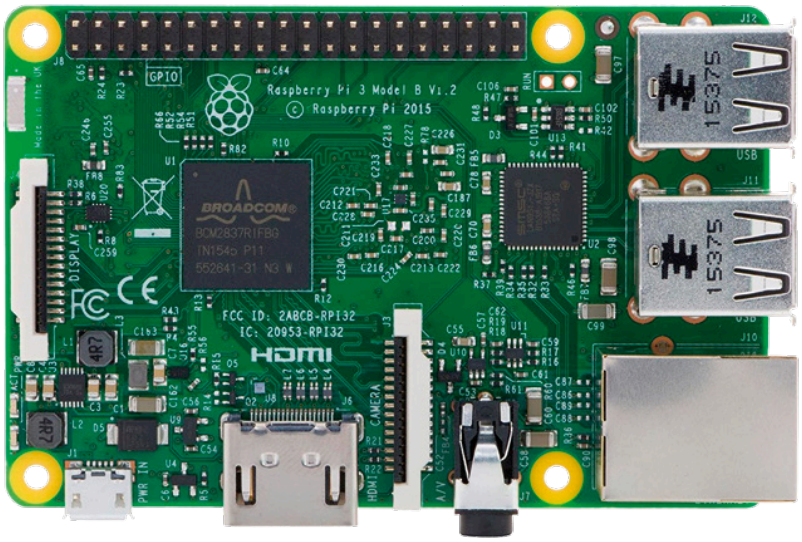
(160033)

Web Links

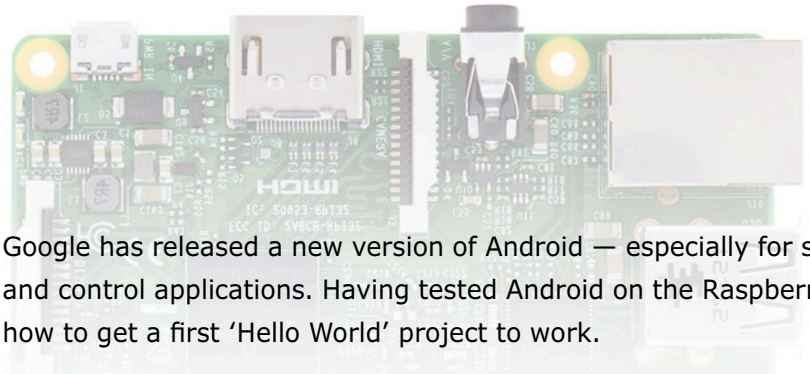
[1] www.walabot.com/applications/projects/piano
[2] www.walabot.com/applications/projects/pageflipper
[3] www.youtube.com/channel/UCULlgO9fDk3jFHdVWKAcDjg
[4] www.walabot.com
[5] http://api.walabot.com
[6] www.walabot.com/applications/projects

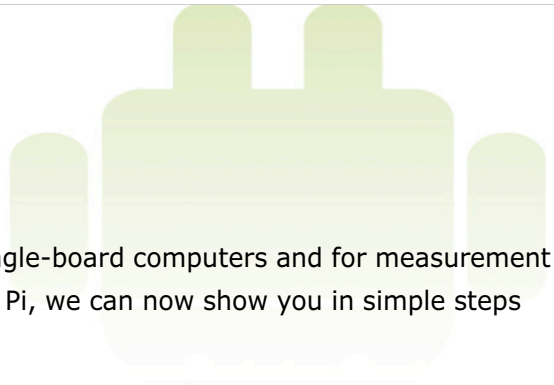
Android on your Rpi (1)

Using GPIO Pins for measurement and control functions









Google has released a new version of Android — especially for single-board computers and for measurement and control applications. Having tested Android on the Raspberry Pi, we can now show you in simple steps how to get a first 'Hello World' project to work.

By **Tam Hanna** (Slovakia)

Microsoft's rapid advance into Raspberry Pi territory has certainly caused Google some headache. After desktop computers, tablets and smartphones, the Internet of Things (IoT) looks like becoming the next battleground.

Pay attention to quality!

If you are ordering memory cards from China make sure you carry out detailed quality tests. Many manufacturers offload batches of inferior quality products that may well report the full complement of memory to your microcontroller. In actual use, however, your data may be lost.

With the Redmond brigade now well advanced along the road towards platform independence, convergence is the new keyword. Google simply cannot afford to ignore the IoT arena, as otherwise Microsoft could exploit this and open a second front on which to attack the smartphone market.

However, Google's *Brillo* platform and language for the Internet of Things announced about a year ago made little impact. Maybe the name was the problem. Whatever the reason, for its second attempt, Google knew it must go direct for the jugular and named its new product simply 'Android Things'.

Simple access to hardware

Let's begin with the obvious. Despite all the kerfuffle — comparisons with Microsoft's 'Windows 10 IOT Core' are purely coincidental — Android Things is by and large a normal version of Android, albeit one that contains some extensions for interacting with hardware. **Figure 1** shows the structure of the operating system.

The *Things Support Library* incorporated with this consists of two modules. First up is the *Peripheral I/O API*, which at the time of writing provides access to the following four types of peripheral (whether OneWire will be retrofitted is currently unclear):

- GPIO with PWM
- I²C
- SPI
- UART

The second component is the *User Driver API*. We're talking here about a program-

ming interface that developers can use to make information on their own proprietary sensors available to the rest of the operating system. Further information on the modules included in the *Things Support Library* can be found at [1].
Let's venture now into some first attempts using practical hardware.

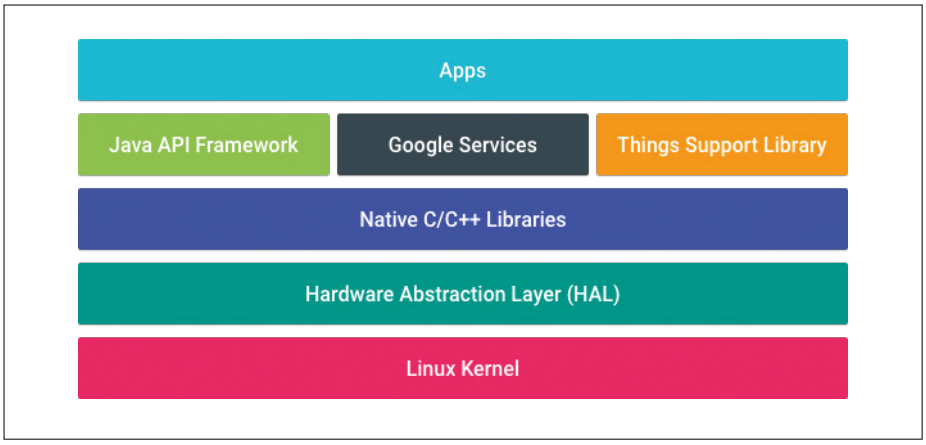


Figure 1. Android Things is a variant of Android (image credit: Google).

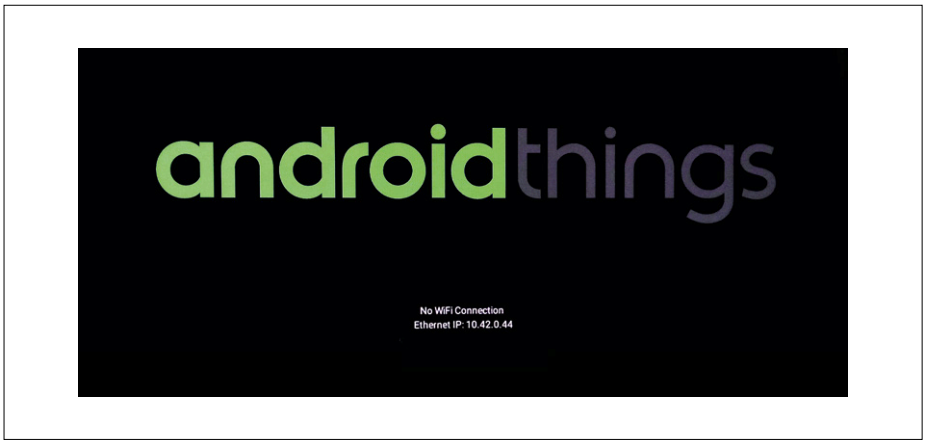


Figure 2. Our own 'Thing' is ready to put into use — remember to make a note of the IP address.

Preconfigured images

At press time Google (doubtless having learnt from its flop with *Brillo*) already offered ready-built Images for the Intel Edison, NXP Pico and Raspberry Pi 3 platforms. In this first article we shall focus on the (by far most widely selling) Raspberry Pi 3.
The first stage involves downloading the Image that awaits you at [2] and then burning it in the normal way onto a memory card (in this connection don't miss the **text panel** *Pay attention to quality*). For the steps that follow the author used an 8 GB capacity card. Using cards of greater capacity should not be a problem; on the other hand the Image size of 4.6 GB means that it will not work with cards any smaller than this.

The next step is to connect the screen and a network cable to your router on the RPi, and start it up by connecting the power supply. Note that a keyboard and mouse are not required, because Android in itself registers only one desktop, without relevant interaction possibilities. We do not want to get involved with debugging over WLAN, because the higher latency of wireless connections can lead to delays while you are tracking down for errors. Nevertheless, if you do want to try this, you will need to connect the RPi to the wireless network following the instructions given in [3].
The first time you start up Android Things on a Raspberry Pi 3 it will take a couple of minutes. At the very beginning, a message appears about Ethernet being missing; you can safely ignore this if the network cable is already connected.
With this work done, the operating system presents the start screen shown in **Figure 2**. The IP address — 10.42.0.44 in this case — will be needed shortly.

Development environment

For space reasons, we have to assume at this point that you already have a working installation of Android Studio on your development PC. If this is not the case, please refer to [4] for help and advice. The author used an AMD eight-kernel workstation with Ubuntu 14.04 to perform the following stages (it's very similar under Windows and Mac OS).
In the next step, change to the root directory of the *Android Debug Bridge* (ADB) on the PC. Then connect the development computer to the RPi by typing the console commands shown in bold as follows:

```
tamhan@TAMHAN14:~/Android/Sdk/
   platform-tools$ ./adb connect
   10.42.0.44

connected to 10.42.0.44:5555

tamhan@TAMHAN14:~/Android/Sdk/
   platform-tools$ ./adb devices

List of devices attached
10.42.0.44:5555 device

tamhan@TAMHAN14:~/Android/Sdk/
   platform-tools$
```

Inputting adb devices is not absolutely necessary here — we are showing you the command because it allows you to identify all devices connected to the Android Debug Bridge.
Also, remember that the connection between the ADB and the RPi can be lost, for example when your PC falls asleep.

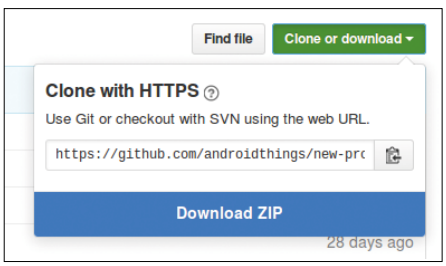


Figure 3. Using GitHub has always been a challenge and remains so.

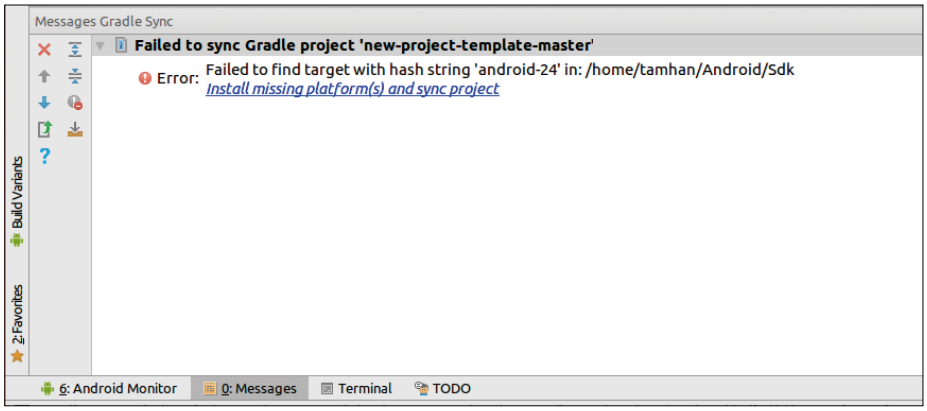


Figure 4. A compatible SDK is missing here.

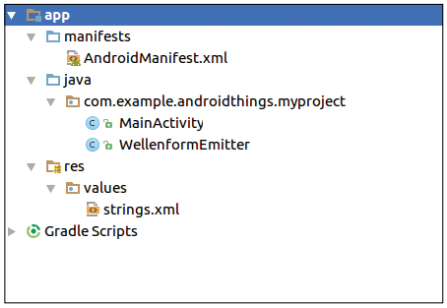


Figure 5. This application gets by without XML markup.



Figure 7. Anyone who blocks the GUI Thread will be punished (image credit: Google).

Android Things is still at an early stage of development and there is no template available yet in the project generator of Android Studio. In its absence you need to call up the existing GitHub Repository [5] and download a sample project skeleton by clicking on the Button shown in **Figure 3**.

If you are already running a project in Android Studio now, close it by clicking *File → Close Project*. The IDE will then display the Welcome dialog. At this point, you must extract the Archive that you downloaded in the previous step and move this into a convenient location in the file system. Then click *Open existing Android Studio Project* and cross over into the directory that contains this project. After you have clicked OK, the IDE will begin to deploy the project. As part of the synchronization of the Gradle project, the IDE will attempt to download any missing components automatically. If any problems arise, you will see an error message along the lines of **Figure 4**; clicking the hyperlink usually resolves the problem.

Do not be surprised if Android Studio complains more than once to report it has found obsolete versions. Android Things requires you to use the absolutely latest version of several components of the operating system, which must be installed in several steps.

After launching the IDE successfully, you should click *Build → Make Project* once more in order to launch a complete compilation. It is important to note that an Internet connection is required when you make the first compilation; afterwards you will also have a realistic chance of being able to work offline with your project skeleton.

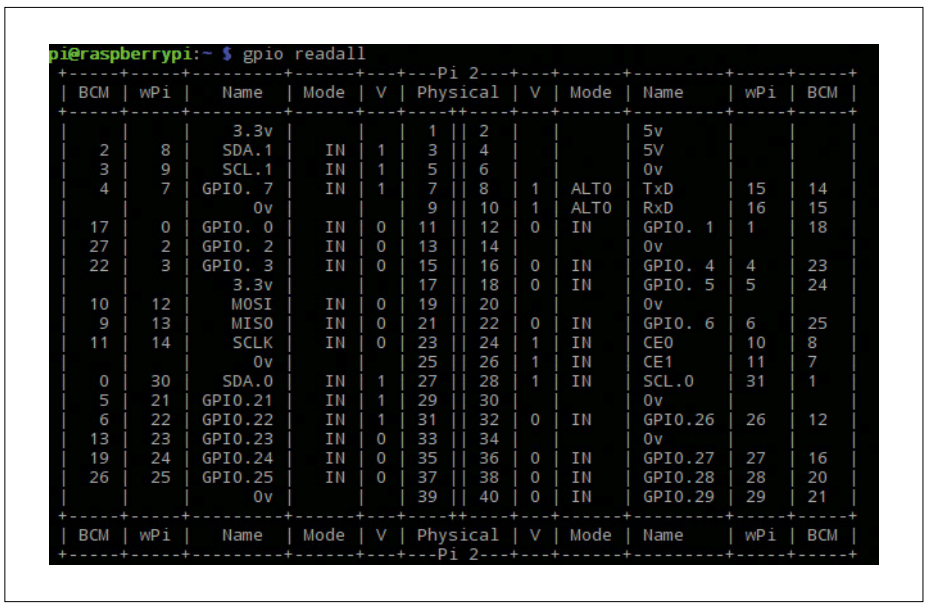


Figure 6. The table indicates the Pins of the Raspberry Pi.

Alles anderDifferent strokes…

The MainActivity code serving as starting point is different from normal Android applications, because Google has added some extra logic calls. These are necessary because Android Things is able to work either with or without a monitor screen. In the latter case, thanks to the Log Calls, you do at least receive information in the debugger console.

Also note that the *res* directory is blank, as shown in **Figure 5**. There is no XML Markup supporting this activity. This affects the code as shown in **Listing 1** — the call to load of resources from XML normally available is missing here.

Another thing to concern us is the construction of the Manifest file, shown in abbreviated form as **Listing 2**. Two modifications are of interest here. Firstly, the Library section ensures that the Libraries mentioned above are inte-

grated into the project. Secondly, MainActivity has an additional *Intent Filter* written into it, which characterizes it as an entry point for Android Things devices.

This is relevant because Android Things has to make do without a program start. RPi is basically a 'one trick pony' that simply processes its host application.

Input and output

For our first demo let's turn to the GPIO Engine. There's an old saying stating that the real-time capability of an operating system decreases linearly (or even logarithmically!) with its complexity. Java-based systems are particularly disagreeable in this respect, with the *Garbage Collector* causing grief from time to time. At this point you must be utterly and completely forewarned that the Raspberry Pi is a 3.3 V platform — connecting 5 V subsystems will result in certain disaster!

With that said we can now turn our attention to the application code in MainActivity (**Listing 3**). First we declare two additional Member Variables. The Peripheral API is implemented in the form of a System Service. A Service is generally available across the entire operating system; applications that need to make use of it obtain a Referral and then interact with it. The Gpio Class is responsible for the actual switching on and off of the Pins.

The Function onCreate (Listing 3) is used to procure the Service. Even if Google is not 100 % orientated towards the Arduino where GPIO Functions are concerned, it's perfectly clear what's going on here. The BCM Strings can be assigned to individual Pins on the basis of the table in **Figure 6**. This table is obtained by entering the command GPIO Readall on a Raspberry Pi (loaded with the standard operating system Raspbian, not Android).

Start the music!

In order to determine the reliability or real-time efficiency of an operating system, it is a good idea to implement a periodically changing output on the GPIO Pin as quickly as possible. If you examine the spectrum on a modulation domain analyzer, you can draw some conclusions. In theory, there should be nothing wrong with implementing periodic on and off-switching of the Pin directly in the OnCreated Function, which is called up at the start of the program. However, if you try this, you will be confronted with the error message shown in **Figure 7**.

The reason for this admittedly rather strange behavior lies in a peculiarity of the Android operating system: the program's user interface is managed in a dedicated Thread called a GUI Thread. If you block this, the operating system punishes you without leniency by 'shooting down' your application. An endless loop would be a classic 'obstruction' that would certainly earn you no mercy.

To avoid this problem we can outsource our Routine in a further Thread.
The simplest way of generating a

Listing 1. Framework of the MainActivity — starting point of the program without user interface.

```
public class MainActivity extends Activity {
    private static final String TAG = MainActivity.class.
   getSimpleName();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Log.d(TAG, "onCreate");
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        Log.d(TAG, "onDestroy");
    }
}
```

Listing 2. Manifest file with Library integration.

```
<manifest . . .>
    <application
      - - ->
        <uses-library android:name="com.google.android.Things"/>

        <activity android:name=".MainActivity">

            . . .

            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.
   IOT_LAUNCHER"/>
                <category android:name="android.intent.category.
   DEFAULT"/>
            </intent-filter>

        </activity>
    </application>
</manifest>
```

Listing 3. The Gpio Class responsible for switching Pins on and off.

```
public class MainActivity extends Activity {

    private static final String TAG = MainActivity.class.
   getSimpleName();
    PeripheralManagerService service;
    Gpio myGPIO;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Log.d(TAG, “onCreate”);

        PeripheralManagerService service = new
   PeripheralManagerService();
        try{
            myGPIO = service.openGpio(“BCM6”);
            myGPIO.setDirection(Gpio.DIRECTION_OUT_INITIALLY_LOW);
            myGPIO.setValue(true);
        }
        catch (Exception e){Log.d(TAG, “Fehler:” + e.getMessage());}
```

Thread is to create a *Runnable.* So your first step is to create a new Class employing the following structure:

```
public class WaveformEmitter
   implements Runnable {
    @Override
    public void run() {

    }
}
```

Listing 4. The endless loop of signal generation is performed in a Thread of its own — a Runnable serves as Container.

```
public class WaveformEmitter implements Runnable {
    Gpio myGpio;
    public WaveformEmitter(Gpio _which){
        myGpio=_which;
    }
    @Override
    public void run() {
        try {
            while(1==1){
                myGpio.setValue(true);
                myGpio.setValue(false);
                myGpio.setValue(true);
                myGpio.setValue(false);
                myGpio.setValue(true);
                myGpio.setValue(false);
            }
        }
        catch (Exception e){}
    }
}
```

Runnables are basically a kind of container in which we ‘pack’ the logic that executes the new Thread. In addition to the run Method described here, you can of course implement various Members in order to provide the information necessary for executing the Thread.
In its fully completed state the resulting Class now looks like **Listing 4**.
In addition to a Constructor responsible for accepting the GPIO Instance, we also have now populated the Method run(), which is what produces the actual waveform output. We generate a characteristic waveform from three rectangles: what’s interesting in this instance is that the duration of the execution of the while loop is shown separately.

The question now arises how we activate the Runnable in OnCreated. A classic mistake made by novices is to invoke run() directly — doing this executes the code in the context of the activating Method (that means in the GUI Thread again). The correct way to activate a second Thread is to create a new Thread Class, with an Object indicated as *Payload* for the Parameter (**Listing 5**). The Thread is initiated using the start()Method.

We are now ready to toss the program in the RPi’s direction.
Thanks to the ADB, which acts as an abstraction layer, it’s sufficient to merely click on Run — the RPI behaves like a phone connected to the PC via USB. Since the MainActivity does not contain a user interface, a plain white screen appears if a monitor is connected, letting us know that our activity is cheerfully getting on with its work.

Debriefing

The next step is to connect your RPi to a modulation domain analyzer, so you can admire the screenshot shown in Figure 8 (the author provides an English language video at [6] giving further information on the function and benefits of using a modulation domain analyzer).
The most conspicuous feature, apart from the occasional jitter, is the formation of two prominent peaks. The region around 2.073 kHz shows the two ‘linear sweeps’ during the transition through the while loop, leading to the somewhat lower frequency of ‘only’ 2.062 kHz.

It's obvious that, in this situation, Android simply cannot match the pace of classic Unix; the interposed Java VM is taking its toll and slowing things down. You can read more on the real-time behavior of Android in various papers on the Internet (for instance in [7]).

The verdict

The program just created is a classic example of overkill in the embedded domain: a shrewd programmer could generate a stable waveform without any real-time operating system, using just a few lines of code and an IC.

Mind you, Bit banging and all that — let's be honest — is definitely not the intended application use of Android Things. On the other hand, the platform will always play its strengths when it comes to realizing demanding applications. In our next issue we'll show you how to interpret data from sensors and display the result in a diagram. Until then I wish everybody good coding! ⏮

(160361)

Listing 5. Start of our signal generation Thread in MainActivity.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    . . .

    WaveformEmitter myEmitter=new WaveFormEmitter(myGPIO);
    new Thread( myEmitter ).start();
}
```

Web Links

[1] https://developer.android.com/things/sdk/index.html

[2] https://developer.android.com/things/preview/download.html

[3] https://developer.android.com/things/hardware/raspberrypi.html

[4] https://developer.android.com/studio/install.html

[5] https://github.com/androidthings/new-project-template

[6] www.youtube.com/watch?v=lBLEfVUVGyU

[7] www.utdallas.edu/~cxl137330/courses/fall14/RTS/papers/4a.pdf

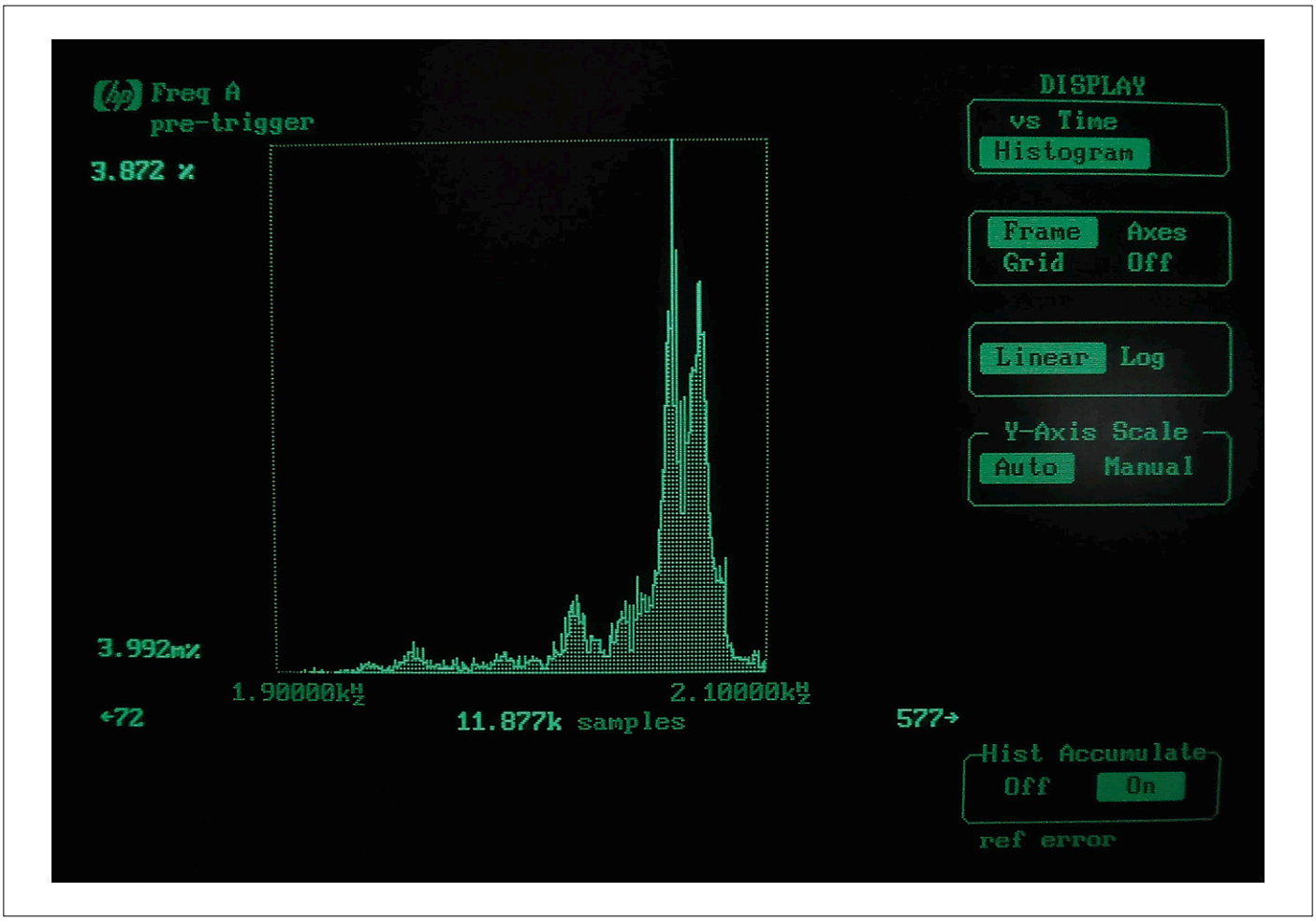


Figure 8. Arduinos generate significantly 'prettier' histograms in this application.

RETRONICS

Made in Belgium

Bilingually, naturally



Over the past few months some friendly advice ("*gene goesting, Jan?*") came in from Belgian readers saying their national products are not receiving enough coverage in the Retronics section. Amends are made here with descriptions of a few DIY kits from the erstwhile MBLE company, aimed at affluent hobbyists in the early 1970s.

By **Jan Buiting**

Hands up all of you having started out electronics with a Philips 'EE' (Electronic Engineer) kit stored under your bed and used on the kitchen table! Fond memories of the fantastic projects you could build from discrete parts (like sirens and electronic canaries) and equally the 'extension' kits you dreamt of and put on your Xmas wishlist preferably filed to dad. These EE kits definitely had the "addiction factor" as they were horribly expensive not only shiny new from the shop but also as far as "replacement components" were concerned (like in my case 1 pc. AC187K = 1 week's allowance) so you'd tread cautiously.

And mum was happy the howling and blinking stuff you'd build could be made to disappear magically into the cardboard box again after the proud demo.

Dutch, German, …

The full history of the Philips EE series of electronics educational kits is a complex one and beyond the scope of this article. The complexity is mainly owing to Philips Holland being a company constantly in turmoil shifting around whole factories, departments, and thousands of people.

The story is rendered in a nutshell here mainly to put the MBLE kits I want to discuss into the right perspective.

Elonco, Philips' formidable electronic components division, started to serve Dutch hobbyists with radio and audio construction kits in 1955, the engineering and design being contracted from the 'Amea' engineering bureau in The Hague. The business flourished with many renowned kits like the 'Pionier' radio, the HF302 amplifier and the FM13 tuner released in the late 50's. A little later the first multi-project educational kits appeared like the EE8 and EE20. Around 1965 the EE kit creative team was dropped into Philips RGT (radio, gramophone, television) and the range was extended with mechanical engineering (ME) [1] (**Figure 1**). Part of the EE kit development team worked on the (splendid) documentation, others on the selection of components (they had to buy in-house), and still others on the didactic aspects, graphics and marketing. You may remember at this point the "father-and-son" and "ideal-son" images printed on the early box covers (**Figure 2**).

By around 1970s the dreaded Scrooges at Philips Holland advised that the EE multi-project boxes were "not a core activity" and first had the EE creative team put on side track. Next the operation (i.e. not the staff) got moved to Hamburg, Germany. A new team there got cracking with *Gründlichkeit* and perfected and extended the EE educational kits with tremendous success within a bristling market (with many excellent competitors and good turnovers for all). They developed chemistry (CE), physics and mineralogy kits also.

Meanwhile Philips Holland continued to develop DIY kits covering mainly audio and loudspeaker projects. The activity was formally stopped in 1980 but the educational kits developed in Hamburg were sort-of localized to the Dutch market and language by Philips TSCA (technische service ? afdeling) — consequently the build instructions in these post-1980's EE kits had lots of droll wordings due to German language interference.

The wider perspective

Behind the guise of teaching electronics was Philips' ulterior motive to "sow the seed for brand awareness" at a young age. Yes, brainwashing boys to buy Philips parts and gear later … when they graduate as Electronics Engineers preferably at Eindhoven Technical University! Philips' EE kits differed from, say, the US approach by top-dog electronic kit maker Heathkit in at least two ways:

1. Heathkit had no "aftermarket", and used parts from a wild variety of manufacturers (sometimes rebranding parts 'Heathkit' though);
2. using the same parts over and over again, the EE kits were intended to build various small projects, not complete instruments.

Still, I am sure an unassembled Heathkit kit was on a Philips EE creative team member's desk sometime somewhere and he (unlikely, she) was envious at (a) Heathkit's rigor and sheer quality of the construction guidance and (b) the final result, i.e. a fully working instrument or electronic apparatus like a tubed Hi-Fi amplifier [2], an oscilloscope, or a ham radio transmitter — all at competitive prices (in the US). Until well into the 1980s Heathkit would set the standards not only for DIY equipment kits proper but also the fantastically detailed construction guides and after-sales service.

Cracks appear

Despite their high price the Philips EE kits were popular from the start and succeeded in fulfilling their missions: to amuse, impress and educate. With hindsight the brainwashing was not successful on part of Philips I'm afraid as electronics as a "pastime with career potential" took off big time in the late 1970s and early 80s, and with it the number of component suppliers in the market. And AC187K's sold by the pound.

While at full steam ahead with the EE series multi-project educational kits, Philips recognized a market for the 'Heathkit' approach too. i.e. affordable **instruments** built from a DYI kit (by a young man). Call it a no-brainer but Philips in Holland had two trump cards staring in their faces:

1. being able to draw from their own organization's massive stock of parts;
2. being allowed to copy-cat and then

ESTD 2004
www.elektor.tv

Retronics is a regular section covering vintage electronics including legendary Elektor designs.
Contributions, suggestions and requests are welcome; please telegraph editor@elektor.com



Figure 1. Philips 'ME' mechanical engineering kits, *ca.* 1965.



Figure 2. Those Philips Electronic Engineer (EE) kit covers were iconic, always emphasizing the career aspect of the bright and well-behaved young electronician.

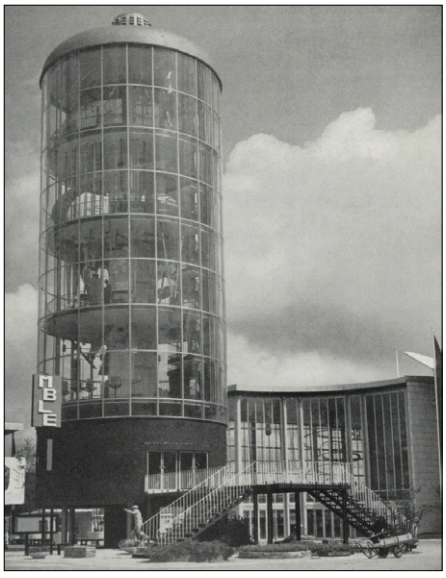


Figure 3. The MBLE Pavilion the World Expo in Brussels, 1958 (the same year when Philips Holland staged the world's first multimedia show, the *Poème Electronique* in their own pavilion designed by Le Corbusier).



Figure 4. The BEM004 signal generator. Note that this is not a commercial instrument from MBLE but built from a kit of parts.

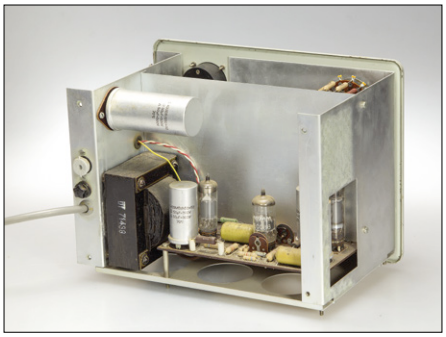


Figure 5. Inside the BEM004: conventional construction techniques using vacuum tubes, and a chassis.

down-spec their own 'pro' level lab instruments (GM series) and exquisite Hi-Fi audio equipment (AG/22AH series). We'll see an example further on.

And so, a glut of 'kits-of-parts' got developed, of which I always found the Audio Mixer the most impressive. The range was massive and immediately recognizable from their blue-and-white cardboard boxes which ruled the roost till the mid-1990s. The kit prices continued to be horrendous compared with what could be built from hobby magazines and "equivalent" parts from your own drawers or dad's. The quality though was superb. Fine overviews of the Philips educational kitting activities may be found at [3] — warning: it's all Dutch there but no double Dutch! And our German readers will delight at the e-nostalgia at [4].

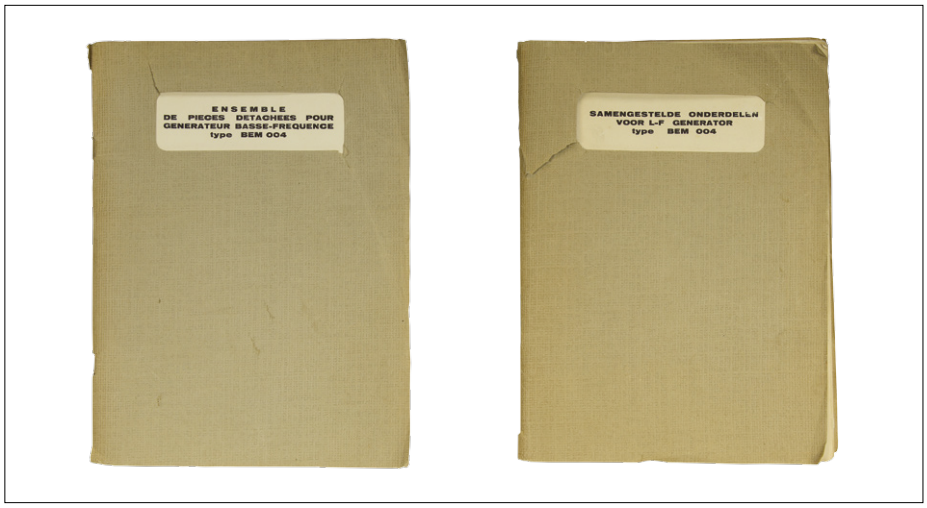


Figure 6. A bilingual ergonomical miracle! The BEM004 Assembly Manual contains Flemish and French — all you have to do is flip the book.



Figure 7. The BEM005 oscilloscope. Note again that this instrument was built from a kit of parts.

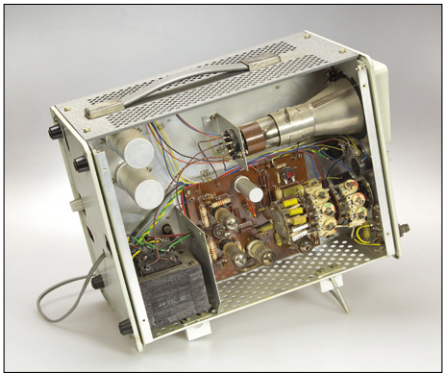


Figure 8. Inside, the BEM005 has plenty of space. Note the use of "printed wiring boards" (PWBs) in combination with vacuum tubes, which as we know is felicitous only if quality tube sockets are used in combination with proper soldering and solid copper traces.

Manufacture Belge de Lampe et de Materiel Electronique

After the move to RGT Philips' lab gear kits were forgotten initially but then resuscitated and assigned as a minor operation to MBLE in Belgium, headquartered at 80 Rue des Deux-Gares, Bruxelles-7. MBLE had been adsorbed by the mighty Elonco ('electronische onderdelen') subdivision of Philips, and was assigned to manufacture special resistors (VDRs, PTCs), 'mustard' capacitors, electrolytics and possibly tubes also (Ultron), in their plants at Roeselare, Evere and other locations. MBLE was a medium-size company with a fantastic history you can read up at [5]. I found a picture of their 'pavilion' at the 1958 World Expo in Brussels (**Figure 3**) and it looks like the largest PL500 tube ever built. Hopefully it's not on vacuum!

Literally sitting on electronic parts, MBLE under Philips Holland governance developed a range of fine kits to build home-lab equipment mostly. Two Belgian readers of *Retronics*, both Flemish-speaking and former teachers of electronics (!) proudly sent me a few specimens of lab gear assembled from MBLE kits aimed at the advanced hobbyist. Here they come.

BEM004 Low-Frequency Generator Kit

'Low-frequency' may be a bit of misnomer, or just modesty, for this tubed signal generator since it covers 10 Hz right up to 1 MHz in five ranges. The electronic design is not spectacular around the usual complement of Philips Miniwatt tubes like EF86, ECC88, EL83, and EL85 not forgetting four small bulbs in the oscillator. Basically, this is an economy version of one of Philips' professional lab grade instruments from the GM56xx series. **Figures 4 and 5** show the pristine and perfectly built BEM004 I was given. By today's standards the plastic knobs may have a cheap look and feel

but back in the day they were top notch and *avant garde*. You may contend with the use of a SO-239 ('UHF') socket for the output signal considering the wording AUDIO GENERATOR on the front panel.

The BEM004 manual (**Figure 6**) is an educational masterpiece with extremely detailed, step-by-step instructions to assemble the instrument, as well as huge fold-out diagrams and exploded views of the chassis. It's also very Belgian and even Brusselian in being bilingual — you just turn the book to read it in either French or Flemish (a pure variety of Dutch). I can read both languages and confidently state that the French version is the original written by the MBLE engineers, and the Flemish one is 'a' translation with some hilarious instances of literal translations and word order mess-ups — but not worse than Google Translate today.

This BEM004 works, though a 50+50 µF 400 V dual electrolytic can capacitor liter-

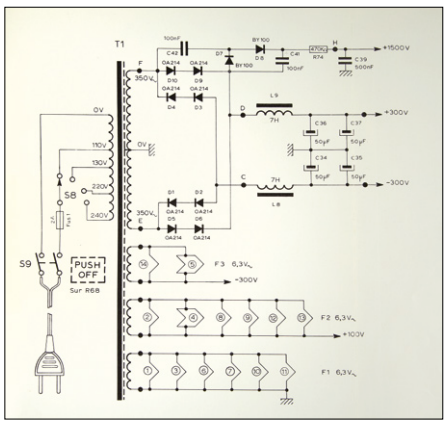


Figure 9. With such a simple unregulated power supply in an oscilloscope you run the risk of image distortion, hum, noise, and general instability of the CRT trace. (reproduced from BEM003 manual)

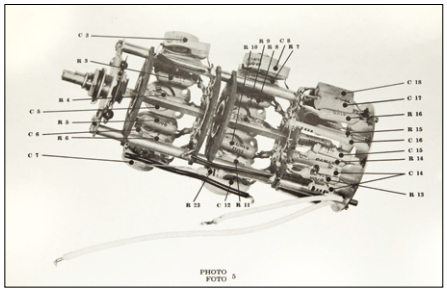


Figure 10. To assist with complex assemblies, retouch type photographs printed on separate sheets were included with the assembly manuals.



Delayed Trigger

Since its inception in 2004 the articles published in Elektor's *Retronics* section have drawn substantial response from readers and not limited to writing. Below are a few instances of donations and other contributions received over the years, along with the original article that was the 'trigger source' for readers to respond. For reasons of privacy, the names of the contributors are withheld unless specifically permitted to print here.



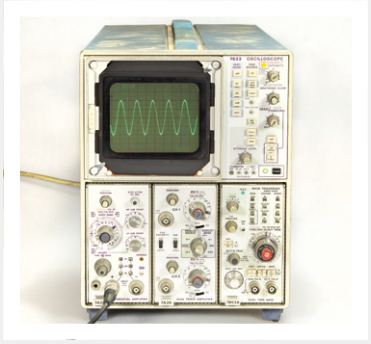
Object 01: General Radio 1608-A LCR Bridge. This rare apparatus was donated by a reader after a chat with a Dutch-to-English translator of Elektor Netherlands Magazine at a small trade show in Utrecht in 2016. The 1608-A is the 0.1% accuracy, non-portable, way-more-costly version of the GenRad 1650-A LCR Meter I covered back in Elektor June 2007 and on video: https://youtu.be/YfrW79pBbs4. Compared to the 1650-A the 1608-A is a boatanchor :-). Status: to be restored and calibrated.

Trigger 01: Type 1650-A impedance Bridge (1960), Elektor June 2007.



Object 02: Elektor SC/MP Computer. I misjudged the popularity and impact of the SC/MP computer (pronounced: /skamp/) mainly because the one left on the Elektor premises was not modular. Reader Friedrich Bollow sent a real SC/MP with 3 pcs 4 K RAM, serial I/O, Elbug loaded … The Works, professionally built with all boards mounted in a rack enclosure. The stuffed CPU and RAM boards together cost around DM 1000 back in 1978. Friedrich is fondly holding on to his 45-rpm 'ESS' vinyl discs with the system software and other utilities. Status: to be powered up.

Trigger 02: Elektor SC/MP Computer (1978), Elektor April 2005.



Object 03: Tektronix Type 7633 100-MHz storage oscilloscope. Tektronix seems to be the darling manufacturer of CRT'ed oscilloscopes, witness the dozen or so instruments I was given since my tale on the 564B. This 7633 'scope was happily traded against a few beers, a book and some microcontroller boards, at Elektor Live in Hanau back in 2013. After cleaning one switch and replacing two capacitors the 7633 worked again and is now a resident on the Classic Repair Desk. Status: calibrated.

Trigger 03: Tektronix 564B Storage Oscilloscope (1963), Elektor May 2011.

Feel free to post-trigger and be listed here:
Email **editor@elektor.com**; Subject: **trigger [nn]**.



Figure 11. The BEM003 was BMLE's top line oscilloscope, sadly this one is let down by a few defective parts. It's up for repair though.

ally cried out for restuffing with modern replacements (4 pcs 100-µF 250 V). The highest sinewave distortion measured was 0.2% at 100 kHz, which is well within spec (0.3%). The same for the 75 ns risetime of the square wave (the fall time is poor though).

BEM003 and BEM005 Oscilloscope Kits

By the end of the 1960s if you were young with an interest in electronics, to own an oscilloscope was heaven as it was the instrument whose bright trace "revealed all mysteries". Sadly these things were dangerous and daunting to build from 'plans', and the ones you saw advertised were well out of financial reach. One option that remained, at least in theory, was save & scramble the money to buy a DIY kit from a reputable manufacturer, in this case MBLE. By 1975 the available 'scope kits were still tube based in essence, including:

- BEM003, 0–7 MHz, V: 20 mV – 50 V/div.; H: 0.2 µs – 20 ms; price HFL 945.
- BEM005, 0–3.5 MHz; V: 50 mV – 50 V/div.; H: 0.5 µs – 20 ms; price HFL 645.
- BEM009, 0–0.7 MHz; V: x1/x10/100 attenuator only; H: 10 Hz – 100 kHz; price HFL 430.

Believe-you-me, the HFL (Dutch guilder) prices even of the BEM009 were very steep and I imagine these kits reached the professional repairman rather than any student or hobbyist.

I was given a BEM003 and a 005. The 005 (**Figures 7** and **8**) I was able to get back to life and then full working order after slowly waking it up on a variac. But plugged into a power socket the next day at another location it failed completely, not even the power light would come on. A brief examination revealed a heavily corroded AC line fuse. The little culprit was cleaned, the fuseholder as well, and the 'scope then powered on okay again. The vertical sensitivity was way off and the relevant rotary switch required a few squirts of Tuner 600 deoxidizer to clean the contacts on two decks.

To be honest this 'scope is a mediocre performer but I may be heavily biased and spoiled by my Tektronix gear like the 502. The trace is not ultimately sharp and seems to shift, shrink and blow up somewhat when you operate the intensity, focus, and even the H and V controls. It's a sign of poor supply voltage regulation throughout the circuitry and indeed there is none, not even an 85-V or 150-V stabilizer tube — see the PSU schematic reproduced in **Figure 9**. The lack of regulation badly affects the +1.5 kV HT on the CRT anode, hence the geometry issues. This is where the design of the BEM005 and 003 differ from, say, a professional Philips GM56xx 'scope from (essentially) the same company with (essentially) the same parts. But remember, in those days tubes and transformers were among the most expensive parts in any electronic instrument and if economies could be made, these parts fell victim first — except at Tektronix.

Still presented as state of the art, the BEM003 and 005 kits used printed circuit boards. These are the horrid brown pertinax boards from Philips. Fortunately the tube sockets are ceramic and you did the soldering yourself.

The BEM005 manual is again bilingual (i.e. French/Flemish) but not of the flip-it-around type as with the BEM004 sig gen. The mechanical construction and wiring diagrams are enormous on 65 x 45 cm foldout sheets, which easily doubled as posters to stick on bedroom or classroom walls. Tech photos like in **Figure 10** of really complex assemblies help to achieve the ultimate goal: a working oscilloscope! To all this paperwork the previous owner found it wise to add copies of the datasheets of all tubes used.

The larger and better equipped BEM003 (**Figure 11**) still has a few issues for me



Figure 12. With a Belgian pinch of salt MBLE's BEM045 multi-instrument can be called an electronics lab in a box. Note though that this was never a kit for home assembly but an important adjunct in Philips' PRACTRONICS course for audio and acoustic engineering.

to grapple with, like oxidized switch contacts and leaky capacitors in the time-base causing the green trace to disappear or shorten on some ranges. I have put the 003 aside for the moment to function as a doorstop.

BEM045, student lab-in-box, not a kit

Pictured in **Figure 12**, the BEM045 is a hinged, dull grey plastic box you can open at the handle side to reveal in the left-hand part:

- a 20–20,000 Hz, 0–1 V out, sine/square wave generator;
- 2 mA and 20 mA current sources;
- an adjustable 0–12 V power supply;
- a 6-0-6 V (quasi) symmetrical supply.

The left part also contains the battery compartment. In the right-hand part we find a simple multimeter comprising:

- a 0.1–300 mA DC ammeter;
- a 10 mV / 30 mV / 0.1–30 V AC voltmeter;
- a 0.1–30 V DC voltmeter;
- a large moving coil meter with volt and dB scales.

From which we can easily conclude that the BEM045 was aimed at audio electronics and acoustics study & research, once an honest and momentous craft. Originally supplied by Philips TSCA for a course called PRACTRONICS this very portable unit must have caught the attention of the MBLE marketing people due to its 'student potential'. And indeed the manual shows ELA EDUC PRODUCTS printed on the front page, where ELA is likely to mean electroacoustic. A Dutch-language forum thread with contributions from Belgium and Holland [4] suggests a different chain of events though, like MBLE having manufactured the BEM045 set especially for the ELA division. This is corroborated by the fact that the BEM045 was never supplied as a DIY kit or featured in any MBLE kit brochure. Also "the lab in a lunchbox" appears occasionally on Dutch markets, online and real.

Although the exact origins of the BEM045 remain unknown, for sure it's a remarkable unit, if only for the "educational projects" built from the "lessons" in the "book" being constructed on top of… the book page! See **Figure 13**. *What do you mean*, web and apps and VR for e-education?

My BEM045 has two large numbers on the case, like '78' scrawled with a yellow crayon, and '39.' (*sic*) burnmarked with a hot solder iron — both suggest classroom use.

The lab-in-a-box did not suffer from corrosion due to leaking batteries (6 pcs 1.5 V C size dry cells). The previous owner managed to install a round BNC socket (yes) in the small side panel of the left part to allow an external 12-VDC bench supply to power the instrument via a cable which I was fortunate to get as part of the donation.

All components of this mini lab turned out to work as expected, except the signal generator which takes quite a while to wake up and stabilize its amplitude when the frequency is changed on the round dial. Clearly the stabilization loop has a defective component somewhere — my money is on an electrolytic capacitor.

I was amazed to see the V/A meter section work apparently without a power connection (the 12 V supply is fed to the left panel with the sig gen). After some careful dismantling and inspection the secret was revealed: the 12-V supply wires are conveyed to the right panel by passing them through the two hinges. This construction reportedly is the Achilles' heel of the BEM045.

Is it a nice instrument to work with? Hardly. I do not have a manual. The feel is extremely 'plastic' and the multimeter lacks resistance measurement (sure, $R=V/I$ but still). Plus I am forever afraid I am destroying something with amps. The moving coil meter is large but lacking in scale detail, and the range indication is less than intuitive. There is no frequency sweep option to quickly analyze filters. On a positive note, the sinewave once stabilized is fine in terms of distortion, and the frequency control is sufficiently accurate. Still, the combination of instruments in this XL lunch box will only suffice for audio electronics design just above Grade 0.

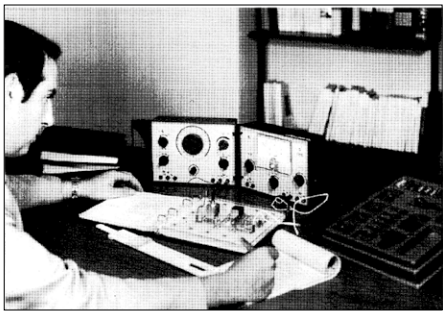


Figure 13. The Philips PRACtical elecTRONICS course, ca. 1975. Using the signal generator and multimeter in the BEM045 as tools, the student would build and understand audio circuitry. The circuit of the day is built literally onto the page of the courseware book! With our apologies for the inferior image quality, this is rare material.

Will it blend?

One well-meant question I heard from a few non-technicals was: "*so did you manage to repair them scopes with that giant lunchbox?*" I did not. The 045 is not suitable for anything beyond 30 V, and its sinewave is distorted or unstable initially as well as poorly defined in amplitude, creating a multi-variable conundrum together with, say, vertical sensitivity issues inside the scope … and low AC line power like 212 V (??) on the one power outlet in the Retronics attic … Sorry!

In short: I had to rely on a known-good Tektronix 7633 'scope, a solid and accurately reading MBLE (!) variac, and a Delta benchtop power supply to reach the minimum levels required to get a trace on the CRTs. Achieving that, the cheers from colleagues descended for a minute from Embedded Heaven, and the reporting on a piece of e-history from Belgium is great though and I say *Allez Mannekes*. ⏮

(160303)

Web Links

[1] Philips ME construction Kits, www.elektormagazine.com/070277

[2] Heathkit AA-100 Tube Amplifier (1960): www.elektormagazine.com/140333

[3] www.hansotten.com/electronic-kits/ee-series/ee8-ee20-a20

[4] Norbert Old: http://norbert.old.no/kits/ee2000/index.html

[5] MBLE company: www.radiocollection.be/fr/MBLE1_fr.html

[6] MBLE045: www.philipsradios.nl/forum/index.php?mode=thread&id=23681

Electric Motors Q&A

A first acquaintance

All electric motors turn and can therefore be used to move something. However, each type of motor has its own characteristics, so it's helpful to choose a motor type and drive arrangement that match the sort of motion you need.

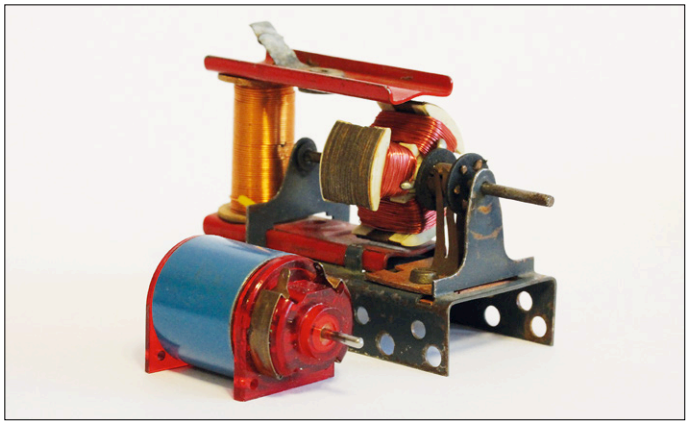


Figure 1. "My first motor" (or two): starting with the basics. The one with the blue housing is a DC motor; the one on the right can run on DC or AC, so it can be connected directly to a model train transformer.

By **Ties Wijffels** (Netherlands)

Q *Let's start with the supply voltage. Should it be DC or AC?*

A The motive force behind every electric motor (**Figure 1**) is magnetism: a magnetic field drives another magnetic field. In order to produce motion, one of these magnetic fields must be constantly changing. This is always achieved by using an electromagnet. The static field can be produced by a permanent magnet, a second electromagnet, or a short-circuited winding on the rotor.
In a DC motor the rotor windings are switched by a commutator, which is a sort of changeover switch mounted on the rotor. However, that can also be done by electronic circuitry. With stepper motors the windings are also switched electronically, but with external circuitry, and the brushless motors commonly used in model aircraft have electronic commutation. AC motors usually work with a rotating magnetic field produced either by the AC line voltage at 50 or 60 Hz, or by a voltage generated by a frequency converter. However, AC motors can also be equipped with a commutator and carbon brushes.

Q *What's the best way to keep it simple?*

A The most common type of motor is the DC motor (**Figure 2**). Most relatively small DC motors have a permanent-magnet stator and a rotor made from soft iron with windings which are supplied with current through carbon brushes and a commutator.
A nice feature of DC motors is that the speed is nearly proportional to the applied supply voltage.
Although DC motors are specified for a particular speed at a particular supply voltage, that is not a hard specification. You can easily use a higher voltage for a higher speed or a lower voltage for a lower speed.

Q *How do I connect a DC motor to my circuit?*

A The control voltage for a DC motor is usually a PWM signal, which means that the voltage is applied to the motor in the form of a series of pulses and the mechanical inertia of the motor converts them into a steady speed.
If the PWM frequency is too high, you can run into problems with the inductance of the motor. For that reason you should choose either a low PWM frequency (at most a few hundred Hertz) or a very high PWM frequency (20 kHz or higher).
The switching transistor should be selected according to the desired switching frequency and the expected motor current. Small motors operating at low PWM frequencies can be driven by a simple transistor or power transistor. Darlington transistors are a good choice for higher motor currents.
Power MOSFETs are almost always used for real high-power applications. The very low on-state resistance and high drain current capacity of power MOSFETs (more than 100 A is by no means unusual) result in much less power dissipation with power MOSFETs than with bipolar transistors.
It's important to always use a freewheeling diode. When the current is switched off, the rotor inductance produces a voltage spike that must be diverted to the positive supply voltage rail.

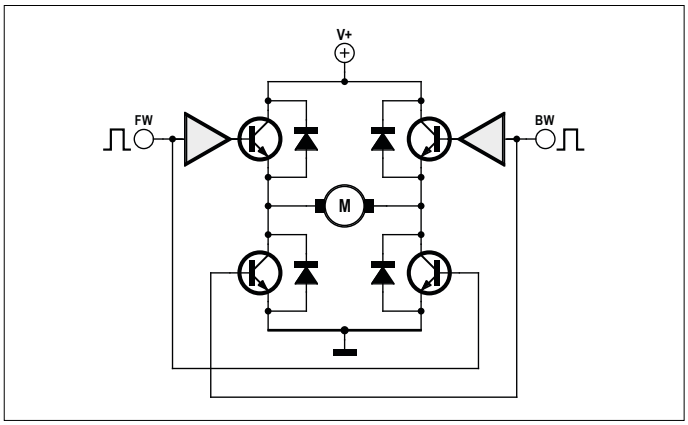


Figure 3. With an H-bridge you can run a DC motor in either direction.

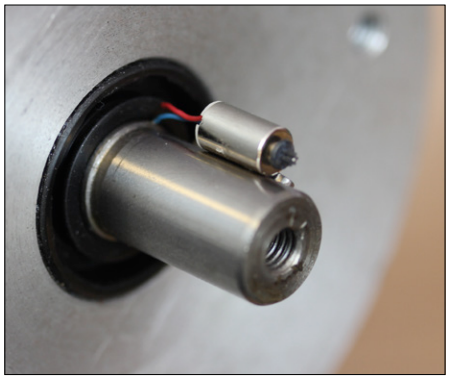


Figure 2. DC motors: big and small.

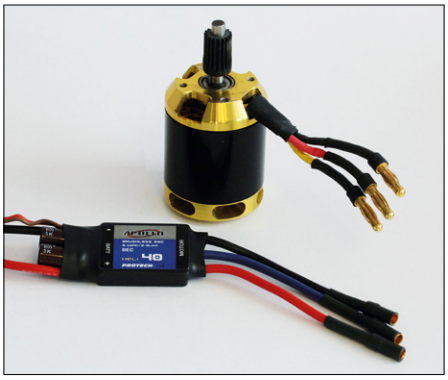


Figure 4. A brushless DC motor and motor controller.



Figure 5. Encoders: you can also make your own.

Q *How can I control the direction of rotation of the motor (clockwise or counterclockwise)?*

A If you want to have a DC motor rotate clockwise instead of counterclockwise, you have to change the polarity of the supply voltage. That can be done with a changeover switch if the direction only has to be changed occasionally. If the direction has to be changed frequently, for example in a servo system, you need an H-bridge output stage with four transistors in an H configuration, where only two diagonally opposite transistors are conducting at any given time (**Figure 3**). Freewheeling diodes are also necessary with H-bridge circuits (four in that case).

Q *What if I want to use a brushless motor to avoid wear?*

A Then you can use the type of motor often used in model building, which has three connecting leads (**Figure 4**). These motors are actually equivalent to brushless AC motors, but with the rotating stator field generated externally. If you want to use one of these brushless motors, it's a good idea to buy a ready-made controller. The controller is driven by a pulse signal with a range of 1 ms (stopped) to 2 ms (full speed). The frequency of the control pulses can vary, but 50 Hz is almost always a good choice. The direction of rotation can be changed by swapping two of the three phases. If you want to be able constantly change direction, you can use a controller with a zero midpoint (1 ms = reverse, 1.5 ms = stop, 2 ms = forward).

Q *Okay, now we have motion, but what if I want to position something precisely?*

A Positioning can be done in different ways. You can use a normal motor (DC or AC) with position feedback from an encoder or a potentiometer. Potentiometers are used in servos, such as the servos commonly used in model building. It's not surprising that the popular Arduino family has a separate servo library. However, for positioning tasks the usual choice is a stepper motor.

Q *Can stepper motors be operated with a different supply voltage, the same as DC motors?*

A Although stepper motors operate on the same basic principle as DC motors, the situation is more complicated due to the different application area. A DC motor runs when a supply voltage is applied and stands still when no voltage is applied. A stepper motor stands still most of the time, but it needs a supply voltage to maintain its position. The stator resistance of a stepper motor is relatively high to limit the current when the motor is standing still. That is different from DC motors, where the rotor resistance is kept as low as possible for efficiency reasons. If you raise the supply voltage of a stepper motor, heat dissipation may become a problem, especially at standstill. For that reason you should stick to the specified supply voltage.

Q *Do stepper motors also have drawbacks?*

A The most commonly encountered problem with stepper motors is losing steps when the stepping rate is too high or the load is too heavy. When that occurs, the positioning is no longer accurate. Another frequent issue is resonance at low stepping rates, which causes audible humming. What's worse is that resonance can destroy drive trains. For that reason speed reduction with gears, which always have a certain amount of play, should not be used. Toothed belt drive is a better solution.

Q *Is there an alternative to stepper motors?*

A The nice thing about stepper motors is that you do not need feedback for positioning. If you opt for explicit feedback, for example with an encoder on the motor shaft, then you can basically use any type of motor for positioning (**Figure 5**).

Q *Aren't encoders awfully expensive?*

A They don't have to be. If you do not need very high precision, it is certainly possible to make your own encoder using a light-barrier sensor and a simple encoder wheel, or a vane or spot on the rotating shaft. ⏮

(160219)

Web Links

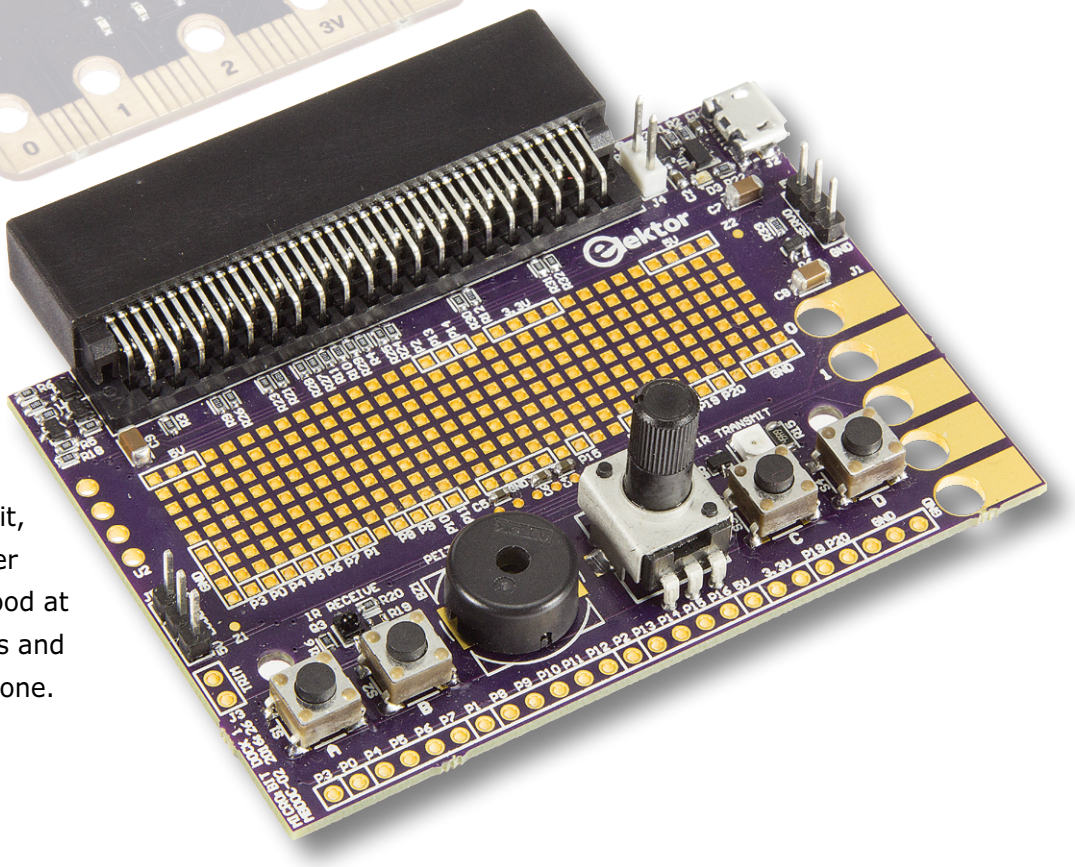
https://en.wikipedia.org/wiki/DC_motor

https://en.wikipedia.org/wiki/AC_motor

https://en.wikipedia.org/wiki/Stepper_motor

Dock for BBC micro:bit

"Base to micro:bit, you are cleared to dock."



By **Jason Long** (Canada)

There should be at least a million BBC micro:bits floating around the UK and other parts of the world right now. Though there are indeed many things you can do with a micro:bit, you have a somewhat limited user interface unless you get really good at writing Bluetooth-connected apps and take advantage of your smart phone.



We know that the young (and the old) hackers out there don't want to be tethered to a phone all the time, and no doubt there are endless projects you can do that will require your micro:bit to stand alone.
So we set out to design a docking board (and maybe a series of docking boards) to bring many additional features and functionality to the base micro:bit system. The goal of the first dock was to provide some more I/O, prototype space, and an easy way to break out the full micro:bit pin array to a bread board.

That edge connector...

The first challenge to solve was the edge connector. While it looks kind of cool and is essentially free to put the contacts on the board, it makes plugging the micro:bit into anything practically impossible. Edge connector receptacles are not exactly cheap and their pin out is not breadboard compatible. Initial attempts to locate something cost effective almost ground this project to a halt before it even started. It is funny how simple problems can have such a significant impact! Welcome to engineering. Fortunately we were able to finally source a low cost edge connector and we all celebrated as we could build the dock we really wanted to bring you!

We've done a lot of development boards and have our favorite features. The first iteration of the dock board had ALL of these plus the standard Engenuics daughter board connection so you could attach infinite other boards. We made some assumptions about how the micro:bit would work because at that time it wasn't easy to get our hands on one. This board also used our original two-stage connector idea as we hadn't found an edge connector yet (**Figure 1**).

We did a small-turn prototype run and played around with this for a little while. The BOM cost was high on some parts and the board seemed too big. In the end, we decided we'd go back to the drawing board and focus on keeping everything as low cost as possible while still delivering some solid features.

The second iteration dropped some high cost components and added the newly found edge connector. There were a few pin changes in the final released micro:bit so we made those adjustments, too. We decided to make the LCD optional and positioned it over the prototype space to minimize PCB size (**Figure 2**). We don't really know what will happen with micro:bit in the future, so all of the pins we used for dock hardware can be disconnected by removing a jumper resistor. It would be a shame to have strange behavior on a great project because of some unanticipated interactions with the dock.
For production we made some minor tweaks to the layout and silkscreen but generally we were ready to pull the trigger and release the design. Let's take a walk through what we've come up with for you! (**Figure 3**)

More buttons

Of course the easiest way to get information into an embedded system is through buttons and it never seems you have enough! We debated about how to add them and eventually decided to duplicate the two existing buttons (A & B) and add two more (C & D). This puts all four button inputs at your fingertips when holding the dock and integrates easily into the base micro:bit code.

Buzzzzzzzzzz and servo control

While LEDs are number one when it comes to easy, low cost digital output indicators, audio is arguably the next best. With a simple PWM peripheral or bit-bashing task, you can drive people (and dogs) crazy with irritating tones across the audible frequency spectrum. Even though a piezoelectric buzzer does not produce the most beautiful audio (let's be honest, it's horrible), it does add a fantastic new dimension to your projects. Sorry to all those around you, especially to teachers who have a classroom full of micro:bits.

A standard servo control header is included and the SERVO line is level converted to 5 V from the 3.3-V micro:bit drive as long as you're USB powered. The SERVO line is multiplexed with the IR_OUT line, so if you keep both resistors in place you'll be driving the motor and throwing random IR bits into the air. We figured you were unlikely to be running a motor and playing IR laser tag at the same time, although as soon as we wrote that we envisioned fighting laser robots…

IR transmit and receive

The micro:bit supports Bluetooth Low Energy (BLE) but that's going to be a significant learning curve for the majority of the target audience. But wireless communication is so cool! The dock simplifies this substantially using an infrared transmitter and receiver that can be used for simple tasks like make-and-break detection between two boards, all the way up to complex communications like IrDA. There are a lot of great applications that can be created to harness the 'magic' of sending and receiving information with invisible bits flying through the air.

Analog

What would an embedded system be without analog? As the saying goes, 'we live in an analog world'. A 10-kΩ single-turn potentiometer is on the dock and connected so you can set any voltage between the supply rails on the analog input. The micro:bit MCU has a 10-bit analog-to-digital converter (ADC), so you can do a lot of great things.
Since we're sharing this analog input with the main pin 1, we included a jumper to completely disconnect the trimmer so it doesn't load any other analog circuits you want to run. This allows for easy testing either standalone or with a quick check of your own projects by patching in the trimmer to test the full range of input that should be coming in from your own system.

Power

Most circuits in the dock will run from the micro:bit's supply, but the intent is to use an external supply. The dock fea-

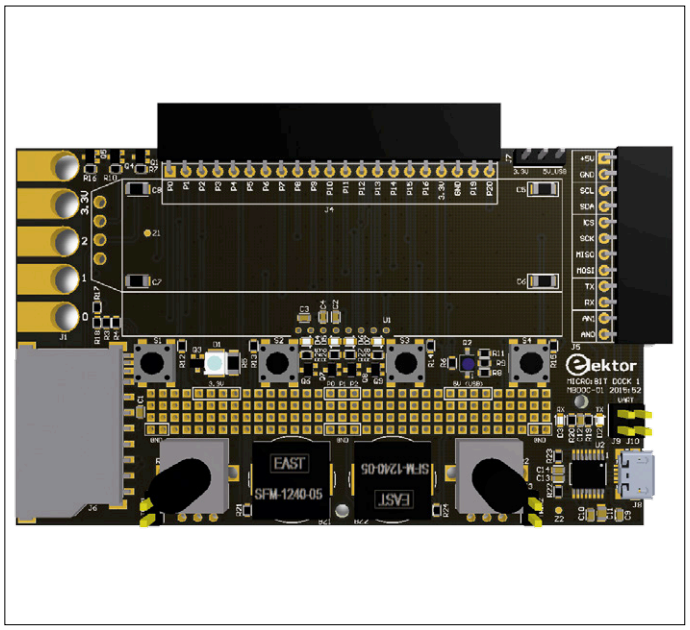


Figure 1. A 3D rendering of our first go at a BBC micro:bit dock.

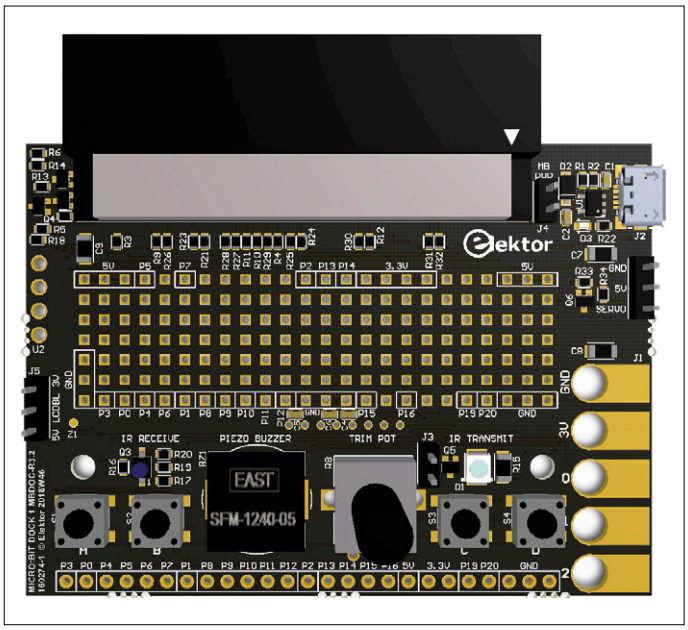


Figure 2. The final iteration of the BBC micro:bit dock looks like this.

tures its own USB micro connector so we can use the 5-V rail and we have a 3.3-V LDO to source the dock and micro:bit. There's a jumper in case you want to keep the micro:bit and dock supplies separate. Any 5-V USB micro supply should work fine, so don't throw away your old phone chargers!

LCD (optional)

The king of the user interface is an LCD, so it seemed essential to have this option as part of the docking board. The display we chose is a really nice 20x2 ASCII character display with an RGB backlight. In this way we add an RGB LED to the system so even if you never write a line of code to turn on the LCD characters, you can create some beautiful lighting effects with your board.

The LCD driver itself is compatible with the classic HD44780 standard but uses an I²C interface to save a boat load of pins. Jumper J5 lets you power the RGB backlight from either the 3.3-V or 5-V

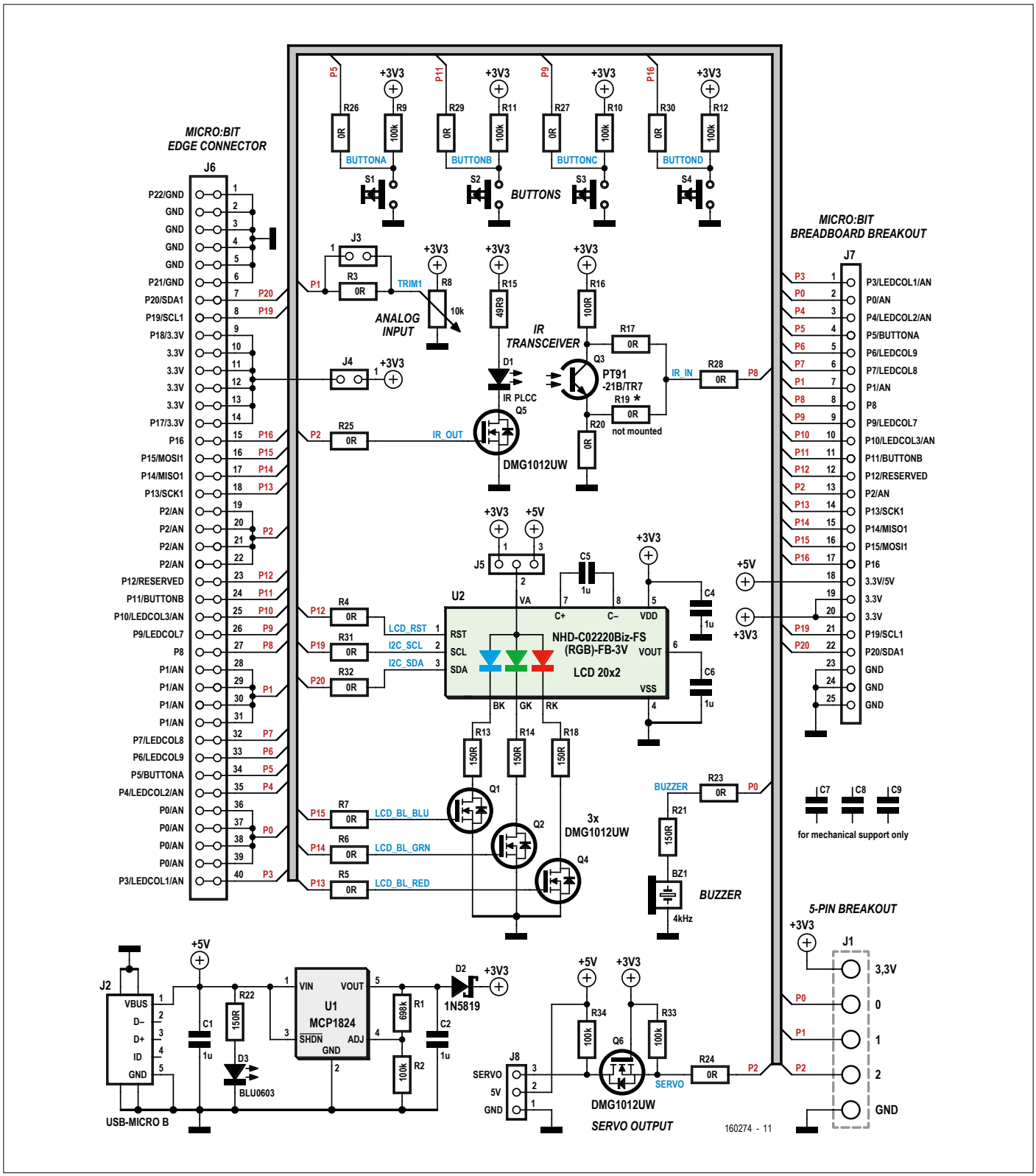


Figure 3. The schematics of our dock.

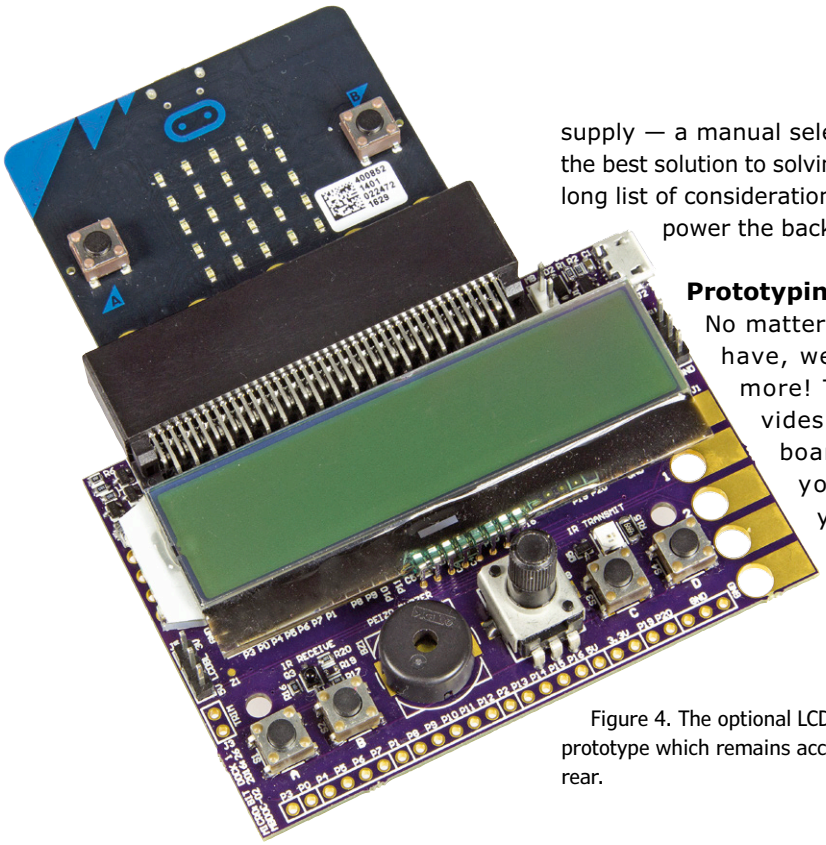


Figure 4. The optional LCD covers the prototype which remains accessible from the rear.

supply — a manual selection that was the best solution to solving a surprisingly long list of considerations about how to power the backlight.

Prototyping

No matter how much we have, we always want more! The dock provides a decent perf board area where you can solder your own circuits. The pads are designed so you can use 0603 SMT components if you prefer that over those old school thru-hole parts. Notice that all of the micro:bit pins are available along with lots of power and ground. Both sides of the board are labeled, so if you add the LCD and cover the front you can still build circuits on the back side.

The 0, 1, 2, GND and 3.3-V alligator clip points are also brought out since they're not available when the micro:bit is docked. The LCD will partially cover 3.3 V and GND (**Figure 4**) but you can still get a clip on and the three I/O lines remain totally accessible for banana jacks.

Lastly, the full array of micro:bit pins are brought out on a 0.1" pitch header at the bottom of the dock board so you can add a connector on whatever pins you need and plug the whole assembly into

Table 1. BBC micro:bit dock pin mapping.

micro:bit pin	Edge Connector Pin(s)	Functions	Dock Main	Dock Alt	Comment
3.3V	10-13	3.3V	3.3V		J4 connects micro:bit 3.3V to Dock 3.3V
GND	2-5	GND	GND		
P0	36-39	P0/AN	BUZZER	P0 Access	Can disconnect with R23
P1	28-31	P1/AN	TRIM1	P1 Access	Can disconnect with jumper J3 or R3
P2	19-22	P2/AN	SERVO	IR_OUT	Disconnect SERVO with R24; Disconnect IR_OUT with R25
P3	40	P3/LEDCOL1/AN			
P4	35	P4/LEDCOL2/AN			
P5	34	P5/BUTTONA	BUTTONA		Disconnect with R26
P6	33	P6/LEDCOL9			
P7	32	P7/LEDCOL8			
P8	27	P8	BUTTONC	IR_IN	Disconnect BUTTONC with R27; Disconnect IR_IN with R28
P9	26	P9/LEDCOL7			
P10	25	P10/LEDCOL3/AN			
P11	25	P11/BUTTONB	BUTTONB		Disconnect with R29
P12	23	P12/RESERVED	LCD_RST*		Disconnect with R4
P13	18	P13/SCK1	LCD_BL_RED*		Disconnect with R5
P14	17	P14/MISO1	LCD_BL_GRN*		Disconnect with R6
P15	16	P15/MOSI1	LCD_BL_BLU*		Disconnect with R7
P16	15	P16	BUTTOND		Disconnect with R30
P17	14	P17/3.3V	3.3V		
P18	9	P18/3.3V	3.3V		
P19	8	P19/SCL1	LCD SCL*		Disconnect with R31
P20	7	P20/SDA1	LCD SDA*		Disconnect with R32
P21	6	P21/GND	GND		
P22	1	P22/GND	GND		

*Only with optional LCD.

your bread board. We kept the pin ordering that appears on the edge connector though we snuck in a 5-V contact, too.

Putting it all together

After identifying all of the features we wanted, it wasn't too difficult to build our schematic and then create the PCB. We paid very close attention to the pin mapping (**see Table 1**) and avoided any LED matrix pins since claiming any one of those would take out all of the LED functionality on the main board. That didn't leave too many options which is why we still had to double up on a few functions.

So what about projects?

We're starting to really like the micro:bit (**Figure 5**) and are hitting the keys to code some great projects for the standalone board and now the dock. Here are a few ideas we're working on:

- Laser tag: use the IR transmit and receive function to "blast your friends away". Tons of options here though we'll see what kind of distance we can reliably get with the relatively low power we have on the IR LED.
- IR beam alarm: Secure your room with an invisible beam of light that will increment a counter or trip an alarm if anyone crosses!

COMPONENT LIST

Resistors (1%, 0.1 W, 0603)

R3-R7,R17,R20,R23-R32 = 0Ω
R19 = 0Ω, not mounted
R16 = 100Ω
R2,R9-R12,R33,R34 = 100kΩ
R13, R14, R18, R21, R22 = 150Ω
R15 = 49.9Ω (1206)
R1 = 698kΩ
R8 = 10kΩ potentiometer, linear

Capacitors

C1,C2,C4,C5,C6 = 1µF (0603)
C7,C8,C9 = any value, 1206, 0.85mm height

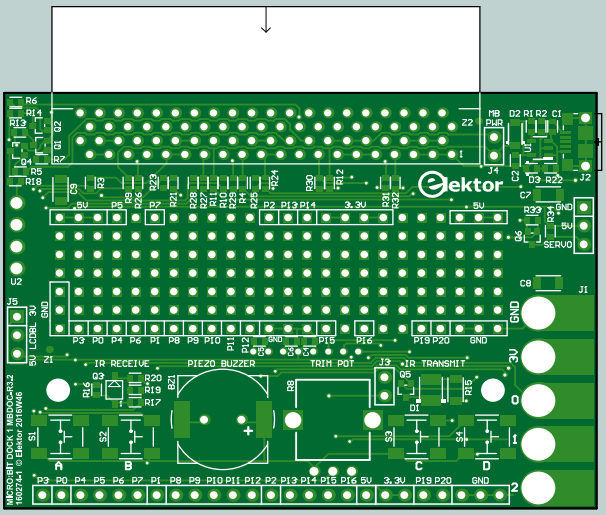
Semiconductors

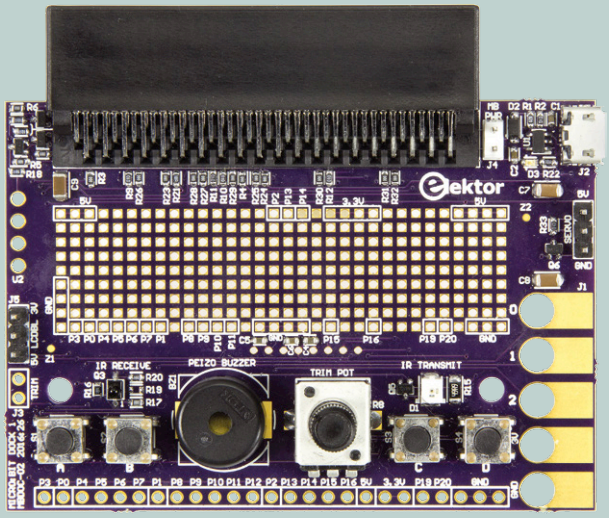
D2 = 1N5819HW-7-F
D1 = IR67-21C/TR8 (IR diode)
U1 = MCP1824-ADJE/OT
U2 = LCD NHD-C0220BiZ-FS(RGB)-FB-3V NNP (optional)
D3 = LED, blue, 0603
Q1,Q2,Q4,Q5,Q6 = DMG1012UW
Q3 = PT91-21B/TR7

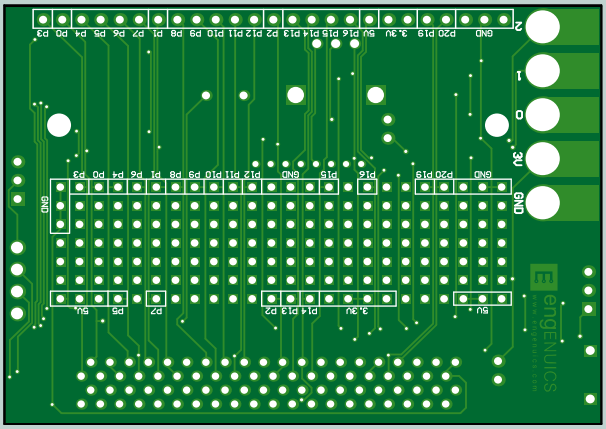
Miscellaneous

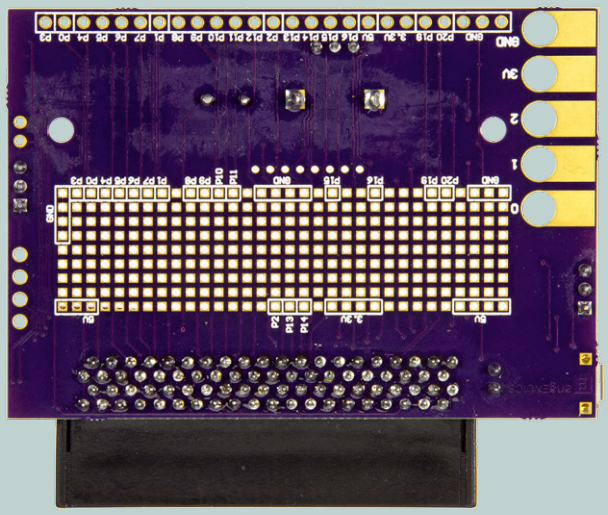
J6 = Edge connector socket, 2x40, 0.05" pitch
J7 = 25-pin pinheader, 0.1" pitch
J3,J4 = 2-pin pinheader, 0.1" pitch
J5,J8 = 3-pin pinheader, 0.1" pitch
J2 = USB micro B connector
BZ1 = Piezo buzzer
S1,S2,S3,S4 = Tactile switch, 6x6mm

PCB 160274-1 rev3.2









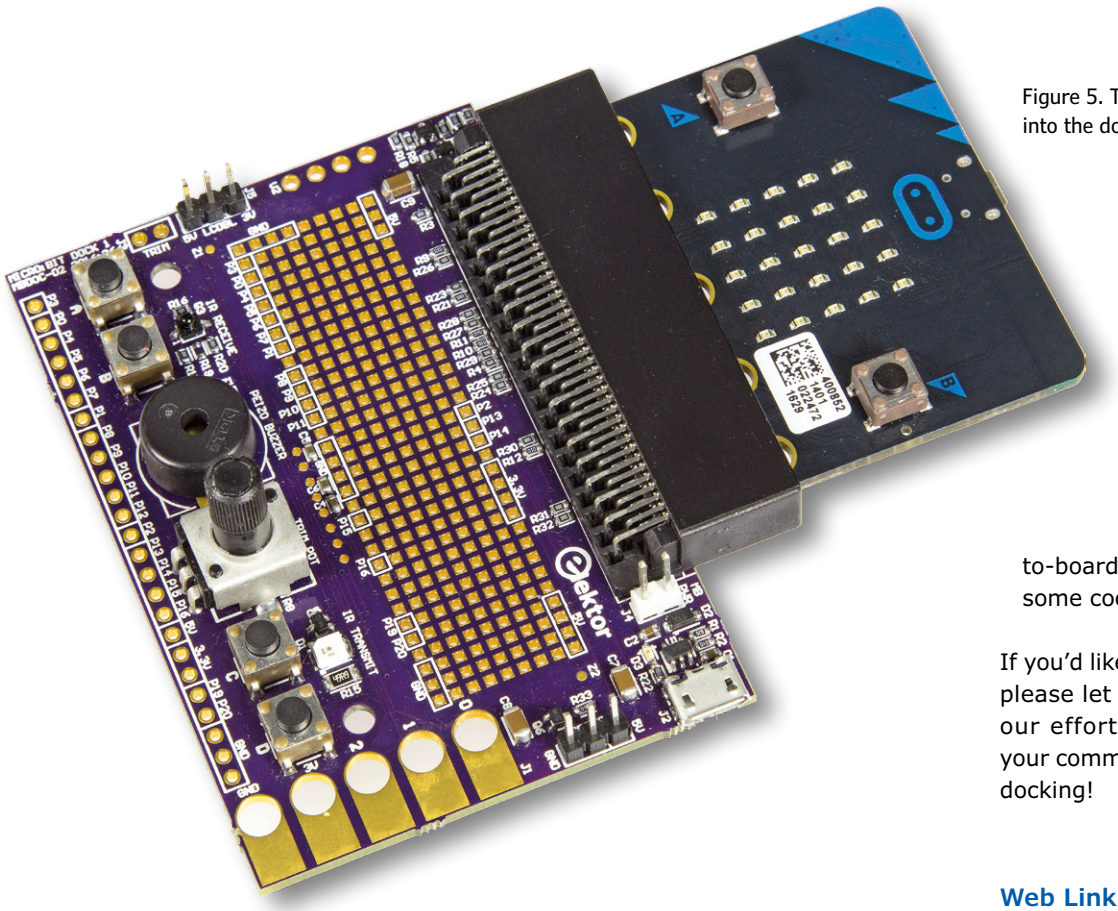


Figure 5. This is how the BBC micro:bit plugs into the dock. Do not stick it in upside-down!

- Light control: use the analog trim pot and BLE connectivity to dim the lights in the room.
- Simple text messenger (with LCD): though the UI would be a little difficult (think dialing-in character by character), you could send messages board-to-board or even board-to-phone. You could code some standard favorite message and select them with the buttons.
- Alarm clock (optional LCD): build your own fully featured alarm clock
- Annoying buzzer 1: Use the trim pot and the buzzer to dial in any tone you wish. If you have an oscilloscope, looking at the PWM signal is a great way to explain what audio actually is.
- Annoying buzzer 2: set a high frequency tone and hide the board in your friend's desk. Turn it on via BLE or just on a timer and watch them look for it. Turn it off if they get too close.
- IR remote control: Use your micro:bit and dock to learn and program the main features of your TV's remote. Or code some simple board-to-board communication and build some cool toys or features.

If you'd like to see one before the other, please let us know and we'll prioritize our efforts. As always, we welcome your comments and suggestions. Happy docking!

(160274)

Web Link

[1] www.elektormagazine.com/160274



FROM THE STORE

→ **160274-91**
Ready assembled module

→ **SKU 17773**
BBC micro:bit

→ **SKU 17589**
5V AC/DC 2A Power supply with micro-USB

or

→ **SKU 18012**
Battery holder 2x AA with JST connector

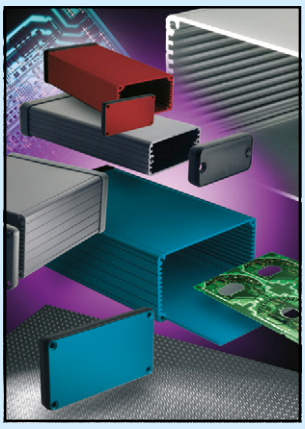
Advertisement

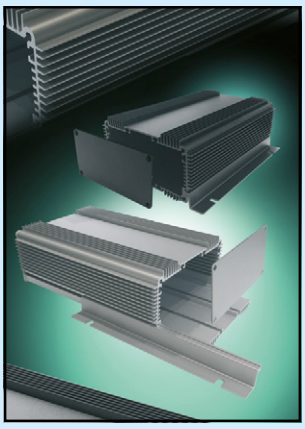
HAMMOND MANUFACTURING®

Extruded enclosures standard and heatsink

www.hammondmfg.com/1455.htm
www.hammondmfg.com/1455NHD.htm

01256 812812
sales@hammondmfg.eu





Bubble Memory

Peculiar Parts, the series

By **Neil Gruending** (Canada)

We've talked about magnetic core memory before, which used toroids as its storage element. They worked well for the era except that they were quite large and expensive. One of the interesting successors to core memory was bubble memory which used tiny magnetized 'bubbles' to store data. Many people thought it would be a universal memory that would replace all other memory types and for a while it looked possible.
Core memory was difficult to manufacture because it had to be built by hand which prompted people like Andrew Bobeck at Bell Labs to research alternatives. One new type of memory was twistor memory that used magnetic tape instead of the toriods in core memory. It was cheaper to produce but it also had a weird side effect where running a current through wires in the tape caused the magnetic fields to move in the direction of the current. This behavior is the basis of bubble memory.
The tape stored bits in relatively large magnetic areas to keep the field stable — these fields were called domains. However, Bobeck discovered that that areas could be shrunk into tiny circles he called bubbles, by applying a magnetic field to the substrate material. Combining the bubbles with Paul Michaelis's work with substrates made it possible for the bubbles to travel orthogonally through a garnet substrate to form bubble memory.

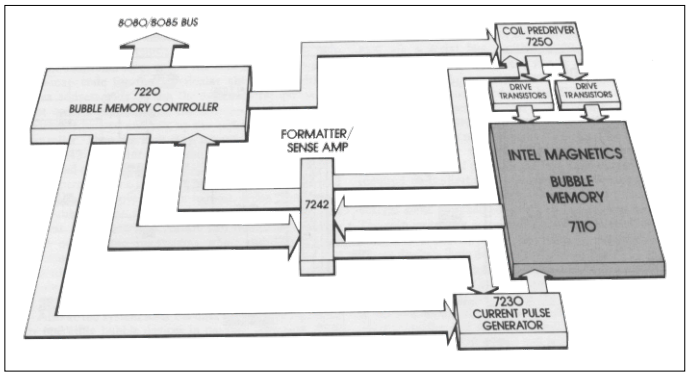


Figure 1. Intel 128-KB Bubble Memory Block Diagram [1]

Figure 1 shows how an early Intel 7110 128 KB bubble memory was constructed. The actual bubble memory portion was laid out as a 2D array on the substrate. One side of the array writes the memory bits to the array, and the other side is used to read them. The bits are moved from the write side to the read side using magnetic coils. The number of bits that can be stored between the ends like a giant shift register is what defines the total storage capacity. Devices were manufactured with excess capacity so that failed memory areas could be masked out. The mask values were then either printed on the label (like the TI TBM0103 in **Figure 2**) or stored internally for the memory controller to access.

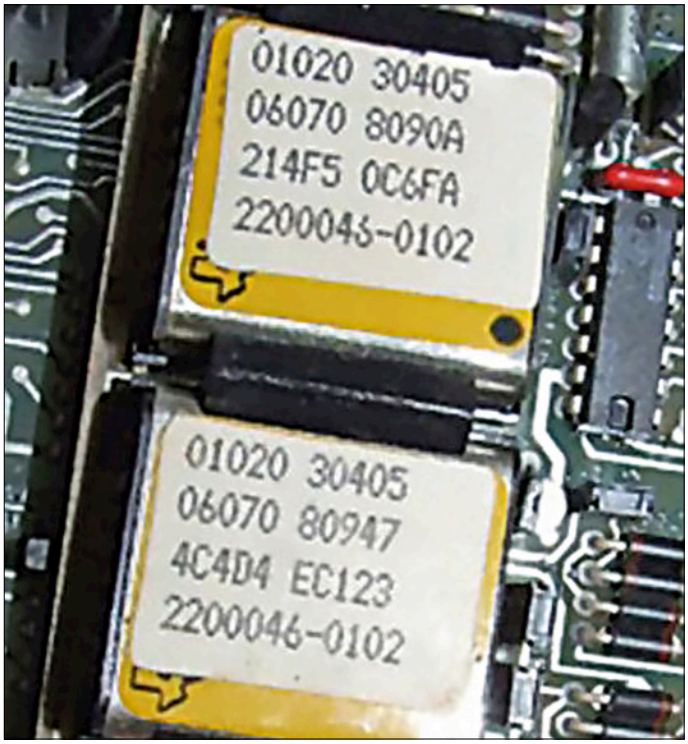


Figure 2. TI Bubble Memory [2]

At one point bubble memory was also called *universal memory* because it was nonvolatile and also allowed 'random' access. Almost every semiconductor company had teams working on it but improved hard disks and RAM quickly replaced it. Bubble memory did find some niche uses though in the early 80s in applications that needed high reliability compared to hard disks that could also tolerate harsh environments. This made them ideal for military applications especially since they don't have any moving parts. However, flash memory with its better density and cost is what finally made bubble memory obsolete.

A lot of information about bubble memory is available on the Internet and used parts are readily available. Maybe you want to give it a try next time you need some nonvolatile memory for your project. ⏮

(160215)

Web Links

[1] Intel Bubble Memory Design Handbook: https://ia802606.us.archive.org/18/items/IntelBubbleMemoryDesignHandbook/BubbleMemoryDesignHandbook.pdf

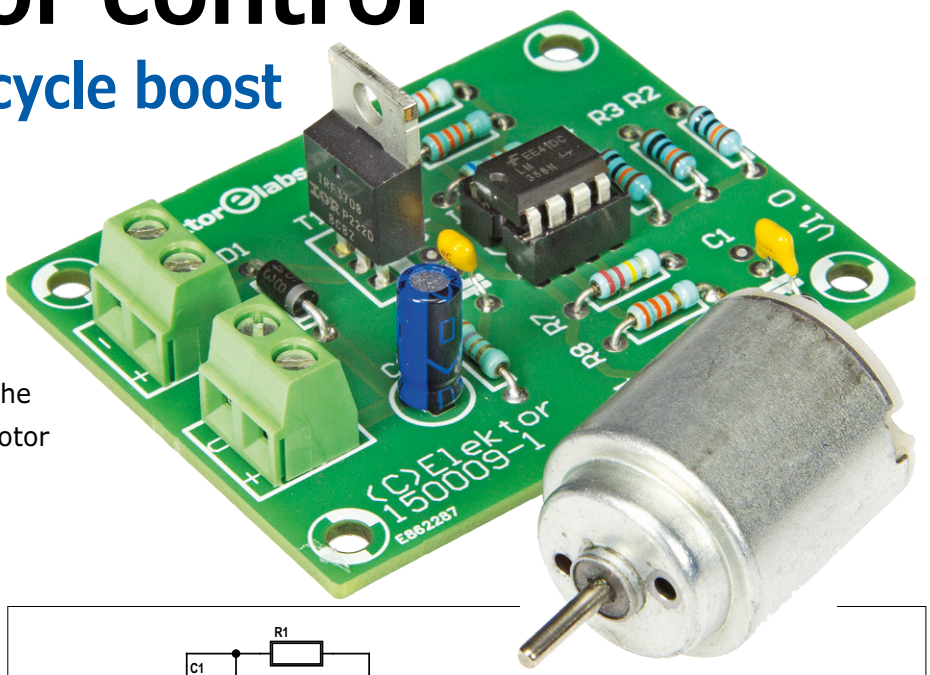
[2] www.wylie.org.uk/technology/computer/bubblmem/bubblmem.htm

eLABS PROJECT

PWM motor control

With added duty cycle boost

By **Rolf Blijleven (Netherlands)**



This circuit is designed to allow a small DC motor to run at low rotational speeds. The motor is driven using a PWM signal, in which the duty cycle increases whenever the motor is forced to work harder.

It sometimes happens that you need to run a DC motor at slow speeds. The standard solution is to drive the motor with a squarewave signal having a low duty cycle. But if the motor is overloaded mechanically, the squarewave may deliver inadequate power, causing the motor to stall. A solution to this problem lies in a PWM (pulse width modulation) circuit with a variable duty cycle.
Circuits offering a manually adjustable duty cycle are of course plentiful, but I wanted a duty cycle that varied automatically. That should not be difficult, I thought. And it should also work without using a microcontroller!
You take a PWM generator, use this to control the motor, measure the current through the motor and feed this back to the PWM generator as a control signal. With a couple of opamps, a MOSFET for the motor control, some resistors and capacitors plus a simple, single 5 V USB power supply it ought to succeed. I started out with the classic circuit in **Figure 1**, taken from the datasheet for the LM358 [1]. (The component numbers have been modified to accord with those in the definitive schematic in **Figure 2**.)

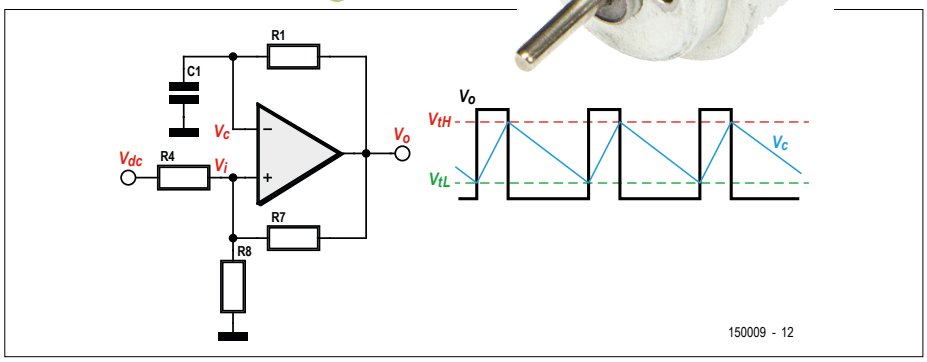


Figure 1. Functional diagram of a pulse generator using a single opamp.

PROJECT-INFORMATION

- motor, PWM, Duty cycle
- ➡ entry level
  intermediate level
  expert level
- 30 minutes approx.
- Soldering iron
  Old motor from a CD/DVD drive
- €15 / £12.50 / $16 approx.

Theory

The basic principle will be familiar to many readers: V_{DC} is a DC voltage (DC for *direct current*, not to be confused with DC for *Duty Cycle* [2]). At switch-on the positive opamp input V_i is always greater than zero, whereas the capacitor voltage V_c at the negative input is still zero. Opamp output V_o goes into saturation at its maximum output voltage (V_{omax}). The output voltage charges capacitor C1 via R1, until V_c becomes greater than V_i. We'll call that state V_{tH}. When this point is reached the output flips back to zero, whereupon the capacitor is discharged via R1 to the point V_{tL} where V_c drops below V_i. Now V_o goes high again and the cycle repeats. The circuit is a combination of a saw-tooth generator and a Schmitt Trigger.
The great thing now is that you can drop these components into a couple of formulae, with which you can calculate exactly which values you need for a particular cycle. We can also approach this a bit more intuitively. Using any 741-type opamp that you choose, V_{omax} will

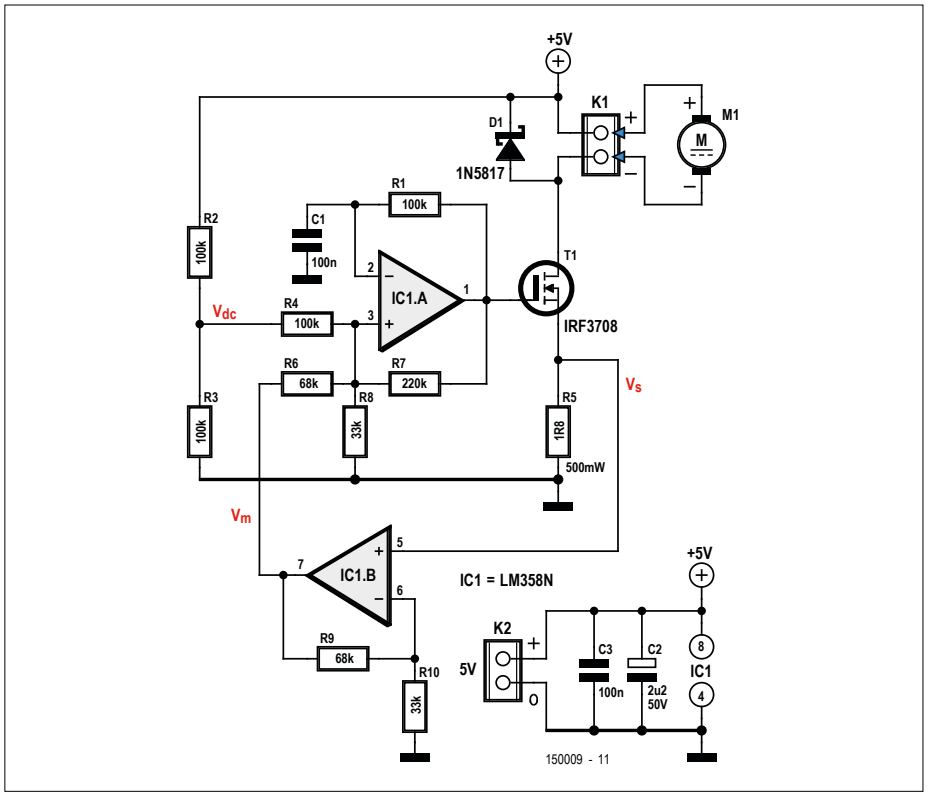


Figure 2. Generator IC1A drives the motor via MOSFET T1, whilst IC1B in conjunction with current sensor R5 assure a degree of positive feedback.

go up to 1.5 V below the supply voltage. If you wish to go all the way up to the supply rail, then take an opamp with an FET output stage, such as the CA3130 or CA3140.

The higher V_{DC} is, the higher V_i will be — and the higher the duty cycle. If V_i is always greater than V_c, then the duty cycle is 100% (in other words the full DC voltage). V_i is the determining variable in the whole affair.

To make this more comprehensible, it may be better to think not in terms of resistance but of conductance. The higher the value of R4 (the lower the conductance of V_{DC} through R4 to V_i), the less V_{DC} contributes to V_i, thus the lower the duty cycle.

The lower R8 is, the more the conductance of V_i to ground (GND) and consequently the lower the duty cycle.

The lower R7 is, the greater the influence of V_{omax} on V_{tH} and hence the greater the difference between V_{tH} and V_{tL}. But what matters in duty cycles is the *sum* of V_{tH} and V_{tL} and this does not change much with R7, meaning that R7 has less influence on the duty cycle than R4 and R8.

At this point you would choose R4, R7 and R8 to have a duty cycle that is always high enough to run the motor. But I wanted a duty cycle that was as low as possible and would automatically increase a little when the engine had to work harder.

Circuitry

You can see the complete schematic in Figure 2 and let me say at the outset that the circuit does not allow itself to be domesticated willingly. More about that more later; the principles of the circuit come first.

R2 and R3 fix V_{DC} at half the supply voltage. The PWM signal at the output of IC1A goes to a MOSFET that switches the motor on and off. Using R5 (1.8 Ω) we measure the current through the motor (nominally around 450 mA). This yields a voltage V_s of 0.8 V at normal loading, rising to 1.2 V when the motor must work harder. In that instance around 700 mA flows through R5. But because the voltage and current are pulsed, the power (I^2R = 0.7² x 1.8 = 0.9 W) in the resistor is reduced by the duty cycle to around 20%. A 0.5-W resistor for R5 will not take any notice of this.

V_s is amplified a little using IC1B and the result goes via R6 to V_i. V_i rises slightly in consequence and as the motor becomes still more heavily loaded, it draws more current as appropriate.

This all sounds very straightforward but selecting the correct resistor values is not so easy. There are still some snakes in the grass. To begin, the frequency of the PWM signal decreases as the duty cycle

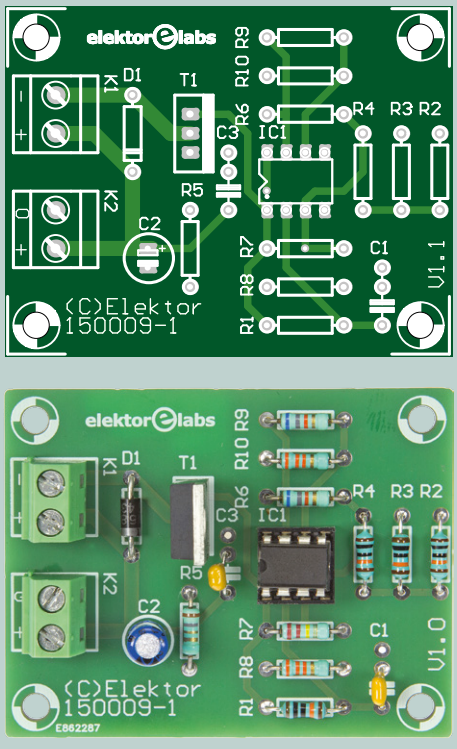
COMPONENT LIST

Resistors
(5%/0.25 W, unless indicated otherwise)
R1,R2,R3,R4 = 100 kΩ
R5 = 1.8 Ω 0.5 W
R6,R9 = 68 kΩ
R7 = 220 kΩ
R8,R10 = 33 kΩ

Capacitors
C1,C3 = 100 nF
C2 = 2.2µF/16 V, radial, 2 mm pitch

Semiconductors
D1 = 1N5817
T1 = IRF3708PBF
IC1 = LM358

Miscellaneous
K1,K2 = 2-way screw terminal block, 0.2" pitch
Small DC motor and gearbox (recycled from an old PC CD drive)
PCB # 150009-1 from Elektor Store



rises. You can see this in the formula. The charging time $t1$ and the discharging time $t2$ of C1 are decisive:

$$duty\,cycle = \frac{t1}{t1+t2} \times 100\%$$

The greater the charging period, the higher the duty cycle, but the frequency is $1/(t1+t2)$ and this then becomes lower. That gives us fewer pulses per second, making the motor speed fall (at a time when you actually want to keep it constant). If you increase the influence of V_m by using a lower value for R6, then you increase the duty cycle but lower the speed even further.

A shorter time constant (R1 × C1) was not a solution because the DC motor made an ominous sound at PWM frequencies above 75 Hz and above 120 Hz the screeching stopped.
Furthermore, the voltage across R5 is a DC voltage with commutation ripple superimposed, caused by switching from one carbon brush to the other while the motor is running. At higher loading, the voltage across R5 is at a maximum and the ripple frequency also increases. This makes sense, because the motor then turns more slowly.
Finally, the motor current (amplified and converted to voltage V_m) was added to the *positive* input of the PWM opamp, achieving positive feedback. According to textbook theory this is a bad idea, in part because the impedance of the voltage source influences the summation. In fact that does not play a major role here. The output impedance of IC1B is very low and much lower than R6. The source impedance of voltage divider R2/R3 here is 50 kΩ, half of R4. I did also experiment with an emitter follower at the voltage divider to reduce the impedance, but that turned out to make hardly any difference to the operation.

If the circuit in Figure 1 using rather simple formulae in a spreadsheet was predictable, then the schematic in Figure 2 displayed some troublesome phenomena trapped in the mathematics. I decided to simply experiment — that is, after all, what breadboards were invented for.
And in this way I got it working. With the values given in the schematic and in the list of components I had a duty cycle of around 17% at 55 Hz, increasing to 21% at 45 Hz under the toughest mechanical stress. This time I did not need to do anything more.

Elektor Labs have designed a small PCB for the circuit, which can be seen in **Figure 3**. This will always be handy in case you want to experiment but you will still have to adjust some resistor values for each type of motor. Next time, if the speed must be kept constant, I think I would design something with a microcontroller.

(150009)

Web Links

[1] LM358 data sheet: www.ti.com/lit/ds/symlink/lm158-n.pdf

[2] http://en.wikipedia.org/wiki/Duty_cycle

FROM THE STORE

→150009-1
PCB

Advertisement

Join the Elektor Community

Take out a GOLD Membership now!





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

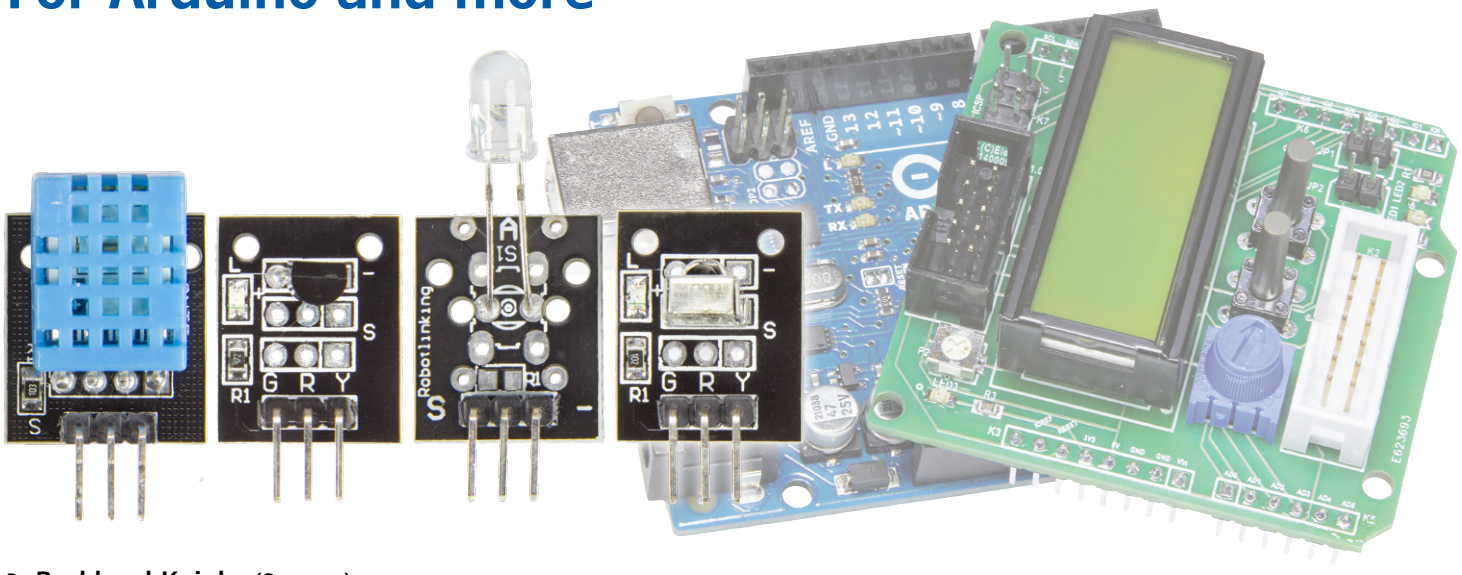
GREEN Membership
- 6x Elektor Magazine (PDF)
- Access to Elektor Archive
- Access to over 1,000 Gerber files
- 10% Discount in Elektor Store
- Exclusive Offers

Also available:
The all-paperless GREEN Membership!

www.elektor.com/member

Sensors Make Sense (3)

For Arduino and more



By **Burkhard Kainka** (Germany)

Ever since the earliest days of telegraphy, electrical communication has been possible over a single wire. Today the 1-Wire Bus and similar protocols function using just one conductor, without the need for an additional clock line. Infrared (IR) remote controls operate on similar principles. In this session we get to grips with temperature and humidity sensors among other things, and in particular with the *IR Transmitter* and *Receiver* in Elektor's 35 Sensors Kit.

We commence with the temperature sensor *18B20 Temp* in the kit, which is available from the Elektor Store [1]. The DS18B20 under discussion may look like a regular transistor in a TO92 package but it's actually a complex IC embracing a temperature sensor and a special interface. The 1-Wire Bus was developed by the Dallas Semiconductor Corporation. One or more sensors can be handled using a single wire, if you discount the GND line. As this wire carries only data, it can also be used to power the IC. In the main, however, people use the third (V_{DD}) connection of the DS18B20 for supplying power, which then adds up to three wires in total. On the sensor PCB there is also an LED plus its dropper resistor, connected to the data line.

Arduino software for the 18B20

There are two Libraries that we need to load into the Arduino IDE: *OneWire* and *DallasTemperature*. Both are supplied on

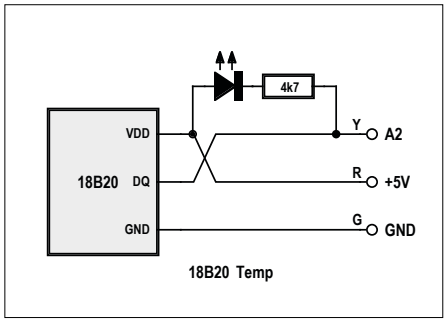


Figure 1. 18B20 temperature sensor connections.

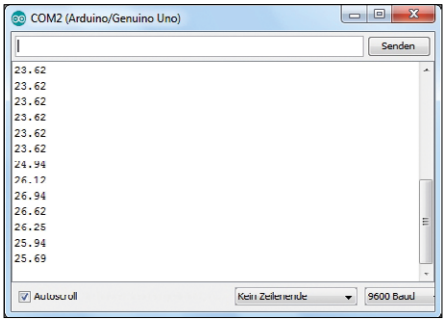


Figure 2. Temperature output in the serial monitor.

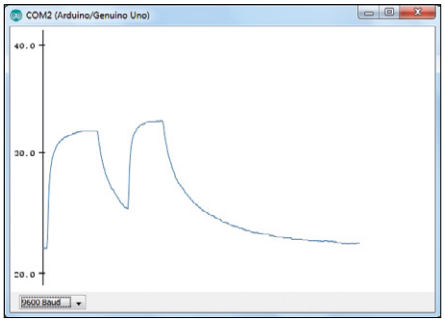


Figure 3. Temperature curve.

Listing 1. Temperature measurement using the DS18B20.

```
#include <OneWire.h>
#include <DallasTemperature.h>
#include <LiquidCrystal.h>
#define ONE_WIRE_BUS A2
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

float temp;
int minTemp;
int maxTemp;
LiquidCrystal lcd(2,3,4,5,6,7);

void setup(void)
{
  Serial.begin(9600);
  sensors.begin();
  lcd.begin(16, 2);
  minTemp = 100;
  maxTemp = -100;
}

void loop(void)
{
  sensors.requestTemperatures();
  temp = sensors.getTempCByIndex(0);
  Serial.println(temp);
  lcd.setCursor(0, 0);
  lcd.print(temp);
  lcd.print (" C   ");
  if (temp < minTemp) minTemp = temp;
  if (temp > maxTemp) maxTemp = temp;
  lcd.setCursor(0, 1);
  lcd.print(minTemp);
  lcd.setCursor(5, 1);
  lcd.print(maxTemp);
  delay (500);
}
```

the Sensor Kit CD and they have to be copied into the *Libraries* folder in the *Arduino Sketchbook* directory. In each case you will find a header file **.h* and a C++ file **.cpp*. It's interesting to look closer at these files. If you manage to get all the way to the end, leaving nothing at all unread, that's pretty good going! After that, all you need do is integrate the header files into your own program and then call up a few functions. Thanks to Arduino, all this is straightforward. The program in **Listing 1** shows how to interrogate and retrieve a temperature pure and simple, repeated every 500 ms. The data line is connected to A2 (= PC2) (**Figure 1**). All the code mentioned in this article can of course be downloaded from the Elektor website [4].
Not only does the thermometer have a serial output but it can also indicate the temperature on the LCD screen, if you are using the Elektor Extension Shield [2] [5]. And because there is space on the second line of the display, we have additionally programmed a minimum/maximum thermometer.

The temperature is indicated as a real number to two decimal places. The actual resolution amounts to 0.06 degrees C per step. The absolute accuracy is given within 0.5 degrees. On the serial monitor (**Figure 2**) you can read the data. A touch on the sensor will demonstrate a change in temperature.
The serial plotter indicates changes in temperature over time (**Figure 3**). In the plot shown the sensor was touched (warmed) twice by finger. This shows clearly the differing time constants for heating and cooling. Also interesting is how the finger was visibly warmer on the second touch. Something must have caused the rise in temperature during the intervening time. Thanks to the high resolution of the sensor even small variations can be detected.

18B20 in Bascom

Bascom supports the 1-Wire Bus, although the Bus alone does not provide a complete solution for using the 18B20. So you need to dive a little deeper (into the data sheet) and do some programming of your own. Of course this takes some time but it does also open up some interesting opportunities. You can then make use of some special features of the sensor, for instance for altering the resolution or for reading out the unique ID number included in every IC.

The Bascom sample code (**Listing 2**) shows how we read out temperatures. Two commands need to be sent, (*Skip ROM, &HCC* and *Convert T, &H44*). Following a delay while the measurement itself is taken, we send the command *Read Scratch-*

The 1-Wire protocol

With the 1-Wire Bus everything is channeled down the single wire DQ. However, we also have GND and V_{CC} wires. In standby mode the data line DQ is taken High with a pull-up resistor. The Master (Controller) can now send a Reset pulse for initiating communication with Slaves, in order to then send commands or receive data. Both partners can load data onto the Bus. A 0 Bit is represented by a 15 µs long Low pulse and a subsequent, 45 µs long High state. In contrast a 1 Bit is symbolized by a 60 µs long Low pulse. Between individual Bits there is a resting state, during which the data line is taken High by the pull-up.
The Master always sends a Reset followed by one or more commands. A sensor chip then responds with the wanted data. If you study the data sheet for the chips in detail, you will find information there not only on the general Bus protocol but also countless commands and the makeup of the data structure that is repeated back. Matters become even more complex, because multiple Slaves can be attached to the same Bus. You can hardly imagine how much work it would take to program all of this yourself. Thank goodness you don't have to reinvent the wheel every time and can refer to ready-made code almost always.

Listing 2. Temperature measurement in Bascom (excerpt).

```
'DS18B20LCD  AD2, PORTC.2
...
Do
  1wreset
  1wwrite &HCC
  1wwrite &H44
  Waitms 800
  1wreset
  1wwrite &HCC
  1wwrite &HBE
  Dat(1) = 1wread(2)
  1wreset
  Temp = 256 * Dat(2)
  Temp = Temp + Dat(1)
  Temp = Temp * 0.0625
  Print Temp
  Tempint = Round(temp)
  If Tempint > Maxtemp Then Maxtemp = Tempint
  If Tempint < Mintemp Then Mintemp = Tempint
  Locate 1 , 1
  Lcd Temp ; " C   "
  Locate 2 , 1
  Lcd Mintemp
  Locate 2 , 5
  Lcd Maxtemp
  Waitms 200
Loop

End
```

pad (&HBE) and then read out the two Bytes. From these the software calculates a 16-Bit number and the temperature (in steps of .0625 degrees). That's pretty clever and even better, the program is only slightly longer than the Arduino version. Once again we have the bonus of a minimum/maximum thermometer with LCD readout.

Temperature and humidity using the DHT11

At first glance the combined humidity and temperature sensor DHT11 looks like a straightforward resistive humidity sensor. This impression is reinforced when you see that an analog Pin is recommended (**Figure 4**). In reality, however, the insignificant-looking exterior conceals a complex sensor with a digital interface.

The Chinese company Aosong has come up here with a scheme that (only at first sight) brings to mind the 1-Wire Bus of Dallas Semiconductor. Each measurement is initiated by a Low pulse from the Master (controlling device) lasting at least 18 ms. After this a total of 40 Bits are read out, which the Master requests each time by means of an 80 µs long Low pulse. The sensor responds with High pulses, in which a period lasting 28 µs maximum stands for a zero and a period of 70 µs for a one. The 40 Bits then contain one High Byte and one Low Byte for the humidity and the temperature plus an additional parity Byte for checking correct transfer of the data. With the DHT11 the Low Bytes are always set to zero, meaning that no post-decimal point figures are transferred. However, there is also a DHT22 device using the same protocol, which does indeed handle the decimal places in addition. Consequently both types of sensor can be interrogated using the same software Library. Once again we are fortunate that someone has already taken the trouble to re-format the complicated protocol into an Arduino Library. Using this is quite simple, once you have copied the Library directory *DHT* from the CD into the Arduino Library folder. The sample code (**Listing 3**) indicates how the two

Listing 3. DHT11 in Arduino-C.

```
//DHT11LCD, pin AD2

#include <dht.h>
#define dht_apin A2
#include <LiquidCrystal.h>
LiquidCrystal lcd(2,3,4,5,6,7);
float temperature;
float humidity;

dht DHT;

void setup(){
  Serial.begin(9600);
  delay(500);
  delay(1000);
  lcd.begin(16, 2);
}

void loop(){
   DHT.read11(dht_apin);
   humidity = DHT.humidity;
   temperature = DHT.temperature;
   Serial.print("Humidity = ");
   Serial.print(DHT.humidity);
   Serial.println(" %  ");
   Serial.print("Temperature = ");
   Serial.print(temperature);
   Serial.println(" C  ");
   lcd.setCursor(0, 0);
   lcd.print(temperature);
   lcd.print (" C   ");
   lcd.setCursor(0, 1);
   lcd.print(humidity);
   lcd.print (" %   ");
   delay(2000);
}
```

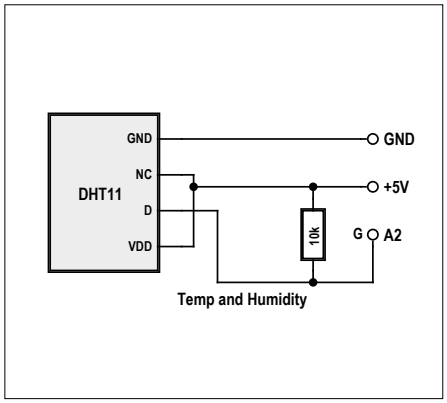


Figure 4. Temperature and humidity sensor DHT11 connections.

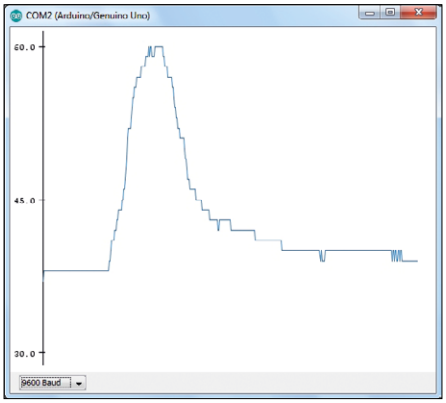


Figure 5. Measuring ambient humidity by finger touch.

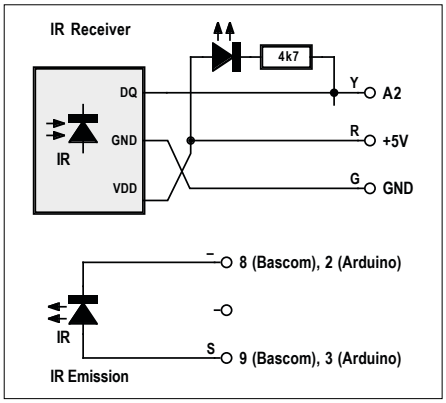


Figure 6. IR receiver and transmitter.

measured values are interrogated and can be displayed on the LCD screen. Once again we select Pin AD2 for connecting the data line.
On the datasheet the absolute accuracy for temperature measurements is stated modestly as within 2 degrees, that for air humidity as within 5 %. Ambient humidity is difficult to judge because all run-of-the-mill moisture meters are fairly imprecise. However, for temperature at least, the results measured on our sample device seemed really good. A digital thermometer that was to hand indicated 23.8 °C, the DHT11 showed 24 °C, and the DS18B20 read 23.37 °C, in each case leaving enough time for the sensor to settle properly.

DHT11 and Bascom

Many ready-to-use commands and functions exist in Bascom of course but the DHT11 is not supported directly as such. Consequently for most topics your most profitable approach is to search the Net to see if someone else has tackled them already. On this occasion our search delivered positive results in the Bascom Forum, where a user by the name of Grütze had written a program called *DHT11LCD.bas* that does exactly what we want. The code is easy to read and reflects more or less exactly what is said in the data sheet. Only minor adjustments are needed for using it with the Extension Shield and enabling the sensor to work using Pin AD2. Beyond this, we also incorporated a serial output, initially only for the humidity data, as it was intended that these would be mapped out using the serial plotter from the Arduino IDE (**Figure 5**).
The significant activity takes place in the function *Get_dht11()*. This is where we carry out exactly the same processes that exist in the corresponding Library for the Arduino. In this way we can create a complete miniature weather station equally well using Bascom (**Listing 4**).

Infrared remote control

We are all familiar with infrared remote controls or 'zappers' for TVs and other home entertainment equipment. Numerous different manufacturers and protocols exist that are not compatible with one another, meaning that a remote control handset must be a correct match for the equipment it commands. One feature common to all designs is that the signal sent by the infrared transmit diode is modulated with a frequency between 30 and 40 kHz. This signal is then pulsed (in one of a number of methods) so as to transmit individual packets of data. An integrated infrared receiver detects the signals with a photo diode, amplifies and filters them and demodulates them back into a digital signal. By design the internal filters are dimensioned for specific frequencies in the range 30 to 40 kHz, although the bandwidth is sufficient to also receive 'off-frequency' signals at shorter ranges.

You can also use an IR zapper for other, more general, remote control or switching tasks. The Sensor Kit includes an IR transmit diode or emitter (*IR emission* in the diagram) and an integrated IR Receiver (**Figure 6**). In conjunction with an Arduino you have the choice of receiving or transmitting IR signals (or both!). Here we will demonstrate a program that can do both. Two buttons are used to send commands that are evaluated in the receiver in order to switch an output. The same assignment is performed both in Bascom and in Arduino-C. Both programs are sufficiently compatible to the extent that that the Bascom controller can tell the Arduino-programmed Uno what is to be switched and vice versa.

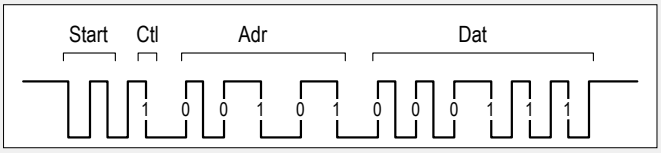
Listing 4. Using the DHT11 in Bascom (excerpt).

```
'DHT11LCD an AD2, PORTC.2
...
Do
   If Get_dht11() = 1 Then
      Print Humidity
      Locate 1 , 1
      Lcd "H: " ; Humidity ; " %  "
      Locate 2 , 1
      Lcd "T: " ; Temperature ; " C  "
    End If
    Waitms 1000
Loop

End
```

The RC-5 protocol

Infrared remote controls for TV receivers, video recorders and other home entertainment devices operate in part using the RC-5 standard defined by Philips. This employs modulated optical signals in the range 30 kHz to around 40 kHz. The remote control sends individual bursts (packets of data pulses) 0.888 ms or 1.776 ms in length. At a modulation



frequency of 36 kHz a short burst contains 32 individual pulses and a long one 64. The complete data packet lasts about 25 ms and is repeated every 100 ms for as long as a button is pressed.
The protocol employs a bi-phase signal. A Bit has a length of 1.776 ms. If the 36 kHz pulse lies in the first half of this time period, it represents a logical Zero; a logical One is signaled by a pulse in the second half. The signal is introduced every time with a start sequence that never changes. There then follow three data fields:

- The Control or Check Bit (Ctl) alternates between 0 and 1 with every key press. In this way the receiver can differentiate whether a key has been pressed once for a long time or several times briefly.
- The Device or System Address Bits (Adr) comprise 5 Bits, in which the high-value Bits are transferred first. Common device addresses are 0 for TV sets and 5 for video recorders. In this way several remote controls can be deployed in the same room.
- The Data or Command Bits (Dat) comprise 6 Bits for up to 64 differing keys (pressbuttons). The number keys 0 to 9 generate codes from 0 to 9. Here too the highest value Bits are sent first.

A standard frequently used for IR remote controls is RC-5, developed by Philips (see boxout panel). Simple commands for this are provided in Bascom, making decoding zappers a simple task. **Listing 5** provides a simple RC-5 receiver and transmitter in Bascom. All received data is displayed on the LCD screen. The command is also sent over the serial interface. At the same time a check is made for any output to switch to Port B. Given that LED2 is available at B.2 on the Extension Shield, we have designed the code to illuminate this with button 2 on the zappers and switch it off with button 0.

The RC-5 receiver can be connected to any input Pin you choose. So we have again deployed input PC3 (AD3) for this. For the command *Getrc5* Bascom uses the *Timer0* Interrupt in the background, which you must enable globally. In addition we have switched in the internal pull-up resistor for input PC3. It is then possible to use the controller for transmitting (only) without an IR receiver connected. Had we not done this, an open high-impedance input without a pull-up might otherwise assume a Low state and cause the program to hang.

For the transmit output (command *Sendrc5* in Bascom) we normally use PB1 (Arduino Pin 9), because the output OC1A of Timer 1 is connected to this Pin, used for generating the 36 kHz transmit signal (warning: in the Arduino-C++ software a different output is used for this task, meaning that on this occasion we cannot keep the same hook-up allocations). In Bascom the standby state of this Portpin must be defined in advance using a Port command; here it needs to be Low during inactive intervals. For each pulse packet the software then switches over from the Port to the timer output. Because the IR diode has no series resistor, it makes sense to switch this to a different output Port, in order to use the two internal resistors for current limiting. Here we selected PB0, which is also switched Low as an output. In that way we still have four outputs on Port B as potential switching outputs for received data. In transmit mode two commands are supported. If you press button S1 on the Extension Shield, then the code for button 2 on the remote control is transmitted. This switches on the output at the receiver, making LED2 come on. With button S2 you send a zero and switch off the output at the receiver once more.

Arduino and IR

For programming in Arduino-C we need to install the *IRremote* Library (**Listing 6)**. This makes use of Timer2 and its output OC2B (PD3, Arduino Pin 3) for generating pulses applied to the IR LED. This is also the E wire of the LCD screen on the Extension Shield. That unfortunately means the program cannot interact simultaneously with the LCD. However, that's not totally bad news, as we still have the serial outputs for viewing the received data.
The input is a matter of choice and is assigned here to A2 once more. Programming an input pull-up is overwritten by the Library, rendering this ineffective. But it's not even necessary, as reception does not block the program even when an open input is in the Low state. Our goal remains achievable, to use a program without modification or reconfiguration for a choice of receiving only, transmitting only or handling both tasks.

The crucial advantage of the *IRremote* Library is that it can 'speak' not only RC-5 but also a multiplicity of other standards. Most of us have a whole collection of disparate remote controls at home. If so, it makes good sense to point each of these once at the IR Receiver. You will then get a report of the standard employed and the data received. RC-5 is shown as Type 3. If you press several times on button 2 of an RC-5-compatible remote control, the following messages appear:

```
3
382
3
B82
```

The data is output as 12-Bit numbers. The lower 5 Bits denote the key code. Next comes the 5-Bit device address, in this case the address 14 for a DVBT receiver. The highest value Bit is the

Toggle Bit, which changes with every key press and does not need to be evaluated. It's worth noting down this information, so you can resend it identically. You can ignore the Toggle Bit when doing this. If you transmit 382_{hex} via the IR diode, this corresponds to button *2* in RC-5 code.
Because in this situation the Arduino has to get by without the Extension Shield, the switching output is assigned to connector 13. Doing this enables us to control the LED on the Arduino. The relay can now be connected to Portpin 13 and straightaway a load can be switched either with an RC-5 remote control or else by a second Arduino with an IR diode.
Here too the pressbutton commands 2 and 0 can be transmitted back in the reverse direction. As in the Bascom version, the relevant buttons are assigned to A0 and A1, in which the internal pull-ups have been enabled. Here you can hook up plenty of the things that the sensor kit has in store. Of course these do not have to be touch switches every time. Why not use a magnetic sensor, a position sensor or an optical sensor? You could turn an infinitely large number of ideas into reality. The TV could be turned off by sunrise at the latest, or make the position sensor on the door handle recognize when somebody enters the room and then switch on the lights and the radio to welcome them!

(160210)

Web Links

[1] www.elektor.com/arduino-sensor-kit
[2] www.elektormagazine.com/160152
[3] www.elektormagazine.com/160173
[4] www.elektormagazine.com/160210
[5] www.elektormagazine.com/140009

Listing 5. RC-5 transmitter and receiver in Bascom.

```
'RC5LCD  In AD2, PORTC.2, Out OC1A, PORTB1
...
Do
  If S2 = 0 Then
    Togbit = 0 : Address = 0 : Command = 2
    Do
      Rc5send Togbit , Address , Command
      Waitms 100
    Loop Until S2 = 1
  End If
  If S1 = 0 Then
     Togbit = 0 : Address = 0 : Command = 0
     Do
      Rc5send Togbit , Address , Command
      Waitms 100
    Loop Until S1 = 1
  End If
  Getrc5(address , Command)
  If Address < 255 Then
    Locate 1 , 1
    Lcd Address ; "   "
    Locate 2 , 1
    Togbit = Command / 128
    Lcd Togbit ; " "
    Locate 2 , 5
    Command = Command And &B01111111
    Lcd Command ; "   "
    Print Command
    If Command = 2 Then Portb.2 = 1
    If Command = 0 Then Portb.2 = 0
  End If
Loop
```

Listing 6. IR control in Arduino-C.

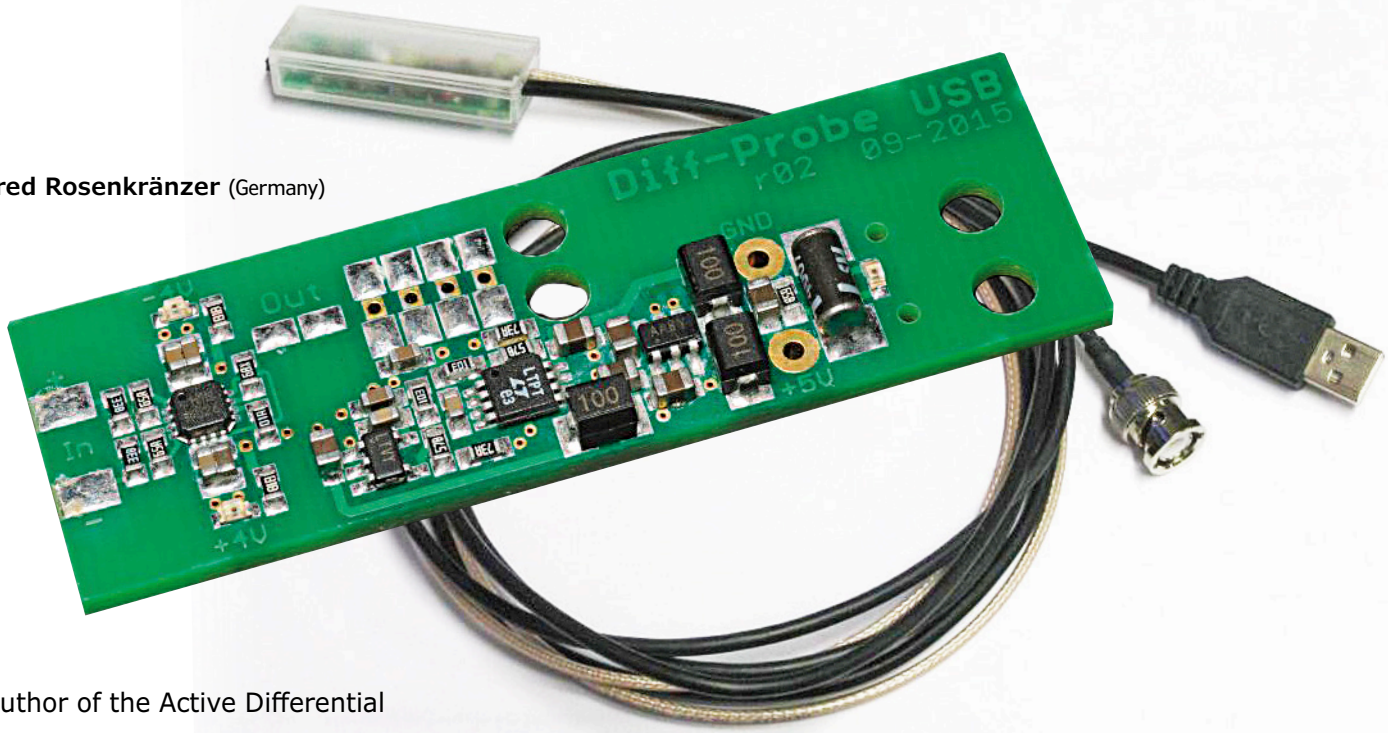
```
#include <IRremote.h>
int RECV_PIN = A2;
IRrecv irrecv(RECV_PIN);
IRsend irsend;
decode_results results;
int d;
int S1 = A0;
int S2 = A1;
int LED = 13;
int kathode =2;
void setup()
{
  Serial.begin(9600);
  irrecv.enableIRIn();
  pinMode(S1, INPUT_PULLUP);
  pinMode(S2, INPUT_PULLUP);
  pinMode(RECV_PIN, INPUT_PULLUP);
  pinMode(LED, OUTPUT);
  pinMode(kathode, OUTPUT);
}

void loop() {
  if (irrecv.decode(&results)) {
    Serial.println(results.decode_type);
    Serial.println(results.value, HEX);
    d = results.value & 15;
    Serial.println(d);
    if (d==2) digitalWrite(LED,1);
    if (d==0) digitalWrite(LED,0);
  }
  if (digitalRead(S1) == 0) irsend.sendRC5(0x382,
32);
  if (digitalRead(S2) == 0) irsend.sendRC5(0x380,
32);
  irrecv.enableIRIn();
  delay(100);
}
```

HOMELAB PROJECT

Active differential probe v2

Now USB-powered



By **Alfred Rosenkränzer** (Germany)

The author of the Active Differential Probe project (Elektor 4 2015 [1]) has developed a new version of this circuit, in which the supply voltage is now taken from a 5-V USB connection.

In **Figure 1** you can see the block diagram of the new power supply arrangements. After the 5-V supply from the USB connector has been filtered, a charge pump generates an unregulated negative voltage of (roughly) the same magnitude as the input voltage. Following some additional filtering, a negative low-dropout regulator brings this down to −4 V. A positive low-dropout regulator generates +4 V directly from the filtered USB supply.

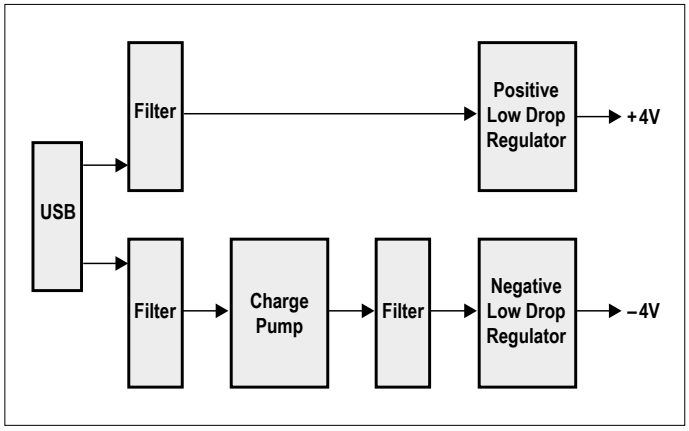


Figure 1. Block diagram of the new USB power supply arrangements.

Figure 2 shows the complete schematic. IC1, an LT1962EMS8, provides the stabilized positive supply voltage. The same function is performed for the negative voltage by IC2, an LT1964ES5. A MAX1697 (IC3) serves as charge pump.

Presence of the USB voltage together with the two regulated voltages is indicated by LEDs (D2, D3 und D4). Coils L1 and L2 take care of filtering the 'raw' USB voltage.

The circuitry of the differential amplifier (IC4) remains largely unaltered. Only the two series-connected input resistors on each input are each combined in a single module. This enables the op-amp to be moved forwards on the PCB, in order to make room for the power supply components.

In contrast to the schematic published in Elektor 4/2015, some alterations have been made. As non-isolated voltage converters have been used, the GND connection of the USB connection has been commoned with the signal GND.

Despite elaborate filtering, the clock frequency of the charge pump (around 200 kHz) is visible on a spectrum analyzer at < −80 dBm. On an oscilloscope it cannot be detected, however.

The last thing to mention is that the construction cost has risen, on account of the expensive regulators.
The project still fits in the USB case used previously and the BNC cable remains unaltered. ⏮

(150801)

For readers who may be interested the author offers ready-made probes. Info: alfred_rosenkraenzer@gmx.de.

Web Link

[1] www.elektormagazine.com/130538

Specifications

Attenuation:	10:1 with differential signal and 50 Ω termination
Differential input resistance:	5100 Ω, ±1%
Single-ended input resistance:	2550 Ω, ±1%
Input common mode range:	±12 V
Output resistance:	50 Ω, ±1%
Bandwidth:	approx. 1.9 GHz (−3 dB)
Rise/fall time:	300 ps
Power supply:	USB 5 V, approx. 70 mA

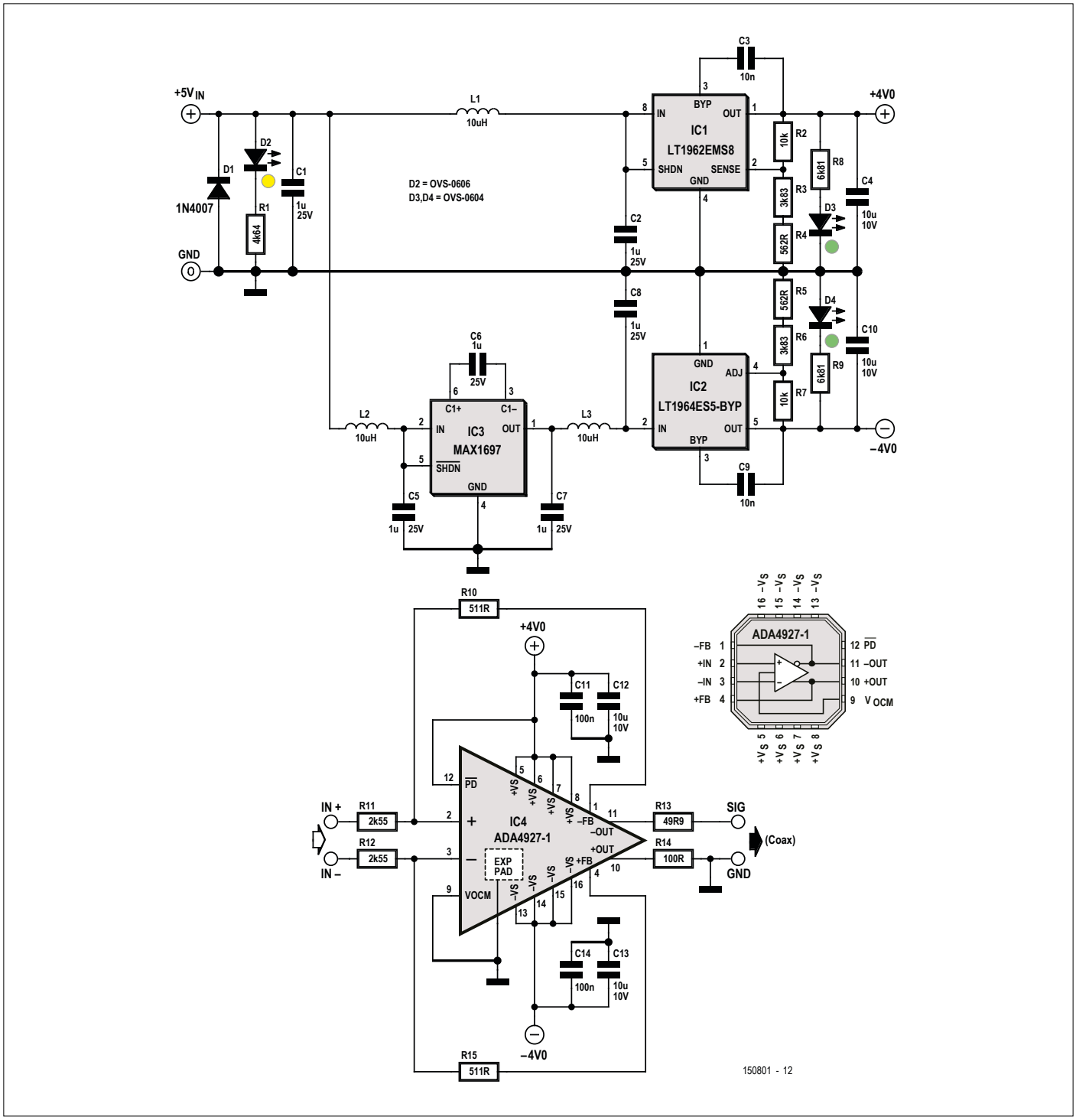


Figure 2. The complete schematic. The section surrounding the differential amplifier (IC4) is fundamentally unaltered.

RFID starter kit for the Arduino Uno

Temperature sensing and door entry system



Using the RFID starter kit from Elektor provides a good basis to carry out numerous electronic experiments with the Arduino Uno. We show just how simple it can be to realize an application.

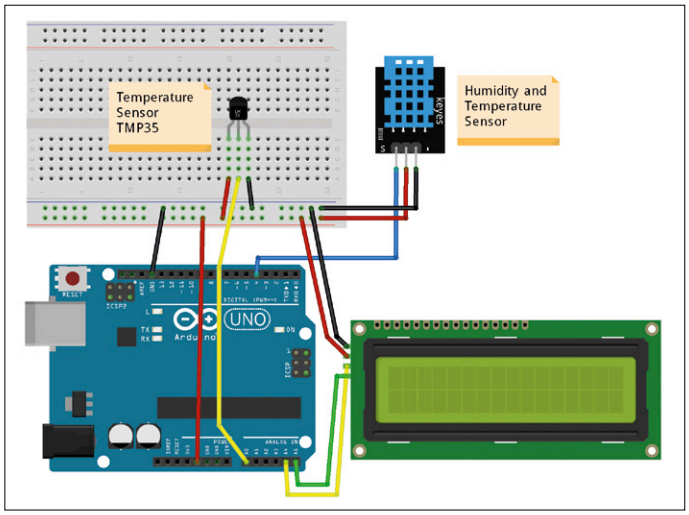


Figure 1. The weather station with two sensors and a display.

The Arduino system or one of its many clones is now often the keystone in many projects built by engineers and makers. Not so long ago a lone microcontroller was the basic building block for any controller-based design, nowadays it's an Arduino board. It has been responsible for introducing computer technology to a whole new generation of users. Instead of needing to get to grips with a relatively complex programming environment, today's makers and electronics tinkerers have the luxury of intuitive development tools in an integrated development environment that is relatively easy to dive into even for a complete beginner.

The situation is a little different from the hardware standpoint. Here developers find themselves largely out on their own. There is of course a wide choice of different shields that plug onto the Arduino board to expand its capabilities but they are often designed for one specific application. To develop new innovative projects it's often necessary to resort to interfacing discrete electronic chips and devices to the controller. This can be a problem for newbies.

For help here we can find some kits containing all the components required to build some projects. Kits like this were popular in the 70s and 80s, at that time you could build things like a siren or a basic medium wave receiver; nowadays we have the added versatility of microcontroller-based projects. One interesting example of the new builder's kits is the 'RFID Starter Kit for Arduino Uno'. This comes in a handy case and contains over 30 state-of-the-art components, devices and modules. The name of the kit is a bit misleading because although it contains an RFID receiver module along with two RFID tags in the form of a credit card and key fob the case is an Aladdin's cave with loads of other useful components.

To begin you will need an Arduino Uno along with the starter kit which amongst other things contains:

- A humidity sensor;
- A multicolor LED;
- A large LED-Matrix with 64 LEDs;
- 4 x 7-segment LED displays;
- A handheld IR remote controller plus IR receiver chip;
- A complete LC-Display module with I2C bus interface.

A more comprehensive list of all the kit contents can be found at [1]. What we go on to describe here is just two example

applications that can easily be built using this box of goodies. The wide range of peripherals included in the kit ensures the number of different experiments and applications you can build Universal weather station with LC-Display.

In the first example application we interface a number of different sensors to the Arduino board to measure environmental data. The DHT11 temperature/humidity sensor is used as well as the LM35DZ temperature sensor. This allows us to build a system to measure and display the values of indoor temperature and humidity as well as the outdoor air temperature.
An LC display is used to display the values. The wiring of all the hardware is shown in **Figure 1** and the sketch is given in **Listing 1**.

The **LM35 Sensor** is calibrated during manufacture to supply an output voltage characteristic of 10 mV/°C. In addition it outputs 0 V at a measured temperature of exactly 0.0 °C. The output voltage level will be measured by an ADC with 10-bit resolution (is outputs a value in the range of 0 to 1023). The LM35 has a 5 V supply so the measured temperature is given by the formula:

```
temp = (5.0 * analogRead
(tempPin) * 100.0) / 1023;
```

This however will give quite a poor resolution for the measured temperature range. The LM35 generates a full range output voltage from 0 V to just over 1 V. If we use 5 V as the reference voltage for the ADC it means that 80 % of the available input range will never be used and the measurement resolution of the limited range we are using will be quite poor. To get maximum resolution we can set the internal reference voltage to 1.1 V.
With a reference of 1.1 V the formula for temperature needs to be changed. Now the measurement resolution is equal to 1.1 V divided by 1023 which comes out at 0.001075 V or 1.0752 mV. With the output characteristic equal to 10 mV/°C the resulting conversion factor equals:

```
float TempCal = 0.1075; //1.075
/ 10
```

The instruction in the code to give the temperature is:

```
tempLM35 = (LM35val * TempCal);
```

To enable the internal 1.1 V reference voltage source we use the Arduino instruction:

```
analogReference (INTERNAL);
```

Now using this reference voltage the system achieves a resolution of one tenth of a degree Celsius. The reference voltage will of course be subject to tolerances which can be compensated for by adjusting the multiplication factor 0.1075. Once an exact value is determined you can expect the displayed measurements to be accurate over a long period of time because the reference voltage will only drift by a small amount.

With this setup we can measure temperatures from 0 to 110 °C. One tenth of a degree is represented by 1 mV output change and it should also be noted that heat produced by the controller and associated circuitry can be transferred via the sensor leads and will influence the measured air temperature. For use as a room thermometer it's sufficient to display the value to the nearest whole degree but for applications requiring more accuracy don't forget to consider these effects.

The combined **Temperature/Humidity sensor type**

Listing 1. Measure environmental conditions with the DHT11 and LM35.

```
// DHT11_LM35_w_LCD_display.ino

#include <dht.h>
#include <LiquidCrystal_I2C.h>
#include <Wire.h>

LiquidCrystal_I2C lcd(0x27,16,2); // LCD address 0x27
dht DHT;
const int DHT11_PIN= 4;
const int A0 = 0;
float tempLM35 = 0;
long LM35val = 0;
float TempCal = 0.1075;

void setup()
{ lcd.begin();
  lcd.backlight();
  analogReference(INTERNAL);
}

void loop()
{ DHT.read11(DHT11_PIN);

  LM35val = analogRead(A0);
  tempLM35 = (LM35val * TempCal);

  lcd.setCursor(0, 0); lcd.print("Ti="); lcd.print(DHT.temperature,0);
  lcd.print(char(223)); lcd.print("C");    // print unit " °C "

  lcd.setCursor(8, 0); lcd.print("Ta="); lcd.print(tempLM35,0);
  lcd.print(char(223)); lcd.print("C");    // print unit " °C"

  lcd.setCursor(0, 1);
  lcd.print("Humidity:  ");
  lcd.print(DHT.humidity,0); lcd.print(" %");
  delay(200);
}
```

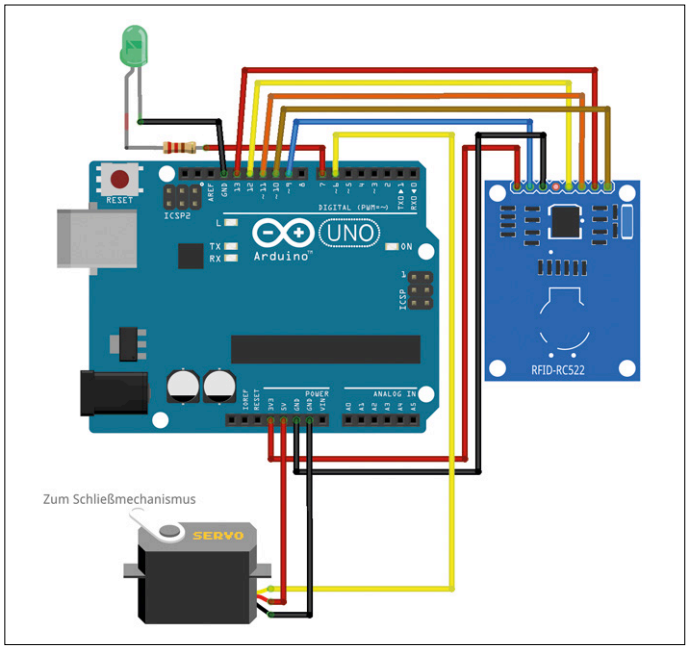


Figure 2. The RFID module plus servo control.

DHT11 communicates using an I2C bus. A complete set of software library functions ensures that reading the measured values is really simple. The **Display** uses the I2C bus and there is a comprehensive set of library functions available also. The function:

```
lcd.print()
```

can be used to display text or measured values. Integrated cursor control ensures problem-free formatting of the values. In this surprisingly compact sketch we use the LiquidCrystal_I2C.h [2], DHTLib [3] and Wire.h libraries. The Wire library is needed for I2C bus communications and like the SPI and Servo libraries (used in the following sketch) are some of the standard libraries included in the Arduino IDE. Both of the other libraries mentioned can be downloaded from the internet free of charge. Should any of the links be dead just use a search engine to find an alternative source.

The weather station can of course be expanded if required by adding more sensors from the RFID kit. The water-level sensor or photoresistor (LDR) can also be used to provide information

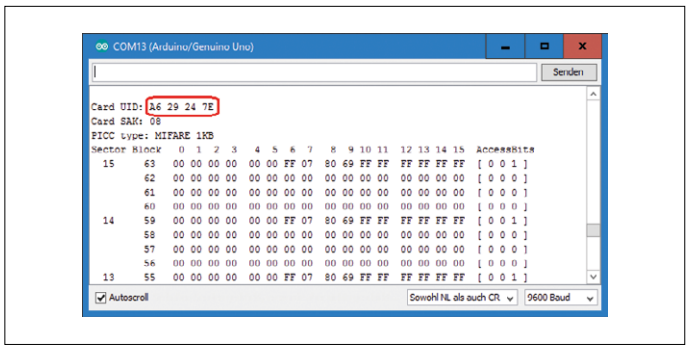


Figure 3. RFID transponder data shown on the Serial Monitor.

to the Arduino. The water-level indicator could tell you if it is raining and the LDR could provide information on light levels which you could then use to record information on sunlight intensity and duration.

Door entry system using RFID security

Another interesting application that can be built with the kit is an electronic door entry system using an RFID tag as a key to open the door. The RFID module included with the kit actually gave the kit its name although it's just one of a bunch of useful items included in the 'RFID Starter Kit'.

Communications to the RFID module take place with the Arduino using the 'Serial Peripheral Interface' (SPI). This type of interface can support communication with many modules all of which use a common Serial Clock (SCK) signal generated by the master controller on the bus to synchronize data exchange. Data is passed using the common Master Output, Slave Input (MOSI) and Master Input, Slave Output (MISO) signals.

In addition to these common signals, each module has its own Slave Select (SS) or chip select signal which is also generated by the master controller when it wants to communicate with the slave module. Each module also has its own reset (RST) input. An Interrupt Request signal (IRQ) can also be generated by a module but we don't use this feature in this application. The RFID module is hooked up to the Arduino as shown in **Table 1**.

Looking at the software we will be using the Arduino SPI-Lib from the Arduino-IDE and the MFRC522-Lib for the RFID module von [4].

With the hardware hooked up according to **Figure 2**, the 'RFID-RC522_data' sketch (**Listing 2**) in the download packet [5] can be flashed to the Arduino. Now when you click on the 'serial monitor' button in the Arduino IDE you will be able to read data sent by the RFID transponder module. When an RFID fob or card is close enough to the coil in the module the data stored in the tag will be displayed (**Figure 3**).

One important parameter which will be used in the sketch is the UID (**U**nique **ID**), a number unique to the tag. The other information stored in the tag is not used here.

An important application of RFID tags is as a key for door entry control. Authorized personnel will carry a RFID card or key fob which allows them to enter a controlled room. At the entrance will be a RFID reader which only releases the door lock when a tag with a valid UID comes within range. An RFID entry system has a number of advantages compared to systems that use traditional keys. Should a tag get lost, it's easy to delete its UID from the list of authorized personnel and issue a new tag, security will not be compromised and there is no need to replace the lock barrel and issue new keys. The price of a tag is also much less than a metal security key blank.

To recognize a particular UID it's simpler if the hex values stored in the tag are first converted to decimal. This makes it easier to check if the code is valid. The UID is made up of four blocks of hex values and here for simplicity we don't check all

of the blocks so the system is less secure than it could be. For maximum security it would be better to evaluate every block but this increases the work involved considerably.

The 'RFID-RC522_servo_lock.ino' sketch for this project is in the download packet for this article. The sketch outputs the UID and its decimal equivalent to the serial monitor. The decimal value can now be used in the declaration:

```
long validCode = 938350; // enter valid code
```

...to assign the card ID to the 'validCode' variable. The system can be used as a door-entry controller; the door will only be unlocked when the system reads a tag containing the correct UID.

A servo motor is included with the RFID kit and this can be used to operate a sliding door bolt. **Figure 2** shows how the servo is hooked up to the Arduino. A green LED is also connected to the board and indicates that the door (or locking mechanism for a locker) is unlocked. The sketch has been expanded at the end to include some basic servo control commands from the servo-library. Make sure that any electronic door entry system you install cannot prevent you from quickly exiting a building in the event of an emergency or power failure.

Summary and outlook

Most modern electronic experimenter kits contain modules that just plug together rather than individual components. This means that we don't get so close to the hardware but reflects the trend in electronics system design so that now we can quickly get modules talking and then decide in software how the system behaves, that can also be quite challenging. Thanks to the wide selection of software library functions we can have a sketch up and running in no time.

These kits are worthy successors to the bags of loose components that were a feature of earlier experimenter's kits. Electronic newbies, practicing engineers and old hands alike are sure to find that these kits interesting. The range of components is sure to get you thinking what you could use them for and they are a good way to build on your existing knowledge and you'll have no excuse not to go on to design and build your own system!

(160322)

Table 1. Hook up of the RFID module and Arduino.

RC522-Pin	Pin Arduino Uno
VCC (3V3)	3.3 V
GND	GND
RST	9
SDA (SS)	10
MOSI	11
MISO	12
SCK	13

Web links

[1] RFID starter kit: www.elektor.com/rfid-starter-kit-for-arduino-uno

[2] LiquidCrystal I2C library: https://github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library

[3] Library for the temperature/humidity sensor DHT11: http://arduino.cc/playground/Main/DHTLib

[4] Lib for the RFID-Module: https://github.com/miguelbalboa/rfid

[5] Project page and download pack: www.elektormagazine.com/160322

Listing 2. Display the RFID-UID using the Serial Monitor.

```
// RFID-RC522_data.ino

#include <SPI.h>
#include <MFRC522.h>

#define RST_PIN         9
#define SS_PIN          10

MFRC522 mfrc522(SS_PIN, RST_PIN);

void setup()
{  Serial.begin(9600);
   SPI.begin();    // Start SPI bus
   mfrc522.PCD_Init();    // Initialise MFC522 Reader
   mfrc522.PCD_DumpVersionToSerial();    // Show RC522 details
   Serial.println("Place RFID TAG in range!");
}

void loop()
{  // Card present?
   if ( ! mfrc522.PICC_IsNewCardPresent())
   { return; }

   // Select card
   if ( ! mfrc522.PICC_ReadCardSerial())
   { return; }

   // Send data to serial interface
   mfrc522.PICC_DumpToSerial(&(mfrc522.uid));
}
```

Designing loudspeaker boxes

Free programs for speaker box calculations

By Harry Baggen (Elektor-Labs)

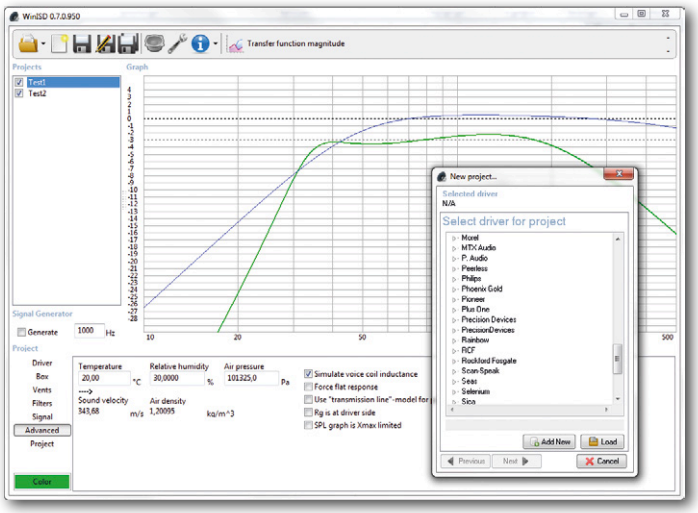
The predecessor to this column first appeared in the March 1996 issue with the name ‘electronics on-line’. In the following years many subjects were described, which these days are nearly all outdated, but a few have withstood the test of time quite well. In the December 1997 issue a few websites and programs for building your own loudspeakers were described. What has changed on this topic in the intervening 20 years?

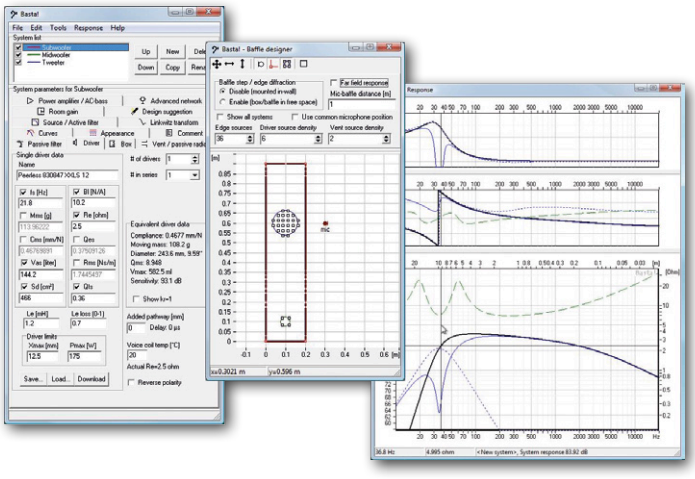
Since I was a passionate box designer and builder at the time, I have calculated many box designs with the help of the loudspeaker design programs available at the time. I thought it would be nice to see what free programs are available today (there are plenty of commercial programs, but as a hobbyist you much prefer to spend your money on loudspeakers than on software). I was initially very disappointed. A large number of programs that you will come across haven’t been updated in years. Often they can still be made to work in a DOS-box or with a few tricks, but there is very little to be found that has purposely been developed for modern computers with Windows 7, 8 or 10. There is nothing actually wrong with that as such, since the underlying calculations haven’t changed, but it is often quite a task to get these programs to function correctly.

One of the programs that just appeared in 1997, is WinISD [1] and is – 20 years later – still being developed, although at a very slow pace. The abbreviation ISD stands for ‘Interactive Speaker Designer’ and the program offers the options of calculating closed, bass-reflex and band-pass boxes. The program can also calculate and simulate various kinds of passive filters and equalizers. There is a database of drivers and most brands are present, but for more recent models you will nevertheless have to enter the necessary data yourself. This is an excellent program, which offers a lot of features and, in any case, runs very well on Windows 7.

Another interesting program, but now already 8 years old, is Basta! from Tolvan Data[2]. This was originally a commercial program, but the developer discontinued the development a few years ago and now makes it available as freeware. The program, according to them, also runs on Windows 8/10 and can calculate and simulate open baffles, closed boxes, bass-reflex and band-pass boxes. Basta! can also deal with ports and passive radiators. In addition there are various other features, such as the calculation of diffraction at box corners, calculation of air speeds from cones and in ports, maximum cone deflection and more.

For the measurements of loudspeakers the program Sirp [3] is also available on the same website, which with the aid of a logarithmic sine sweep measures the frequency response of a driver and from that also calculates the impulse response. Although this is not as effective as an MLS-based measurement for eliminating ambient influences, it is nevertheless a very useful method. Sirp can also measure the impedance curve of a system. The measured data can also be imported into Basta!

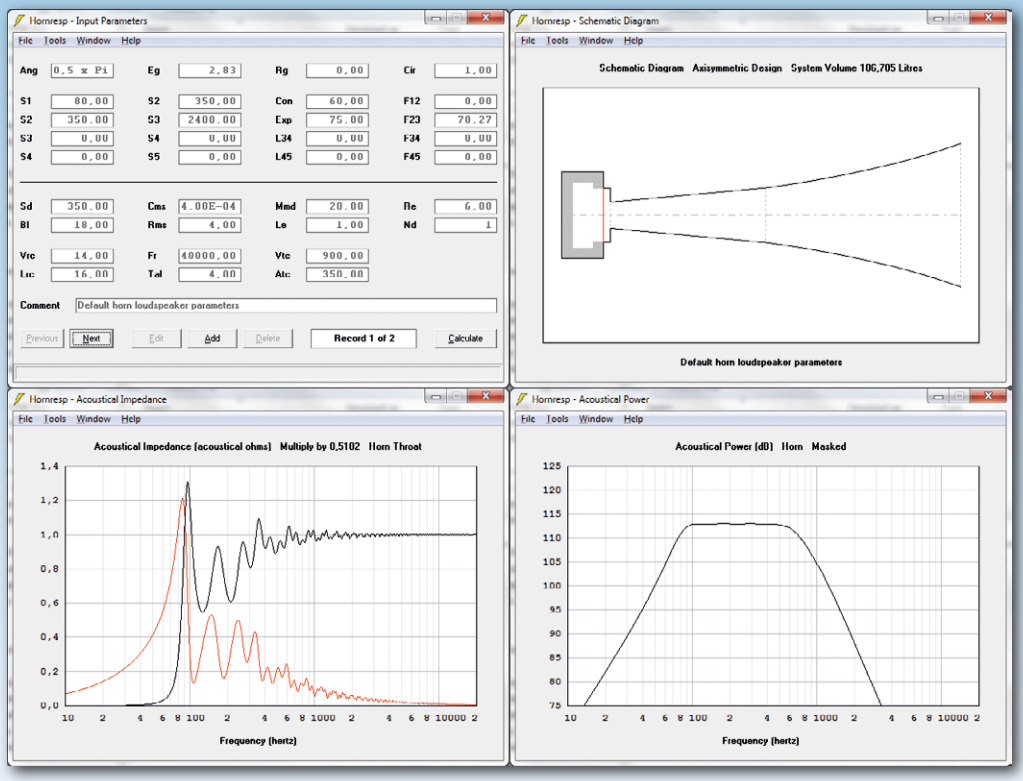




A relatively current program is VituixCAD [4] by Kimmo Saunisto, which offers many features and is suitable for Windows 7 through 10. VituixCAD is able to design and simulate multi-way loudspeaker systems. Various types of filters and boxes can be calculated, including systems with a passive radiator and three different two-chamber boxes. The program can also calculate radiation behavior and corner diffraction. Certainly worth the effort to try it out, although it will take some time to become familiar with all its features (this actually applies to all the programs described here).

Finally we take a look at Hornresp [5], one of the few free programs that is suitable for designing horn loudspeakers. The website where you can download the program consists of no more than a few lines of text. After downloading it would be a good idea if you visited the forum of Home Theater Shack [6], where a detailed explanation of the program can be found. Those who want to design there own horn loudspeakers are likely to thoroughly immerse themselves in the subject first. A sufficient basic knowledge about drivers and horns is certainly required for using this program, otherwise you will struggle to figure it out. In addition to horns, you can also design other box types such as closed, bass-reflex and band-pass boxes. The program has been in existence for a long time, but is still being actively maintained by author David J. McBean; the current version 40.20 runs without problems on Windows 7.

(160301)



Hornresp is an of the few free programs for the design of horn loudspeakers

Web Links

[1] www.linearteam.dk/

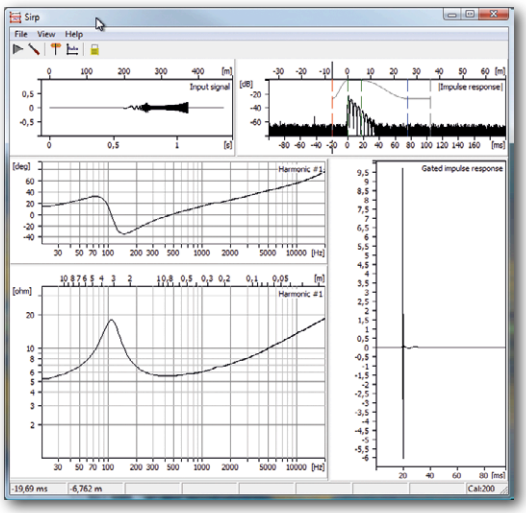
[2] www.tolvan.com/index.php?page=/basta/basta.php

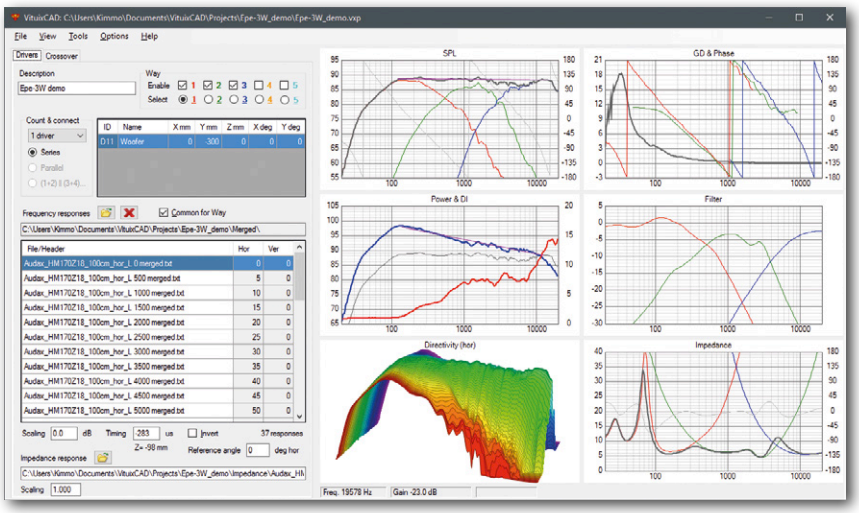
[3] www.tolvan.com/index.php?page=/sirp/sirp.php

[4] http://kimmosaunisto.net/Software/Software.html#VituixCAD

[5] www.hornresp.net/

[6] www.hometheatershack.com/forums/diy-subwoofers-general-discussion/36532-hornresp-dum-hmm-everyone.html



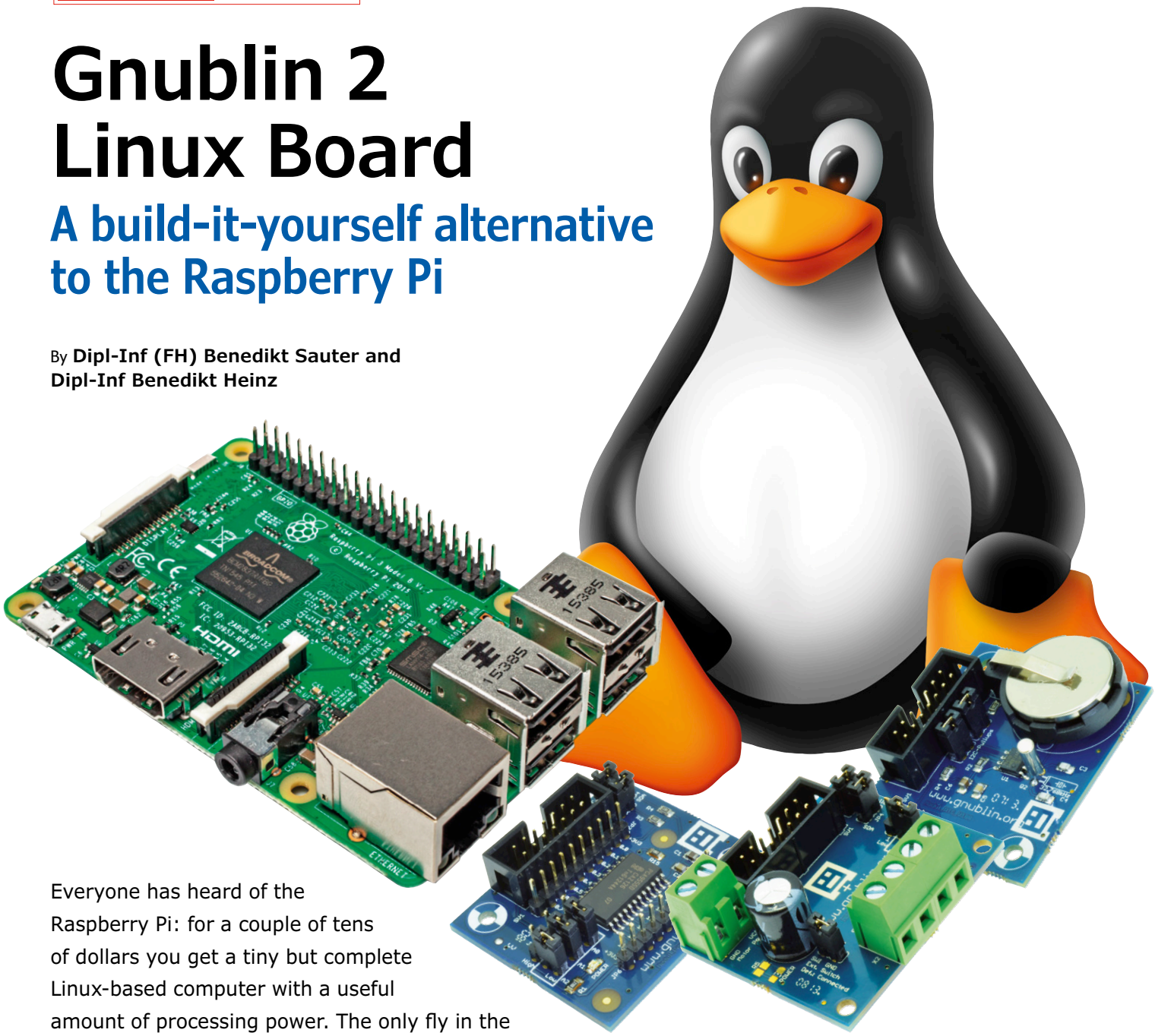


HOMELAB PROJECT

Gnublin 2 Linux Board

A build-it-yourself alternative to the Raspberry Pi

By **Dipl-Inf (FH) Benedikt Sauter and Dipl-Inf Benedikt Heinz**



Everyone has heard of the Raspberry Pi: for a couple of tens of dollars you get a tiny but complete Linux-based computer with a useful amount of processing power. The only fly in the ointment is that although it represents very good value, the hardware design is not completely open. The authors therefore decided to design their own single-board computer using a modern ARM Cortex-A5 CPU. Although our Linux board is not able to compete fully with the Raspberry Pi on price, it's completely open software and hardware can be modified to your heart's desire and customized for your educational, development or industrial application.

As the '2' in 'Gnublin 2' hints, the design has some history behind it. Its predecessor, the Gnublin 1, formed the basis for the Elektor Linux board and an embedded Linux course in Elektor [1]. A number of further articles followed, and a wide range of hardware is still available for purchase from the Elektor Store [2]. Our company 'embedded projects GmbH' also has its own web pages covering the design [3].

Five years have passed since that original design: practically an eternity in microelectronics terms. During that period we have seen not only the appearance of the Raspberry Pi and all its successors, but also an ever-increasing range of other new and improved Linux boards addressing a similar target market. So the question arises: does it make sense to produce an updated version of the Gnublin board? If the answer is going to be 'yes', then we will need to create

something more than a 'me-too' design. Now, the Gnublin 1 continues to be a popular board, and we feel that this can be put down to the following features where other designs fall short:

- Gnublin 1 is a completely open design. The circuit diagram, board layout and software are all made available;
- the components used are available for purchase in small quantities, and the printed circuit board, which has 'only' six layers, can also be made reasonably economically in small quantities;
- in contrast to most modern Linux boards which aim to offer as much computing power as possible, the Gnublin 1 has an exceptionally low power consumption. Many embedded systems applications have no need for flashy 3D graphics: for long-term operation low energy use is a more desirable feature.

And so we started to look to see if there was an ARM SoC (system-on-chip) available on the market that might form a suitable basis for version 2 of the Gnublin board. Our two main criteria were low power consumption and top-notch Linux support.

SoC choice

As low power consumption was important to us, our eyes were naturally drawn to single-core SoCs based around ARM Cortex-A5 and Cortex-A7 cores. We finally plumped for the Atmel ATSAMA5D41 [4], which uses a Cortex-A5 core. As far as Linux support is concerned, we had already had plenty of experience, as well as plenty of frustration, with the Gnublin 1 and other designs. The LPC3131 used in the Gnublin 1 was a particular culprit: the most recent version of the Linux kernel that can be used with this SoC is 2.6.33, which is more than six years old! Although the LPC3131 is still listed as 'active' and 'in production', it has not been supported in the Linux kernel for some time.

The situation is rather different with Atmel. Almost all Atmel SoCs, even those that are several years old, are still supported today. Special 'patches' from the manufacturer are not required, as the official Linux kernel provides built-in support for the devices. Also, Atmel provides extensive documentation without the bother of an NDA (non-disclosure agreement), and the company cooperates closely with the Linux community during development. Consequently the kernel development team maintains the code for these ICs when the kernel is modified. This forms a sound basis for long-term support for the SoC in up-to-date versions of the kernel.

Using the ATSAMA5D41 also opens up other possibilities lacking on other Linux boards. For example, the device has two serial ports available: one for the Linux console and one for application-specific use. Also, the Gnublin 2 can be connected directly to a PC using its high-speed USB device port, something that is not possible on a Raspberry Pi, for example. A common requirement in embedded systems is a real-time clock that can maintain the correct time even when power is lost, and the Gnublin 2 includes the option of adding a back-up battery for this purpose. And it is possible to connect a resistive touchscreen directly to the device, without the need for an external power supply.

All these advantages added up to enough reason to go ahead with a Gnublin version 2 based on this device. Like its predecessor, Gnublin 2 is designed as a basis for your own designs and products: even test reports from the EMC laboratory will be included along with the usual documentation.

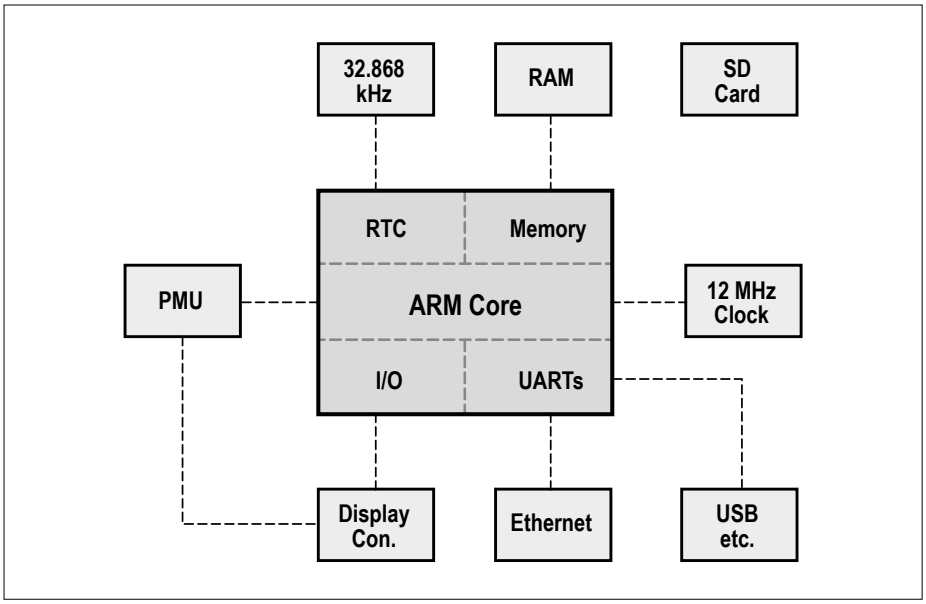


Figure 1. The block diagram of the Gnublin 2 shows the SoC and its submodules outlined in bold, along with the hardware connected to it.

Potential applications

Gnublin 2 is at home in any application involving measurement and control, or indeed in any application or product where it is desired to replace an old-school microcontroller with a more powerful CPU capable of running Linux. The cost of the basic components, including the processor, memory and power supply, comes to only about 15 to 20 dollars, depending on quantity. Gnublin 2 is thus also suitable for short-run production.

The freely-available parts list, circuit diagram and printed circuit board layouts, as well as EMC test results and calculation tools remove a lot of the difficulties usually associated with short-run manufacture of products based on modified versions of 'ready-made' boards such as the Raspberry Pi and its variants.

The documentation also makes connecting up additional hardware a piece of cake. In the usual prototype development process using hand-assembled samples, the first trip to the EMC lab is generally only made when mass production is about to start. Having the results of EMC testing already available saves valuable time at this point. And even for one-off applications there is comfort in knowing where one stands regarding RF characteristics.

Organization and components

Happily, with the Linux system in its

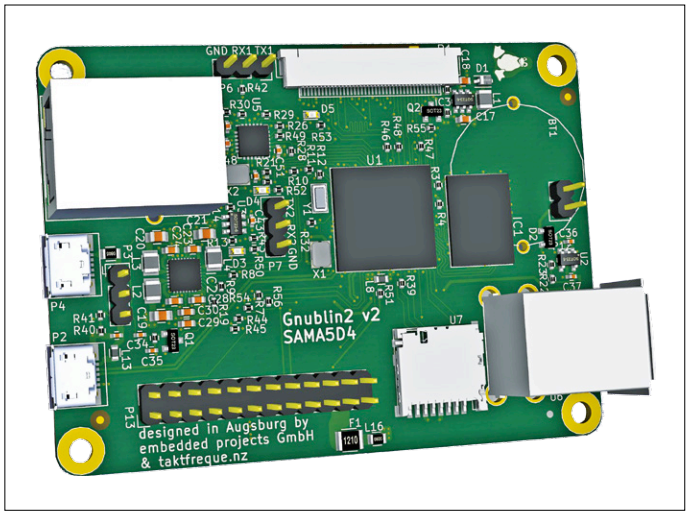


Figure 2. 3D rendering of the Gnublin 2 printed circuit board.

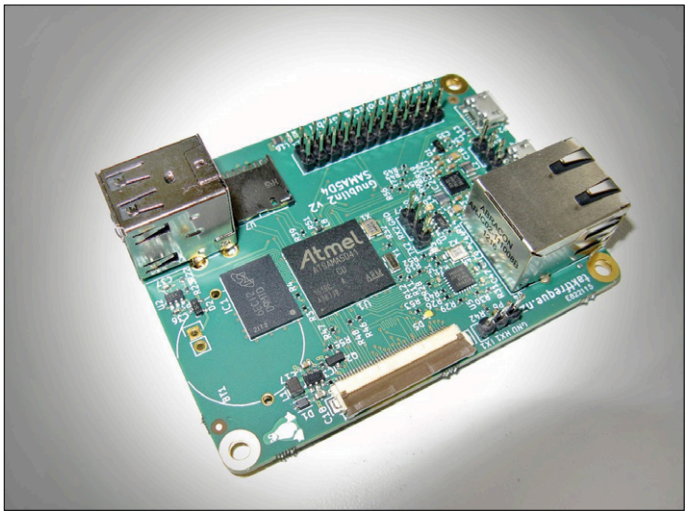


Figure 3. The assembled Gnublin 2 prototype.

idle state, the Gnublin 2 (including its 256 MB of RAM) dissipates less than 0.3 W at a clock rate of 600 MHz. In the block diagram shown in **Figure 1** the SoC is outlined in bold. As you can see, besides the processor core itself, the SoC includes a number of important modules such as the RTC (real-time clock) and interfaces to external memory hardware. A detailed circuit diagram can be downloaded at [6]: it is too large to include here.

The 256 MB memory device IC1 is connected to the ATSAMA5D41 SoC (U1) over a 16-bit bus. Compared to other boards, which use a bus 32 bits wide, this results in a simpler and cheaper printed circuit board and in lower power consumption. Even so, with a RAM clock of 300 MHz the DDR2 interface achieves a total data rate of over 1 Gbit/s.

The SoC needs various supply voltages for its I/O pins and for the processor core. These voltages are generated by IC2, an ACT8865 PMU (power management unit) [5]. The PMU includes several switching and linear regulators and can be configured from the SoC over its I^2C bus.

The supply for the real-time clock integrated into the SoC comes from a type TPS78001 low-dropout and ultra-low-power linear regulator (U2), which in turn can be supplied from a button cell or externally over a connector. The real-time clock of course uses a standard 32.768 kHz watch crystal (Y1). The main system clock

Surfing the Internet with a display and a browser

There is a wide variety of Linux distributions that can be used with our Cortex-A5 board. Our personal preference for development purposes is Debian Jessie [1], but Yocto and Ubuntu are also both possible. The base installation of Debian is considerably more lightweight than Ubuntu, but nevertheless offers a huge choice of software packages that can easily be downloaded over the network and then installed using the 'apt' tool. Installed packages can also be updated automatically over the network, which is very useful and desirable, especially in the case of security-related updates. Even the GNU C compiler can be installed using 'apt'.

Software can be compiled directly on the board itself, as long as the code is not too complex. A graphical user interface is available in the form of LXDE [2]: this is very lightweight and can be installed using the command 'apt-get install lxde'. The available 256 MB of RAM may not be sufficient these days to display the most resource-intensive web pages; the Midori browser [3] is a lightweight option that can handle web sites that use HTML5, CSS and Javascript. The command 'apt-get install midori' can be used to install this browser.



The illustration shows that Gnublin 2 has no problems loading and displaying normal web pages.

[1] Debian Jessie: https://www.debian.org/releases/jessie/

[2] LXDE: https://wiki.lxde.org/en/Debian

[3] Midori: http://midori-browser.org/

is derived from 12 MHz crystal X1.
In contrast to the Gnublin 1 the updated version can drive a TFT display module directly, which opens the door to many new applications. So that the whole system, including the display, can be powered from a single 5 V supply, we have included an extra step-up regulator based on an AP5724 (IC3). This can supply the display's LED backlight with a constant current of 40 mA at around 24 V. The ADC built into the SoC also allows the direct connection of a resistive touch panel.
Ethernet PHY KSZ8081RNA/D (U5) allows the board to be connected directly to a network at 10 Mbit/s or 100 Mbit/s. The necessary magnetics are integrated into RJ45 connector U4, and U3, a type 24AA025E48 I²C EEPROM, is used to provide a valid MAC address.
As is the case for practically every other Linux board, the Gnublin 2 has built-in mass storage: at system start-up the boot loader and the Linux system are loaded from microSD card U7.
The SoC includes a total of three high-speed USB ports. Of these, two are made available as host ports on dual connector U6, while the third port is brought out to micro USB socket P2 as a device port. This allows the board to act as almost any type of USB device when connected to a host computer, and for example it is possible to set up a network connection over USB between the host and the Gnublin 2. It is possible to power the board directly over socket P2, and a second micro USB socket (P4) is also provided as an alternative dedicated connection for a higher-current supply such as a smartphone charger. The latter is handy when several USB devices are to be connected to the Gnublin 2. Jumper P3 selects whether the board is powered over P2 or P4.
The SoC's I/O pins operate at 3.3 V logic levels, and many of the pins are brought out on header P13. As on the Gnublin 1, this connection includes an SPI bus with a number of individual chip select outputs. An I²C bus is also available on P13, along with connections for the SoC's second UART. This serial interface is also accessible on P7: it is not used as a console by the Linux system and so can be freely employed for whatever application-specific purposes you might have. Finally, P6 provides access to the first UART, which is used by the Linux system as a console. Kernel messages are output on this port at start-up, and it is also possible to use it to control the boot loader.
Figure 2 shows a 3D rendering of the printed circuit board, designed using KiCAD. The layout files as well as the circuit diagram, software, documentation and other information are available for free download at [6]. **Figure 3** shows the assembled board. The **Text Box** contains some basic information on how to put together the software required for a complete Linux computer based on a Gnublin 2 board and a display.

Crowdfunding!

Elektor and embedded projects GmbH have launched a crowdfunding campaign for this project. If you would like to take part or order a Gnublin 2 board, you can find more information at **www.gnublin.org**



Outlook
This article has given just a brief overview of the Gnublin 2 system. With the help of the resources at [6] you will find it easy to start building your own projects. During the course of development of the Gnublin 2 several interesting topics arose that are worthy of articles in their own right. For example, we are planning a detailed description of how we went about laying out the printed circuit board using KiCAD: we hope that using the Gnublin 2 board as a case study will be considerably more realistic than the artificially simplified circuits normally used in tutorials. Also, we plan an article on RF interference and EMC testing based on our experience from this design. And, last but not least, we are planning an article that describes step by step how you can go about making your own variant of our Linux board. ⏮

(160090)

About the authors

Benedikt Sauter studied Computer Science; after his studies he worked on a range of microcontroller projects and began to work independently. Shortly afterwards the company embedded products GmbH was founded. He has had a passion for software and hardware from a young age, and has had several open-source projects published in Elektor.
Benedikt Heinz has been developing software and digital electronics for microcontrollers and CPUs for over 15 years. A long-time desktop Linux user, he has a strong preference for open-source software, and contributes to open-source projects. He is currently working on software to simplify the layout of complex boards (using PCIe, DDR DRAM and so on) using KiCAD.

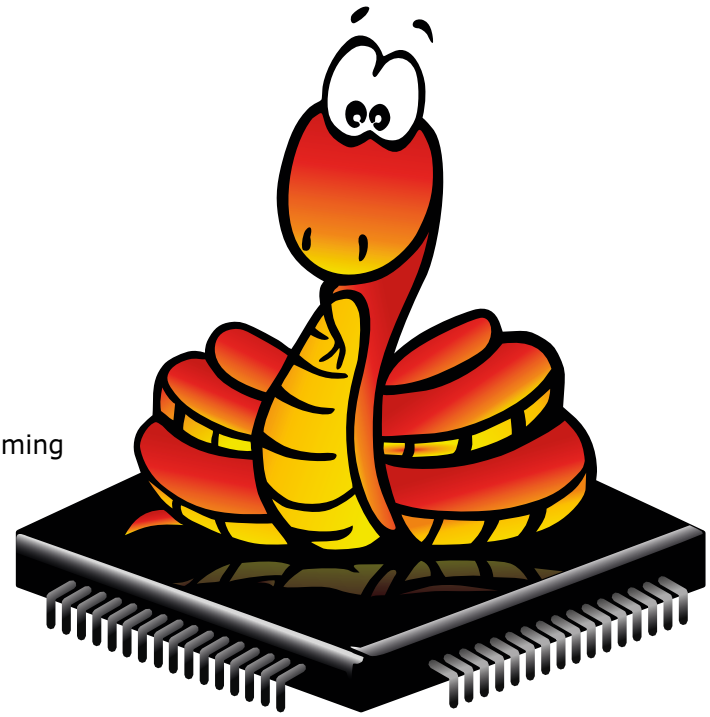
Web Links

[1] Embedded Linux made easy: www.elektormagazine.com/120026
[2] Gnublin products: www.elektor.com/development/gnublin/
[3] Gnublin website: http://gnublin.embedded-projects.net/
[4] ATSAMA5D41: www.atmel.com/devices/ATSAMA5D41.aspx
[5] ACT8865: www.active-semi.com/products/power-management-units/act88xx
[6] Gnublin 2 resources: www.gnublin.org/index2.html

MicroPython

Python for small systems

By **Dogan Ibrahim** (UK)



MicroPython is a highly efficient and powerful programming language derived from Python, and having inherited a small collection of libraries. MicroPython fits in a mere 256 KB of code space and 16 KB of RAM, permitting it to be used on microcontrollers and other embedded systems with limited resources.

Python is used in many universities and technical colleges around the world as the initial programming language for students. The large collection of powerful libraries and the ease of use of Python make it an ideal language for everyone new to programming. MicroPython, originally targeted at 32-bit ARM microcontrollers, is compatible with Python and, thanks to its small size, makes an excellent choice for use on embedded processors. With MicroPython complex and manageable code can be developed to control embedded systems that would otherwise require languages such as C or C++. MicroPython allows experienced as well as novice Python users to program small embedded systems.

The main reasons for MicroPython to not fully support Python 3 are the lack of sufficient memory and missing hardware and software features (such as multitasking and multiprocessing) of embedded processors. The differences between Python 3 and MicroPython can be found at [1].

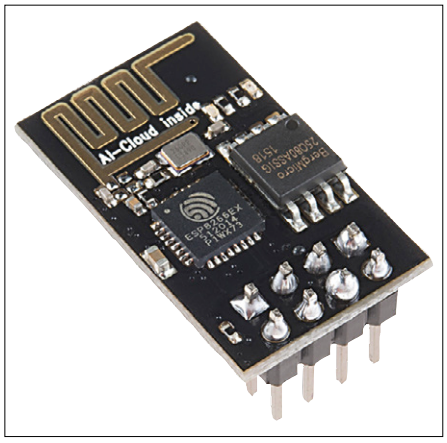


Figure 1. The ESP01, a highly practical and popular ESP8266-based module.



Figure 2. The BBC micro:bit can be programed online in MicroPython.

Why MicroPython?

MicroPython and Python offer some unique features compared to other programming languages. In a nutshell:

- (Micro)Python is interactive meaning the program is not compiled and uploaded into the target processor, but rather interpreted and acted upon at runtime. Although this makes programs run somewhat slower, it has the advantage of the user being able to easily experiment with his/her code. For example, we can simply do interactive calculations as if we are using a calculator, or experiment with parts of the program until we get the desired results. This feature, also known as Read-Evaluate-Print-Loop or REPL, is not available in compiled languages such as C or C++.
- In addition to the large number of built-in functions, an extensive set of function libraries is available that can be included in (or imported into) a program. For example, there are libraries for random number generation, trigonometry, making music, networking, string & file processing, graphics & gaming, and much more.
- MicroPython can be mixed with other programming languages such as C or C++. This gives additional power and flexibility since parts of the code that require high speed can be developed using languages better suited for such tasks.
- Exceptions and error handling are supported, which is especially important in real-time programming. Without proper error handling a crashed program may stop the processor in an unknown state which can have highly undesirable effects.
- It is open source meaning that the latest release can be downloaded [2] and run at no cost. The source code of MicroPython can be modified and ported to specific processors of interest.
- Finally, the language is human-readable and its syntax is easy to learn and understand.

What MicroPython can and can't do

MicroPython can, in general, do every-

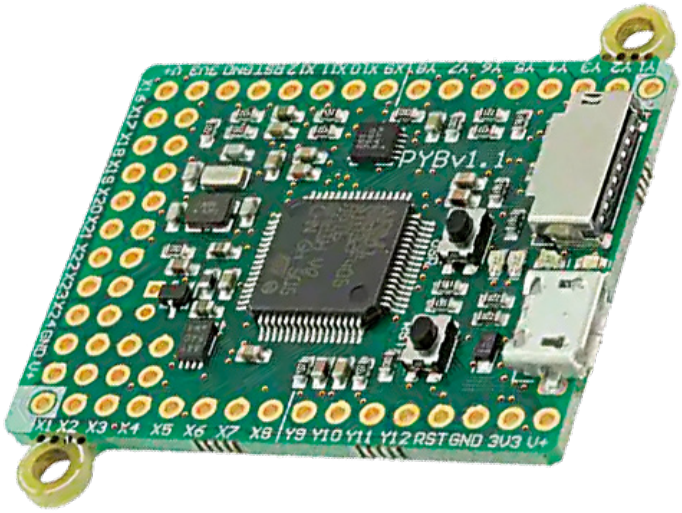


Figure 3. So you know, the pyboard is the official MicroPython microcontroller board.

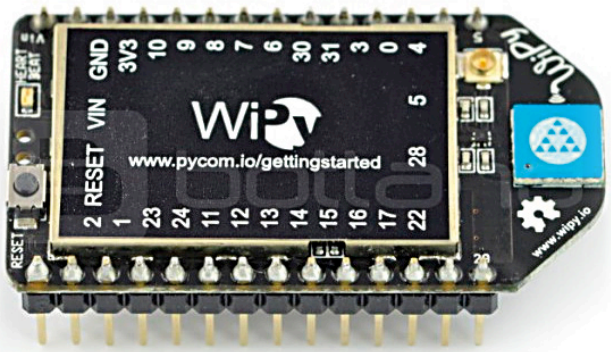


Figure 4. The WiPy is similar to the ESP8266-based NodeMCU board, but with a CC3200 ARM Cortex M4 instead.

thing other programming languages can do too, like controlling hardware devices such as LEDs & displays, switches & buttons, sensors, motors and so on. Communication busses such as RS-232, CAN, I²C, SPI, and others can easily be used thanks to built-in and external libraries. Network and Wi-Fi-based programs can be written to communicate with other devices on a network, or to develop IoT systems.

Because MicroPython is an interpreted language it is slower compared to other embedded programming languages, consequently it is not a good choice for fast digital signal processing or real-time applications where high execution speeds are critical. In addition, although less important nowadays, MicroPython uses slightly more memory than most other embedded languages. Since MicroPython is a subset of Python that does not support all the Python libraries, a program developed in Python may not work on an embedded system running MicroPython.

Boards supported by MicroPython

The number of development boards supported by MicroPython is increasing along with its popularity. Let's list a few.

ESP8266

Boards fitted with this popular Wi-Fi-capable MCU (**Figure 1**) with its built-in TCP/IP stack and USB interface can be programed in MicroPython. The MCU is based on a 32-bit RISC CPU and MicroPython offers support for GPIO, SPI, I²C, UART, ADC and I²S. This may well be one of the cheapest ways to start out with if you wish to experiment with MicroPython.

BBC micro:bit

This credit card sized board (**Figure 2**) with its many built-in features such as 25 LEDs, two pushbuttons, an accelerometer, a compass, GPIO, I²C, UART, and ADC is supported by MicroPython. Programs can be created online which removes the need to do any setup or configuration. The BBC micro:bit is highly recommended for people new to programming.

pyboard

Based on the STM32F series Cortex M4 processor this development board (**Figure 3**) comes preloaded with MicroPython. The pyboard can be connected to a PC through its USB port. The board features a real-time clock, an accelerometer, GPIO, an ADC, four LEDs, and a microSD card slot.

WiPy

Like the pyboard this board (**Figure 4**) too has built-in MicroPython support. Its pins are suitable for plugging it on a breadboard. The board is based on the CC3200 Cortex M4 processor running at 80 MHz and includes UART, SPI, I²S, ADCs, Wi-Fi, GPIO, timers, and hash and encryption engines.

Some other embedded development boards of interest that come with MicroPython support include: Teensy 3.x, SAMD21, LoPy, STM32F4-Discovery, Raspberry Pi and BeagleBone (both run the full Python 3 code).

MicroPython program example

Here is a simple MicroPython program that runs on the BBC micro:bit. It's a thermostat where the CPU temperature is read continuously and an appropriate message is displayed on the on-board LED matrix. When the temperature is equal to or higher than 25 ºC the message 'HIGH' is displayed. If the temperature is between 20 ºC and 25 ºC 'MEDIUM' is shown. Otherwise, 'LOW' is displayed.

```
#Simple CPU thermostat program
from microbit import *

while True:
  temp = temperature()
  If temp >= 25:
    display.scroll("HIGH")
  elif temp >= 20 and temp < 25:
    display.scroll("MEDIUM")
  else:
    display.scroll("LOW")
```

(160315)

Web Links

[1] https://github.com/micropython/micropython/wiki/Differences

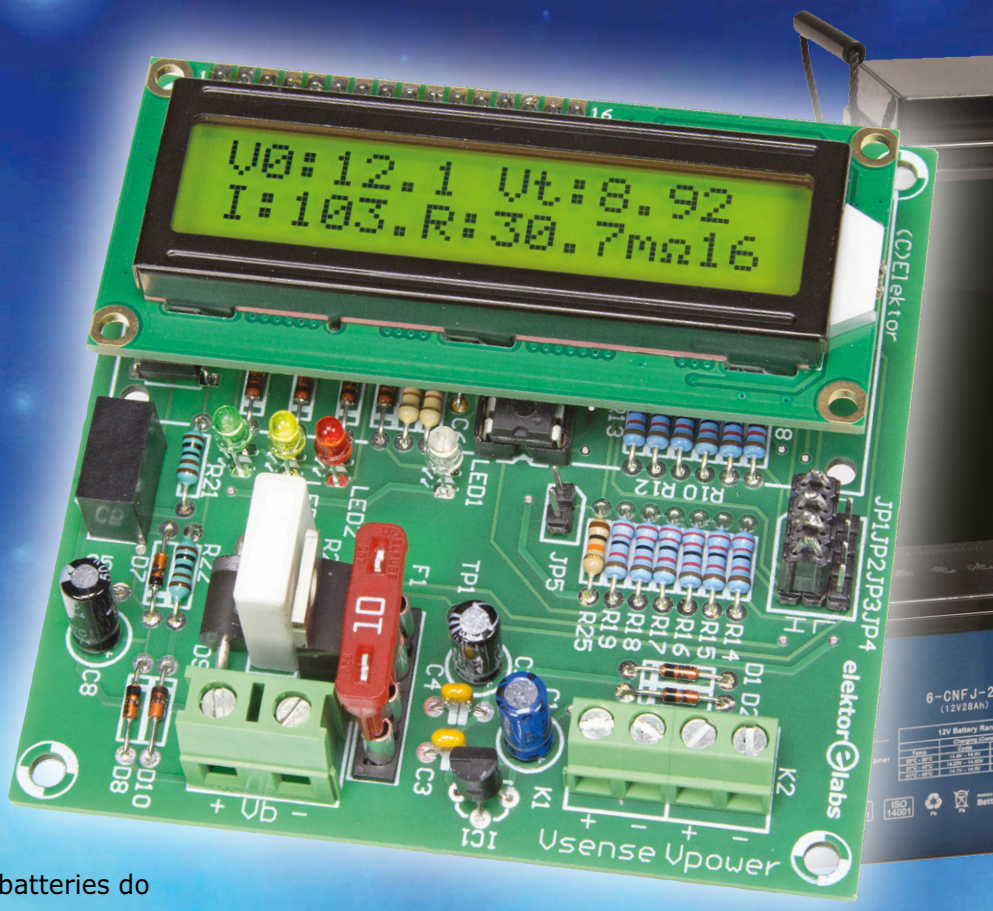
[2] https://github.com/micropython/

[3] http://micropython.org/

e LABS PROJECT

Lead-acid Battery Activator 0-30 V

Also shows the battery quality



A well-known electronics store has been selling a very simple lead-acid battery activator for many years [1]. Although I can't prove it, my lead-acid batteries do seem to last longer since I started using the activator. The principle behind this circuit is very simple. The battery is loaded with a current of about 100 A for a period of 100 µs, which is repeated every 30 seconds. But this circuit is capable of more, which we'll show you in this article.

By **Jan Lichtenbelt** (The Netherlands)

PROJECT INFO

Test & Measurement

Hobby | Modelling | Batteries

entry level
➡ intermediate level
expert level

3 hours approx.

No special tools are required

€60 / $65 / £50 approx.

The theory is that short (large) current pulses will revert the sulfating of the lead plates [2]. I have used the Conrad Electronics circuit in parallel with a simple power supply, which is based on an LM317, set to 13.8 V, and used with a 7-Ah lead-acid battery. This power supply didn't seem to have any problems with these large current pulses.

Not only is sulfating prevented, but these types of peak current can also be used to determine the quality of batteries. We do this by letting a large current (I) flow for a very short period through an external shunt while measuring the reduction in the terminal voltage (V_t). The current flow is given by:

$$I = \Delta V_{shunt} / R_{shunt}$$

The internal resistance is:

$$R_i = (V_0 - V_t) / I$$

Where V_0 is the no load terminal voltage and V_t is the terminal voltage under load.

This internal resistance gives an indication as to the quality of the battery: the lower it is, the better. The circuit shown here has an extended functionality. The design is suitable for use with 2, 6, 12 and 24V lead-acid batteries and works as both an activator as well as for the measurement of the internal resistance of all primary cells and batteries up to 30 V.

At first we thought we'd enhance the Conrad design, but that resulted in a ramshackle circuit that wasn't really suitable for use with such high currents. The disadvantage of the Conrad circuit is that the 0.1 Ω / 2 W shunt wasn't designed for use with these current pulses. In practice it could fail without you noticing.

It is shown in the literature [3] that the internal resistance depends mostly on the temperature, the State of Charge (SOC) and the age of the battery. The resistance increases when the temperature drops. It also increases when the battery is heavily discharged. Older batteries can also cause the internal resistance to increase. When you want to compare internal resis-



tance values, you should always measure them as much as possible under the same conditions. In other words, you should only vary one of the three previously mentioned parameters. Note that a relatively high temperature at room temperature is an indication that the battery is probably old or badly charged. In any case, you will obtain more detailed information on the state of your battery if you have its internal resistance as well as its terminal voltage.

2, 6, 12 and 24-V lead-acid battery activator

From the circuit in **Figure 1** we can see that the lead-acid battery activator uses a microprocessor to output a 100-µs long pulse to a MOSFET every 30 s. A shunt is connected to the battery in series with this MOSFET, which is protected against reverse polarity connections by a diode. The theoretical peak current when a shunt of 50 mΩ is used is about 100 A for a 6-V lead-acid battery.

The voltage measurement at the battery terminals is done with a separate set of wires, so that the large current flow doesn't distort the reading (four point measurement). Both circuits are separated from each other, and they're only connected together at the terminals using battery clips. As mentioned earlier, the current is calculated by measuring the voltage across the shunt.

The microprocessor takes care of the pulse generation. This is made visible via a blue led (LED1). This pulse needs to be fed to the shunt circuit, which happens via an optocoupler. The function of the LEDs for 6, 12 and 24-V lead-acid batteries is explained in **Table 1**.

This activator circuit is also perfectly suitable for use with 2V lead-acid batteries. One disadvantage in this case is that an external supply is required for the generation of the pulses. However, this could be implemented with a 2-5 volt step-up converter. The red, orange and green leds unfortunately won't be able to show the status of 2-V batteries.

Table 1. The three LEDs give a quick overview of the state of charge of lead-acid batteries.

Measurement range	0-10 V			0-20 V	0-30 V
Voltage	V < 2.5	2.5 < V < 5	5 < V < 10		
LED	2-V battery	4-V battery	6-V battery	12-V battery	24-V battery
Red	< 1.98 V	2.5 - 4.00 V	5 - 5.95 V	< 11.9 V	< 23.8V
Orange	1.98 - 2.08 V	4.00 - 4.16 V	5.95 - 6.25 V	11.9 - 12.5 V	23.8 - 25.0 V
Green	2.08 - 2.5 V	4.16 - 5 V	6.25 - 10 V	> 12.5 V	> 25.0 V

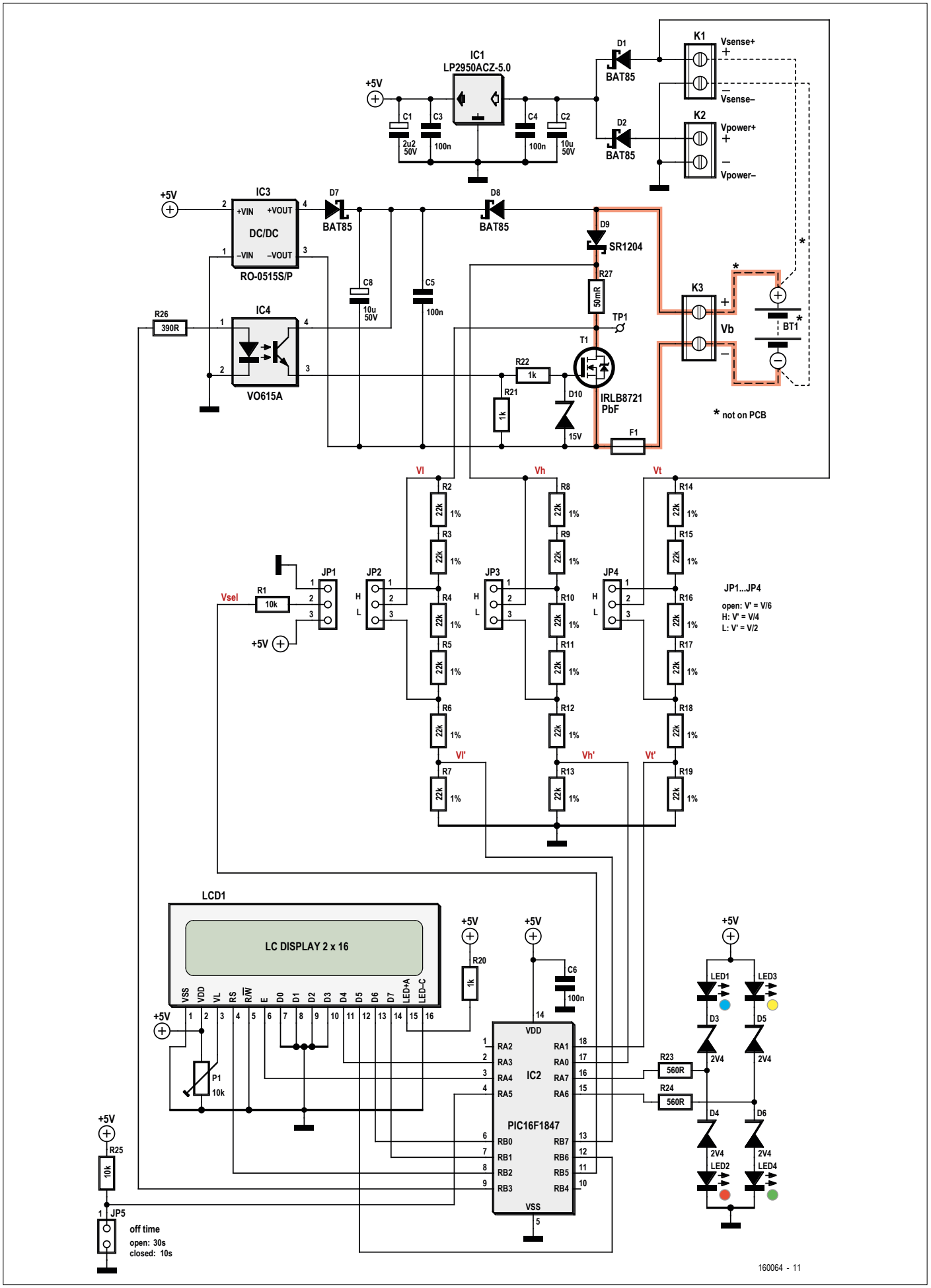

IC1
LP2950ACZ-5.0
+5V
C1
2u2
50V
C3
100n
C4
100n
C2
10u
50V
D1
BAT85
D2
BAT85
K1
Vsense+
Vsense–
K2
Vpower+
Vpower–
IC3
DC/DC
RO-0515S/P
+VIN
+VOUT
–VIN
–VOUT
D7
BAT85
D8
BAT85
C8
10u
50V
C5
100n
D9
SR1204
R27
50mR
TP1
T1
IRLB8721
PbF
K3
Vb
BT1
* not on PCB
R26
390R
IC4
VO615A
R22
1k
R21
1k
D10
15V
F1
VI
Vh
Vt
R1
10k
Vsel
JP1
JP2
JP3
JP4
H
L
JP1...JP4
open: V' = V/6
H: V' = V/4
L: V' = V/2
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
R15
R16
R17
R18
R19
22k
1%
VI'
Vh'
Vt'
LCD1
LC DISPLAY 2 x 16
VSS
VDD
VL
RS
R/W
E
D0
D1
D2
D3
D4
D5
D6
D7
LED+A
LED-C
R20
1k
C6
100n
P1
10k
IC2
PIC16F1847
VDD
VSS
RA0
RA1
RA2
RA3
RA4
RA5
RA6
RA7
RB0
RB1
RB2
RB3
RB4
RB5
RB6
RB7
R23
560R
R24
560R
LED1
LED2
LED3
LED4
D3
D4
D5
D6
2V4
R25
10k
JP5
off time
open: 30s
closed: 10s
160064 - 11

Table 2. Overview of warnings and when the program stops.			
	LED	**Flash rate**	**Action**
Possible open circuit	Green	Slow (1 s)	Wait until this is corrected
No terminal voltage*	Green	Fast (400 ms)	Wait until this is corrected
Terminal voltage above measurement range	Orange	Fast (400 ms)	The program stops. *Disconnect the battery from the circuit as quickly as possible!*
Current too large	Red	Fast (400 ms)	The program stops. Add an extra external series resistor (R_{ext}) and try again.
*This can only happen when an external power source is used to supply the microcontroller, with battery voltages less than 5.5 V.			

Battery quality measurements 0.2-30 V

From here on, when we talk about batteries this implies both lead-acid batteries as well as primary cells.

If a single shunt of 50 mΩ is used, the theoretical value of the current through the shunt varies from 40 A to some 600 A. In practice the total resistance of the shunt circuit will be higher due to the connecting cables, contacts, the internal resistance of the battery and the voltage drop across Schottky diode D9. The resulting currents will therefore be lower than the theoretical ones. Ideally, the current pulse would have a value of 10 A for the range from 0.2-10 volt and 100 A for anything above that.

When an extra resistor is connected externally in series with the discharge circuit, the maximum current will be restricted at higher voltages. Alternatively, you may decide to use a shunt for one specific range only. For ranges of about 0.2-10 V, 5.5-20 V or 5.5-30 V you should use a shunt of 50, 100 or 220 mΩ respectively. As mentioned earlier, you must always use an external supply when the voltage is less than 5.5 V.

The input voltage at the ADC may have a maximum value of 5 V. A potential divider is therefore required for the above-mentioned ranges, with values of 1:2, 1:4 or 1:6 respectively. This can be selected using a set of jumpers in the circuit.

Under certain conditions the program stops, see **Table 2**. The reason is shown on the LCD. However, since the LCD may not be in use, the error condition is also shown using the leds. The most severe error condition is when the current is too high. When this occurs, the red led lights up and the shunt won't be connected again. The circuit will have to be restarted in order to get out of this error condition.

Negative values may be possible when the battery isn't connected properly. The values read will then depend on the charges that happen to be present at the time.

When the circuit starts, the red, orange and green leds flash alternately. When you want to test a number of batteries in quick succession, it would be useful if the pulses would repeat more quickly. A jumper on the board is used to set the period to either 10 s or 30 s. For lead-acid batteries you must always select 30 s.

Figure 1. In the circuit the different sections can be clearly seen from top to bottom: supply, MOSFET circuit, potential divider with jumpers, and control and display via the microprocessor.

Safety measures

In a circuit with currents of 100 A and above, good safety measures are paramount. Of concern here is the shunt circuit. Three safety measures have been put into place (two circuit improvements and one preventative measure).

1. The connections in the shunt circuit have to be able to cope with the short, large pulses. All the copper tracks in the battery shunt circuit on the PCB must therefore be tinned with a thick layer of extra solder. In addition, the long leads of the diode, shunt and MOSFET have to be used to connect them together. The space left between the leads and the copper track should be filled with solder.
2. Cables should be kept as short as possible. Under no circumstances should they be rolled up, which would create inductive loads, along with very high peak voltages. The program stops when the load is too high, but only after the current has been measured. The damage will have been done by then.
3. If for any reason the MOSFET conducts too long or breaks down, the fuse (F1) will blow. For 1.2V batteries we would recommend a 3 A fast blow fuse. For other primary batteries and lead-acid batteries with higher voltages you should use a fuse with a maximum rating of 10 A fast blow.

Electrical circuit

The section with large currents consists of the shunt circuit with D9 as protection diode, low-inductance shunt resistor R27 and finally MOSFET T1. A current of 100 A can be switched using a gate voltage (V_{gs}) of 5.5 V. For a current of 220 A, V_{gs} should be 9-10 V. The switching times given are in the order of 10 ns, but will be longer in practice because of the use of a resistor at the gate input.

The MOSFET was selected because of its low total gate charge (Q_g) of 7.6 nC at 4.5 V. This results in a corresponding capacitance of:

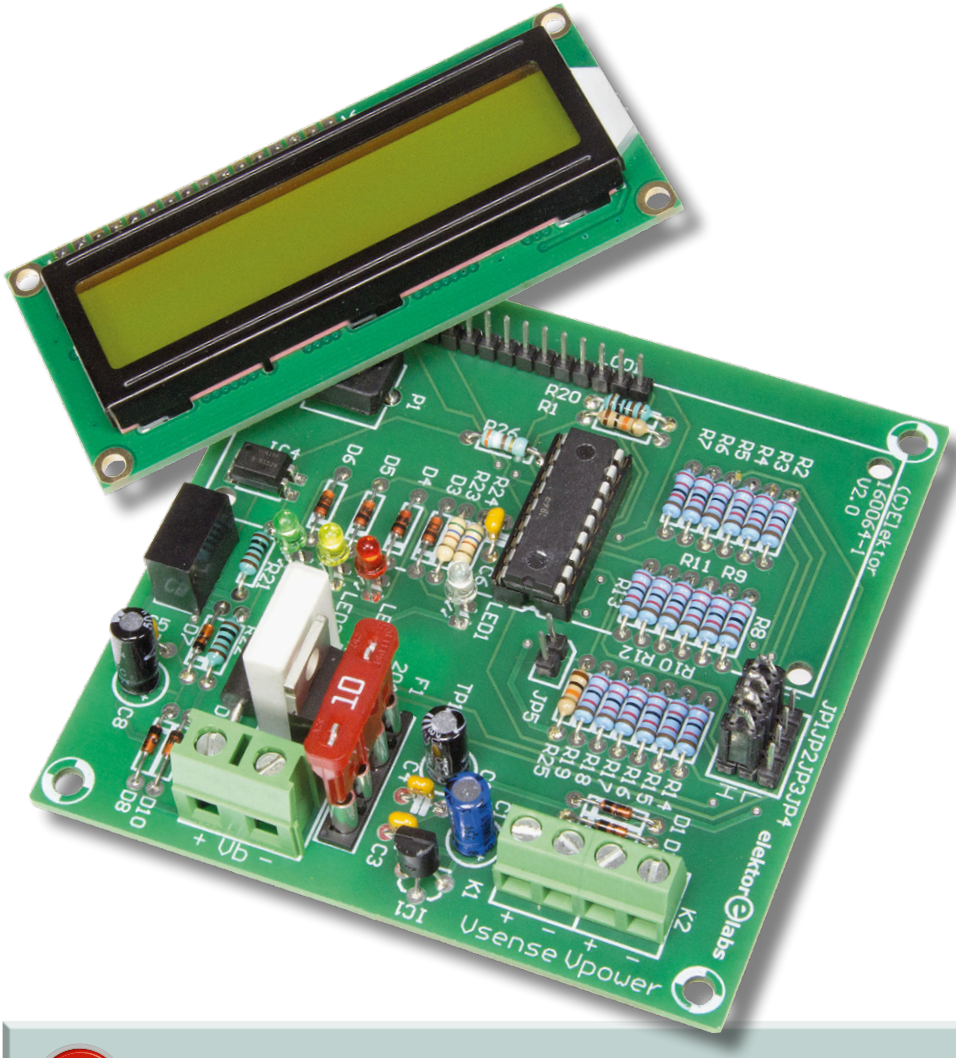
$C = Q / V = 7.6 / 4.5 \approx 2$ nF.

With a 1-kΩ resistor (R22), the RC time is:

$T_{on} = 10^3 \times 2 \times 10^{-9} = 2$ µs.

Adding R21, T_{off} becomes 4 µs. These are very good switching times for a 100µs pulse.

The MOSFET is brought into conduction by raising the gate voltage to a sufficient level in a very short time. Since this gate circuit has to be isolated from the microcontroller circuit, an optocoupler seems the most appropriate to drive the gate. There are commercial opto gate-drivers available that can even supply the required energy to drive the gate. We tried many, but none of them was able to produce a good pulse of 100 A for 100 µs with short switching times. It was often the case that either the turn-on time or the turn-off time was good, but never both. The supply required for the gate voltage (V_{gs}) has to be at least 8 V in order to make currents of 100 A to 220 A flow. When the battery voltage is less than this, the voltage has to be generated some-



where else. For this reason, a separate DC/DC converter (IC3, an RO-0515S/P) has been used to supply the gate voltage when the battery voltage is less than 8 V. Above this, the supply can come from the battery itself, via D8. To obtain a fast switching time, we've used a 5-15 V DC/DC-converter, which is more than enough to satisfy the minimum gate voltage (V_{gs}) requirement of 8 V. However, V_{gs} must not be greater than 18 V, so a 15-V zener diode (D10) and a 1-kΩ resistor (R22) have been added to the circuit to prevent this. Resistor R21 ensures that the gate voltage returns to 0 V quickly at the end of the pulse.

To enable the measurement of 6V lead-acid batteries without an external power supply, we've used a low-drop voltage regulator, IC1 (LP2950C), which requires just 5.3 V at the current of 30 mA used by the circuit. For measurements of voltages less than 5.5 V (increased by 0.2 V for the Schottky diode voltage drop) an external power supply has to be used, which is connected to a separate con-

PARTS LIST

Resistors
(0.25 W, 250 V, unless otherwise stated)

R1,R25 = 10 kΩ
R2...R19 = 22 kΩ, 1%, 0,6 W, 350 V
R20...R22 = 1 kΩ
R23,R24 = 560 Ω
R27 = 50 mΩ, 1 W, MPC75
R26 = 390 Ω
P1 = 10 kΩ, preset

Capacitors:

C1 = 2.2 µF, 50 V, 2 mm pitch, 5x11 mm
C2,C8 = 10 µF, 50 V, 2 mm pitch, 5x11 mm
C3...C6 = 100 nF, 50 V, X7R, 0.2" pitch

Semiconductors

D1,D2,D7,D8 = BAT85
D3...D6 = BZX79-C2V4
D9 = SR1204
D10 = BZX55C15V
LED1 = LED, blue, 3 mm
LED2 = LED, red, 3 mm
LED3 = LED, yellow, 3 mm
LED4 = LED, green, 3 mm
T1 = IRLB8721PBF
IC1 = LP2950ACZ-5.0
IC2 = PIC16F1847-I/P, programmed (store # 160064-41)
IC3 = RO-0515S/P
IC4 = VO615A

Miscellaneous

F1 = PCB-mount fuseholder + fuse (see text)
LCD1 = LCD, 2x16 characters, EPS 120061-74
LCD1 = 16-pin pinheader, vertical, 0.1" pitch
LCD1 = 16-pin pinheader bus, vertical
K1,K2 = 2-way PCB screw terminal, 0.2" pitch
K3 = 2-way PCB screw terminal, 0.3" pitch
JP1...JP4 = 3-way header, vertical, 0.1" pitch
JP5 = 2-way header, vertical, 0.1" pitch
Jumper for JP5
2x5 header bus, vertical for JP1-JP5
IC socket, DIP-18, for IC2
PCB 160064-1 v2.1





nector (K2) in parallel with the connector for the terminal voltage measurement (K1). Note that these connectors aren't interchangeable! The terminal voltage is measured only via the V_{sense+} connection on K1. Another word of warning: the inputs to this connector must not be swapped since it would result in a negative voltage appearing on port A1 of the microcontroller, which could have some serious consequences.

The state of the lead-acid battery voltage is shown with the help of three leds: LED2, LED3 en LED4 (red, orange and green). The blue led (LED1) is used to indicate when a current pulse occurred. Note that it lights up for longer than the pulse itself, to make it clearly visible. The four leds are driven using just two outputs of the microcontroller. Because the two leds each have a zener diode of 2.4 V connected in series (D3+D4 or D5+D6, each 4.8 V combined), neither of the leds will light up at 5 V. When the output of the controller is set to tri-state, both the leds will therefore be off. When the output of the microcontroller is high or low, depending on which led has to be turned on, there will be enough of a voltage across the led to light it up. Apart from the leds, a lot more information is provided on the LCD.

The microcontroller used here, a PIC16F1847 (IC2), is the most extensive 18-pin 8-bitter made by Microchip. Output B3 is used to drive the gate of MOSFET T1 via optocoupler IC4. Just before the pulse, it measures the no load terminal (V_0) and during the 100-µs pulse the terminal voltage under load (V_t), shunt high voltage (V_h), and shunt low voltage (V_l) are measured using its three ADCs on AN1, AN0 and AN6.
When the circuit is first switched on, the voltage across the MOSFET has to be measured between TP1 (MOSFET drain) and the –connection of K3 (MOSFET source). As mentioned earlier, this voltage must not drop when the circuit is turned on.

The microcontroller circuit (the 5-V regulator, the microcontroller, the LCD, and the leds) consumes about 7 mA. The DC/DC converter uses about 16-19 mA. This isn't very economical, but it can easily be provided by a 9-V battery when taking a couple of quick measurements of 1.2-1.5-V batteries.

Voltage measurements and jumpers

Several software settings can be configured using jumpers JP2, JP3 and JP4. JP1 is a tri-state input (high, open, low). The 10-kΩ resistor at the input is essential, since the input is also an output for a short period of time in order to determine the tri-state status. After all, an output must never be connected directly to 0 V or 5 V.
The voltages to be measured (V_t , V_h and V_l) have to be attenuated to no more than 5 V for the ADC inputs. This is done via the resistor networks built using R2-R19. Voltages up to 30 V have to be reduced by a factor of 6. This is achieved by connecting 6 resistors in series across the voltage to be measured. When the ADC is connected to the bottom resistor, the resulting voltage (V_{ADC}) is V_{in}/6. For the 0-20 V measurement range we short R2 and R3, with JP2 in the 'H' position. V_{ADC} is then V_{in}/4. And finally, we short R2-R5 by placing the jumper in the 'L' position to select the 0-10 V measurement range. V_{ADC} is then V_{in}/2.
Apart from the three potential dividers, there is a fourth jumper (JP1), which is used to tell the microcontroller which voltage division has been selected. JP1 to JP4 have been positioned neatly next to each other on a 100-mil grid. We have converted a 2x4-pin header to ensure that all jumpers will be in the same position at the same time, see **Figure 2**.
The repetition period of the pulses can be adjusted via JP5 to either 30 s (no jumper) or 10 s (jumper in place). With lead-acid batteries you must always select 30 s.

It is vital that the three voltage measurements take place during the 100µs pulse. Whether this is possible depends on both the hardware as well as the software. The critical components here are the optocoupler, the MOSFET and the de microcontroller. The VO615A optocoupler has rise times (T_{on}) and fall times (T_{off}) of 9-10 µs with a load of 1 kΩ. This implies that the time left for the three ADC measurements is 100 – (2 × 10) = 80 µs.
With a clock of 16 MHz the minimum time required to convert one bit ('T_{ad}') is 1.0 µs. For a 10-bit ADC the total time required is 11.5 T_{ad}, which is 11.5 µs. The acquisition time (T_{acq}) is equal to Amplifier Settling Time + Hold Capacitor

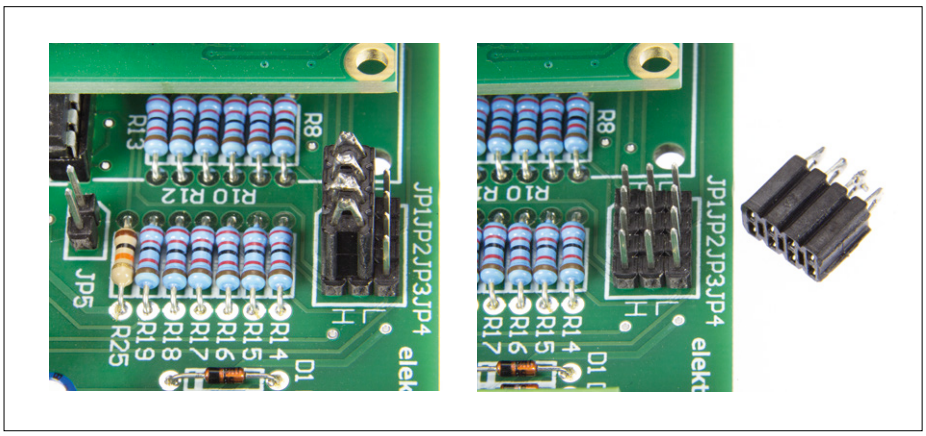


Figure 2. We've made a jumper-block from a 2x4-pin header to help with configuring the potential divider.





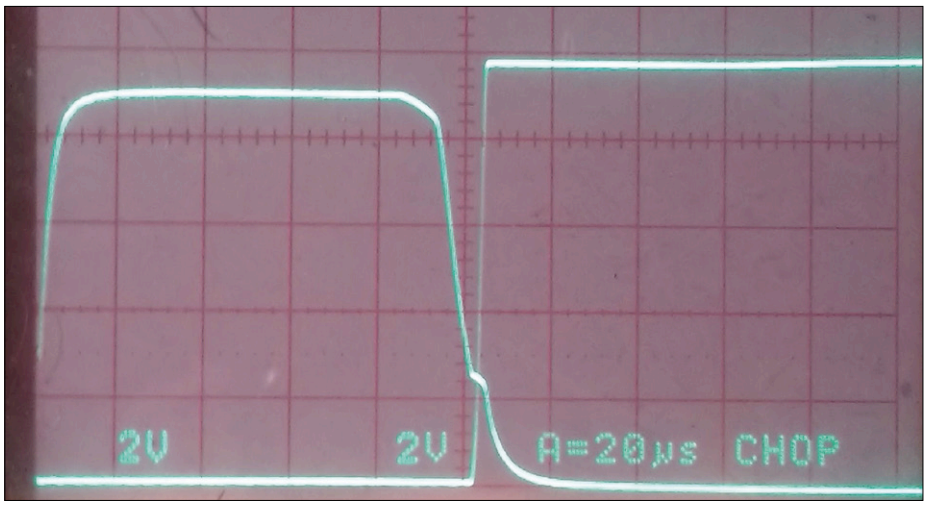


Figure 3: Top curve on the left: The gate voltage (V_{gs}) with a pulse duration of 94 µs for three ADC measurements. The other curve shows the voltage across the MOSFET (V_{ds}), with a total shunt time of about 105 µs (V_{gs} > 3 V at a shunt current of 0.4 A and V_o = 10 V).

Charging Time T_C + Temperature Coefficient = 2.0 µs + T_C + 0.05 ΔT (compared with 25 °C). T_C depends, among other things, on the source resistance (R_s) and for this microcontroller is equal to:
7.62 × 10 pF × (8 kΩ + R_s).

For the optimum source resistance of 10 kΩ, the value of T_C is 1.4 µs. In our circuit, R_s = 22 kΩ//22 kΩ = 11 kΩ to 22 kΩ||(5×22 kΩ) = 18 kΩ, so T_C is between 1-2 µs. The Temperature Coefficient is so small that it can be ignored. The total ADC conversion time then becomes about 11.5 + 2 + 2 = 16 µs at 25 °C. However, in the software a longer period of 10 µs is used for T_C, which makes the total ADC conversion time equal to 24 µs. Three ADC measurements therefore require 72 µs, which falls neatly inside the available 80 µs!
The software also requires 0.25 µs per program step (ADC enable, read, save and disable). The standard available procedures (in Flowcode 6) made the three ADC measurements take up about 150 µs, which was too long. When these Flowcode procedures were changed to procedures written in C, these times were reduced by 33%, which meant that three ADC measurements could take place within 80 µs. That was exactly what we needed.

The previously calculated switching times for the gate correspond fairly well with those measured in practice, as can be seen in **Figure 3**. We can also see that V_{ds} is much steeper than V_{gs}, which is a good thing. The rise time of V_{ds} is about 0.8 µs and the fall time about 3 µs. This is a good result when you consider the high currents that are switched. There is therefore no need to provide the MOSFET with a heatsink.

Software

The software has been written in Flowcode 6. The Flowcode source file and the corresponding .hex file can be downloaded from [4]. The startup screen shows the selected voltage division, the voltage range, the shunt resistance and the repetition period. The program then waits until a battery voltage is measured at the shunt. Following the startup screen, the no load terminal voltage (V_0) and the terminal voltage under load (V_t) are automatically displayed on the first line of the 2x16 LCD. On the second line the size of the pulse current in Amps and the internal resistance of the battery in mΩ are shown. At the bottom-right is a countdown timer that shows how many seconds are left until the next pulse occurs. All of the values are displayed using three digits. The real accuracy is determined by several factors, although it's good enough to correlate the internal resistances with the state of charge of the batteries.
There is a one-second delay after the battery has been connected in order to avoid any readings of possible voltage changes while the battery is being connected. While the circuit starts up, the leds light up sequentially (red-orange-green).
If you wish to modify the Flowcode yourself, it is essential to set the clock to 'internal' and 16 MHz, and to turn off the external MCLR and low-voltage programming (Configuration words: config1 0x09A4 and config2 0x1CFF).

Some results

The circuit had to be tested on my car battery, of course. Since I often drive only short distances, the internal resistance of 6 mΩ shows that it's not in the









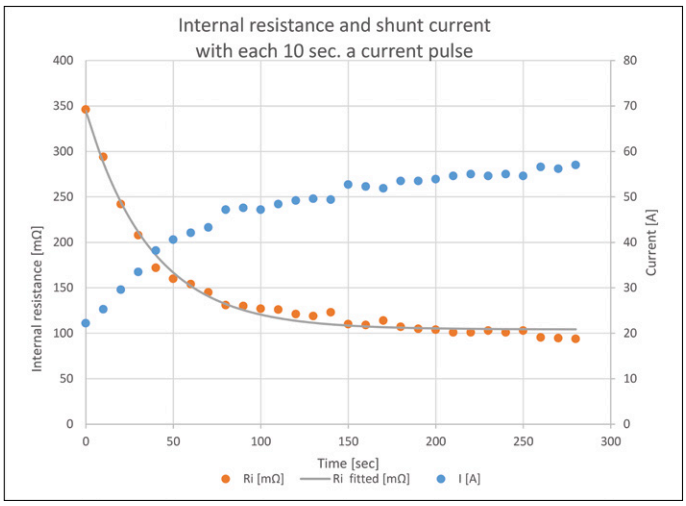


Figure 4. The restorative effect in a worn lead-acid battery (12 V, 7 Ah) with a 10.3 volt terminal voltage.

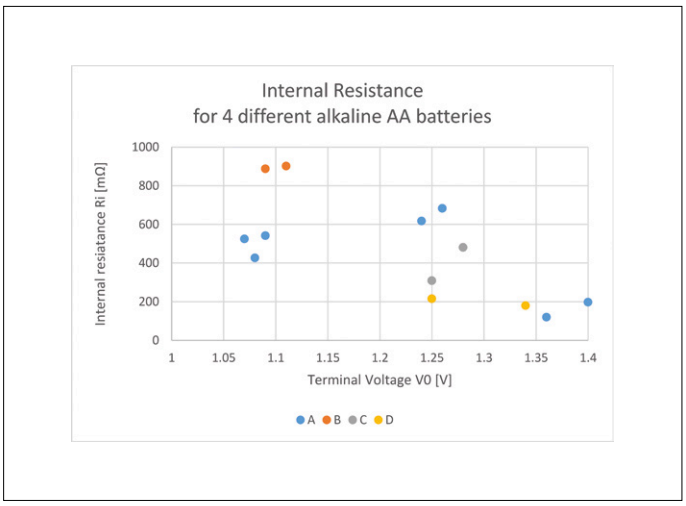


Figure 5. The internal resistance of AA alkaline batteries (type LR6) as a function of their no load terminal voltage. Four different brands of battery were measured, labeled A, B, C and D.

best of health. But at least the car still starts reliably!

The example in **Figure 4** shows the restorative effect of the current pulses in another battery. After just four or five pulses the internal resistance has halved and is getting close to its eventual value. In a 12-V lead-acid battery of 7 Ah we find that the temperature dependence of the internal resistance is about −0.7 mΩ/°C. It changes in value from 34 mΩ at room temperature to 62 mΩ in the freezer (−18 °C).

A new lead-acid battery (12 V, 7 Ah) has an internal resistance of 34 mΩ ± 2 % when it was charged, while an 8-year one of the same type had a resistance of 52 mΩ ± 3 %. Both types have a terminal voltage of 13.2 volts.

Twelve sub-C NiMH batteries of 4600 mAh, connected via solder tags with low resistances, where discharged and had an internal resistance of 101 mΩ. After charging them with 2000 mAh the value dropped to 86 mΩ. We can report the following regarding the accuracy we experienced in practice: The lead-acid batteries returned values with a variation of 1% or less, which was very good. On the other hand, AAA and AA batteries (alkaline and NiMH) had variations of over 10% in the measurements. This means that one measurement of such a battery is not enough to get a reliable value. In order to obtain the correct result you should take a number of measurements for these batteries and then take the average. Fortunately it only takes a minute to take six measurements.

Lastly, I measured all of my AA alkaline batteries. The result of this can be seen in **Figure 5**. It's very noticeable that there was a large spread between the different brands. A well-known brand seems to come off worst, or was that just chance? It can be clearly seen that there is a tendency for the resistance to increase as the batteries are more discharged. However, further research is needed before the differences between the brands can be explained. It looks like this tester won't be getting much rest! ⏮

(160064)

FROPM THE STORE

→160064-1
PCB

→160064-41
Programmed controller

→160064-71
Kit of parts

→120061-74
2x16 character LCD

Web Links

[1] Conrad lead-acid battery activator 191123: www.conrad.com/ce/en/product/191123/

[2] 'A lead-acid battery desulfatation tutorial': www.chargingchargers.com/tutorials/battery-desulfation.html

[3] Battery and Energy Technologies in Electropaedia: www.mpoweruk.com/performance.htm

[4] Software download: www.elektormagazine.com/160064

welcome in your ONLINE STORE

EDITOR'S CHOICE



Andonstar USB Microscope

Building prototype boards with SMD components by hand is no easy task, and with modern tiny components the results are impossible to inspect with the naked eye. A USB microscope is a solution. The Andonstar V160 has a sturdy stand which ensures a stable image. In addition, everything is precisely adjustable. The desired field of view can be set with knobs on the stand, and the focus can be adjusted with a rotating ring on the camera body. The camera is pleasantly small and produces a good, sharp image at a distance of several centimeters above the board, so there is enough room to do soldering work under the microscope. The microscope can also be adjusted to an angle on the stand to allow components and solder joints to be viewed from the side. Now that we have this microscope in the lab, we hardly every use magnifying lamps any more for SMD soldering. The V160 is a lot nicer!

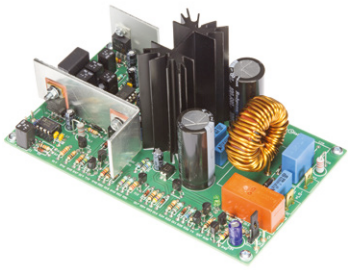


Luc Lemmens (Elektor Labs)

www.elektor.com/microscope-v160

Elektor Bestsellers

1. D-Watt
   www.elektor.com/d-watt



2. Sand Clock
   www.elektor.com/sandclock
3. Mastering Microcontrollers Helped by Arduino
   www.elektor.com/mm3
4. Digital Microscope V160
   www.elektor.com/microscope-v160
5. BBC micro:bit (Book)
   www.elektor.com/microbit-book
6. SmartScope Maker Kit
   www.elektor.com/smartscope-kit
7. Elektor Uno R4
   www.elektor.com/elektor-uno-r4

Mastering Microcontrollers Helped by Arduino

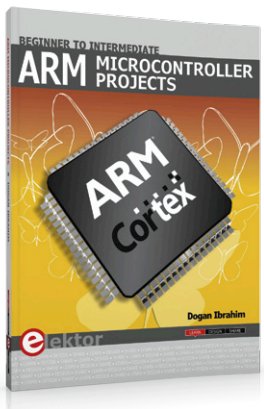


This book will not only familiarize you with the world of Arduino but it will also teach you how to program microcontrollers in general. Having completed this fun and playful course, you will be able to program any microcontroller, tackling and mastering I/O, memory, interrupts, communication (serial, I²C, SPI, 1-wire, SMBus), A/D converter, and more. This third, extended and revised edition contains two new chapters: AVR Playground and Elektor Uno R4.

member price: £33.95 • €38.65 • US $43.00

www.elektor.com/mm3

ARM Microcontroller Projects

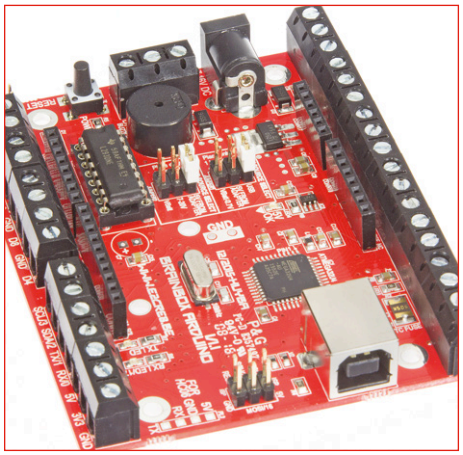


This book makes use of the ARM Cortex-M family of processors in easy-to-follow, practical projects. It gives a detailed introduction to the architecture of the Cortex-M family. The architecture of the highly popular ARM Cortex-M processor STM32F107VCT6 is described at a high level, taking into consideration its clock mechanisms, general input/output ports, interrupt sources, ADC and DAC converters, timer facilities, and more.

member price: £28.95 • €35.95 • US $38.00

www.elektor.com/arm-projects

BrainBox Arduino



BrainBox Arduino is a rugged version of the Arduino Leonardo. Thanks to the solid connectors, the 500mA outputs, the servo outputs and the many power supply options this Brainbox is immediately useful for most projects you can think of. Perfect for educational and DIY purposes by the solid construction and the large amount of free educational materials.

member price: £46.95 • €53.96 • US $57.00

www.elektor.com/brainbox-arduino



BBC micro:bit Bundle

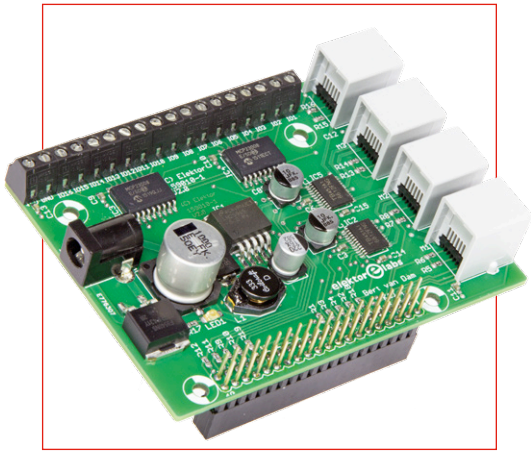
Book + Companion Experimenter's Kit

This book is about the use of the BBC micro:bit computer in practical projects. The BBC micro:bit computer can be programmed using several different programming languages, such as Microsoft Block Editor, Microsoft Touch Develop, MicroPython, and JavaScript. The book makes a brief introduction to the Touch Develop programming language and the MicroPython programming language. It then gives 35 example working and tested projects using these language. The companion experimenter's kit includes everything to get started right away with the projects.

MEMBER PRICE: £43.52 • €49.65 • US $53.55

www.elektor.com/microbit-bundle

LEGO Control Board for Raspberry Pi



This Raspberry Pi HAT puts at your disposal 4 motor-control outputs for LEGO EV3 motors and 16 buffered I/O connections that can be used in combination with a powerful Raspberry Pi. Programming is possible in languages such as C and Python. The board combines the flexibility of LEGO with regards to mechanical construction and the flexibility of the RPi with regards to software, Internet communications and sensors.

member price: £34.95 • €40.46 • US $43.00

www.elektor.com/lego-rpi-board

Red Pitaya for Test & Measurement



The Red Pitaya is a credit card-sized, open-source test and measurement board that can be used to replace most measurement instruments used in electronics laboratories. This book aims to teach the principles and applications of basic electronics by carrying out real experiments using the Red Pitaya. Many fun and interesting experiments are included. The book also makes an introduction to visual programming environment.

member price: £26.95 • €31.45 • US $34.00

www.elektor.com/red-pitaya-book

Sand Clock



This supercool gadget built around an Arduino Uno writes the time into a layer of sand. After an adjustable time the sand is flattened out by two vibration motors and everything begins all over again. This kit contains everything to build this sand clock by yourself: all mechanical parts, the motors, an Arduino Uno, a special RTC/driver shield, a power adapter and even a small bag with sand.

member price: £92.95 • €107.96 • US $114.00

www.elektor.com/sandclock

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 three Elektor book vouchers.

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

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



Solve Hexadoku and win!

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

Participate!

Ultimately March 23, 2017, supply your name, street address and the solution (the numbers in the gray boxes) by email to: **hexadoku@elektor.com**

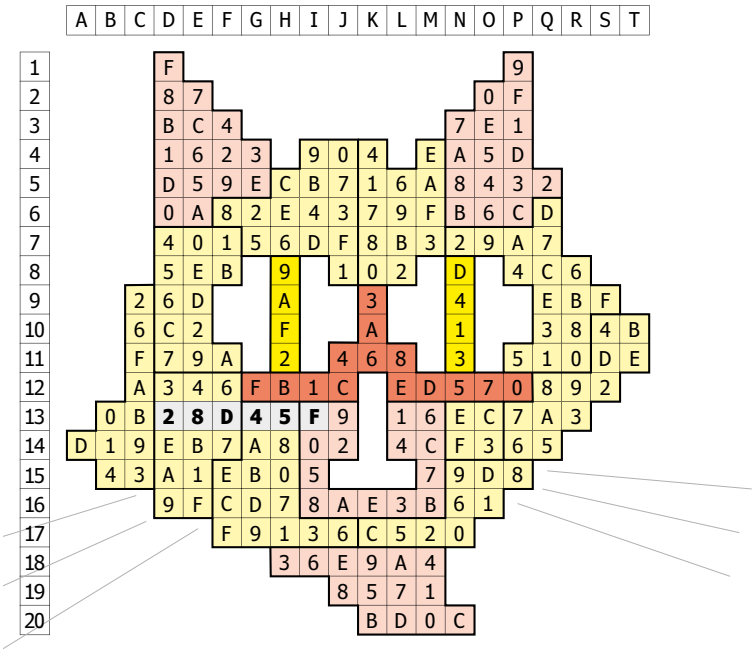
Prize Winners

The solution of Catdoku in edition 12/2017 (January & Ferbruary) is: **28D45F**.

The €50 / £40 / $70 book vouchers have been awarded to: Nico Rozendaal (The Netherlands), Gilbert Luyckx (Belgium) and Torben Munk (Denmark).

Congratulations everyone!

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



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

High Impact Embedded Graphics GUI Design Has Never Been This Simple

Free Visual Design Tools & Graphics Libraries for Your GUI Applications



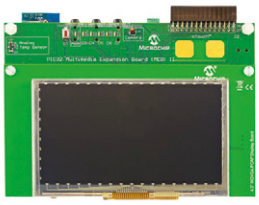
Microchip offers an industry leading complement of comprehensive visual Graphical User Interface (GUI) development tools, software graphics libraries and hardware tools for all your 32-bit graphics needs.

Our graphics solutions are supported with the free MPLAB® Harmony software framework and offers developers the choice of two best-in-class tools:

1 Our MPLAB® Harmony Graphics Composer works in conjunction with our MPLAB Harmony Graphics Library to help you generate professional looking GUIs without writing a single piece of code!

OR

2 You may choose SEGGER emWin Pro as your graphics library and take advantage of its expansive list of widgets and the SEGGER toolchain.



Multimedia Expansion Board II (DM320005-2)

Get started today by downloading training material, documentation and tools!

microchip DIRECT www.microchipdirect.com

MPLAB® HARMONY

MICROCHIP

www.microchip.com/MCU32GFX

The Microchip name and logo, the Microchip logo and MPLAB are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries. All other trademarks are the property of their registered owners. © 2017 Microchip Technology Inc. All rights reserved. DS60001462A. MEC2138Eng01/17

DO YOU WANT THE BEST ELECTRONICS DESIGN SOFTWARE



FEATURES

- Schematic Capture
- PCB Layout
- Gridless Autorouting
- 3D Visualization
- M-CAD Integration
- SPICE Simulation
- MCU Co-simulation
- Built in IDE
- Visual Programming

NOW INCLUDES:
Serpentine Routing, Layer Stackup Manager, and Assembly Variants.

Labcenter Electronics

www.labcenter.com

(+44) 01756 753440