Edition 2/2019 | March & April 2019 | www.elektormagazine.com

elektor

elektorlabs

DESIGN BUILD SELL **ELECTRONICS** magazine

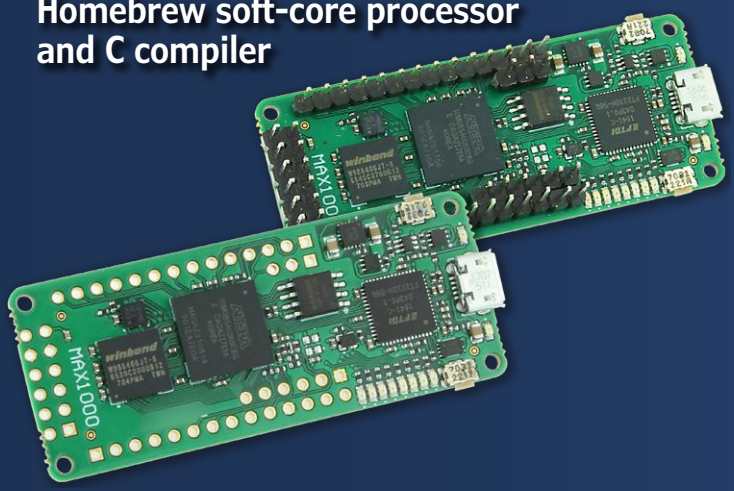
Beat the Elektor Pinball Clock



Ah... the sound of them score drums rolling ...

The SCCC Project

Homebrew soft-core processor and C compiler



Air Pollution Monitor



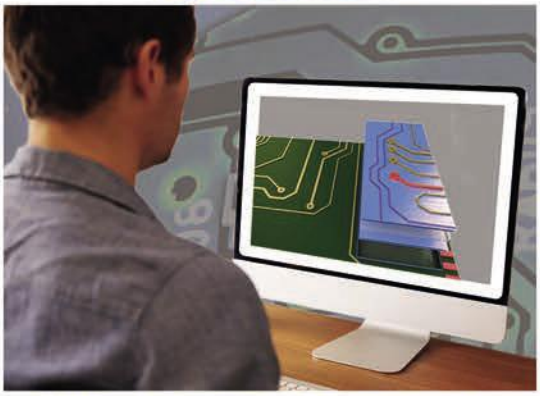
Any smokers hiding in the toilet?

A Beginner's Guide to Microcontroller Development Kits ⊳ An ENS210 Dew Point Sensor e-BoB ⊳ ESP32 Battery Monitor ⊳ Fan Controller ⊳ Hardware Design using (V)HDL (3) ⊳ Interference from LED Lamps ⊳ Interference Sources in Automotive Applications ⊳ IOTA Cryptocurrency (and an FPGA for the RPi) ⊳ Nuclear Event Detector ⊳ Operational Amplifiers in Practice ⊳ and more...

PROTEUS DESIGN SUITE

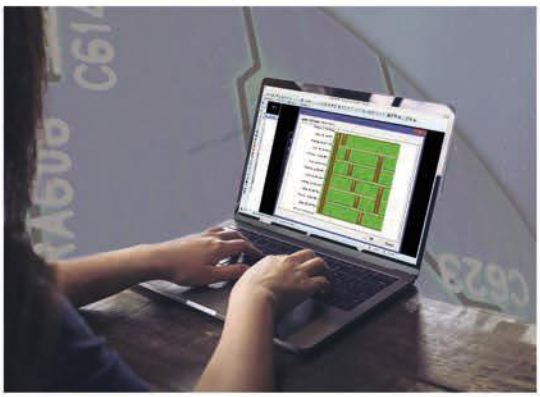
Advanced PCB features for professional board design

DESIGN ROOMS



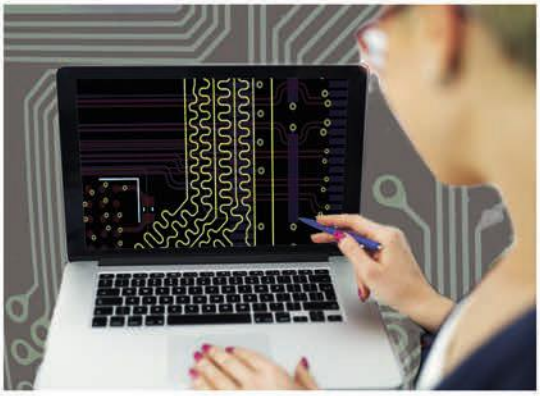
Set design rules that apply in user specified areas of the PCB.

LAYER STACKUP



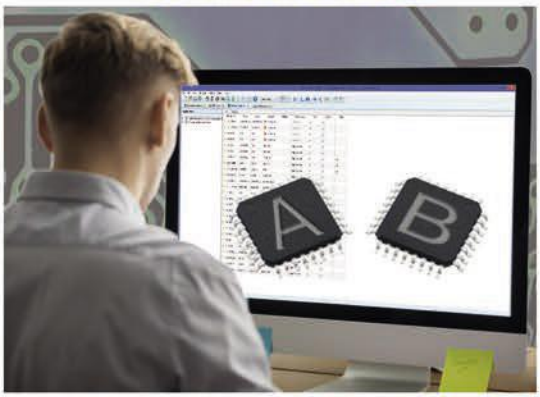
Control the layer stackup and drill ranges for smarter routing.

SERPENTINE ROUTING



Easily length match tracks against each other or to a target distance.

DESIGN VARIANTS



Edit the fitted status of parts or re-place with pin compatible alternatives.

The Proteus Design Suite provides advanced features at an affordable price. Try it today!

Visit: www.labcenter.com

Tel: +44 (0) 1756753440
E-Mail: info@labcenter.com
youtube.com/c/LabcenterElectronicsLtd

Labcenter Electronics www.labcenter.com

Elektorlabs Magazine
Edition 2/2019
Volume 45, No. 494
March & April 2019

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

www.elektor.com
www.elektormagazine.com

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

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

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

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

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

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

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. 2019
www.eimworld.com
Printed in the Netherlands



Programming for Everyone

Good news: our long-time and co-Aachen-based author Martin Ossmann succeeded in realizing his own small softcore processor that “runs“ inside an FPGA. A compiler, also “home grown” by Martin, converts simple C code into machine commands that the 32-bit processor understands (page 26). Professor Ossmann didn’t have to reinvent the wheel — he was able to draw on the work from Ron Cain going back to the early 1980s. Ron at that time had developed a simple but powerful C-Subset for the Z80.

The eight pages of “The SCCC project“ article teach a lot about the hardware description language Verilog, the architecture of a CPU, and compiler building. They are not exhaustive though and more concrete applications are in development at Martin’s. Typical for this day and age is the ability for anyone to participate using a low-cost board and just a few components. For around 30 euros you can get the small MAX1000 board (no. 18816) from the Elektor Store, which integrates a well-equipped FPGA, RAM at 8 MB each and Flash, as well as user LEDs and buttons.

Even cheaper are the boards presented from page 6 onwards by equally well established Elektor authors Burkhard and Fabian Kainka. For little money, all readers not able, willing or brave enough to take their first steps into the exciting world of controller programming, can now take part actively. And the beginner-friendly Arduino development environment and many, many sample programs are available free of charge anyway.

Jan Buiting, Editor-in-Chief

The Circuit

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

This Edition

Volume 45 – Edition 2/2019
No. 494 March & April 2019

Regulars

25 Elektor Labs Pipeline

47 Peculiar Parts, the series
Nuclear Event Detector

62 HomeLab Helicopter
Wondrous Things Electronic Spotted From Above

73 Err-lectronics
Corrections, Feedback and Updates to published articles

76 Elektor Store Highlights
Measuring, powering and printing... with gear from the Elektor Store

86 Questions & Answers
(Almost) everything you always wanted to know about... Enclosures (2)

89 Peculiar Parts, the series
Balanced Antenna Cable

104 Retronics
From the Graphics Department, with Fun

110 Elektor Ethics
How Can You Implement Ethical Values Concretely in your project?

114 Hexadoku
The original Elektorized Sudoku

Features

6 A Beginner's Guide to Microcontroller Development Kits
The first step is the... easiest!

58 Operational Amplifiers in Practice
Part 2: FET-input opamps and wide-bandwidth applications

68 PureBasic
One BASIC Compiler, many platforms

78 Hardware Design using (V)HDL (3)
Driving a WS2812 LED matrix

98 Interference Sources in Automotive Applications
And how to get rid of them

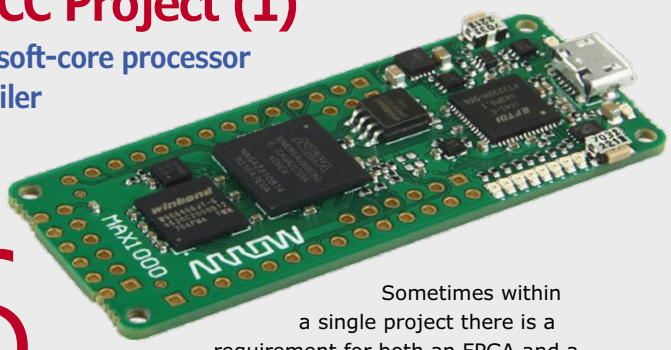
102 Interference from LED Lamps
Do interfering LED lamps conform to EMC standards?

38 The NXP Cup 2019
is Bigger and Better Than Ever!



The SCCC Project (1)

Homebrew soft-core processor and C compiler



26

Sometimes within a single project there is a requirement for both an FPGA and a classical CPU, so that complex program flows can be implemented simply. One solution in such situations is a processor realized within an FPGA. While it is possible to purchase such 'soft cores', they are not usually particularly cheap, and so in this article we look at a simple homebrew CPU in an FPGA. Thanks to the open-source code it is possible to modify the design to suit your own purposes. To this we add a DIY C compiler, which can also be extended. And, as usual, the series of articles describing the project will expand on the theory by including as many practical examples as possible.

Projects

14 Beat the Elektor Pinball Clock!
Ah... the sound of them score drums rolling ...

26 The SCCC Project (1)
Homebrew soft-core processor and C compiler

34 Fan Controller
Your choice of Arduino or analogue electronics



104

Retronics

From the Graphics Department, with Fun



elektorlabs magazine

52 'Wolverine' CNC Gantry Robot Multifunctional Powerhouse
Part 1: Mechanical construction

66 Rotary Encoder Interface without µC
Simple and compact

83 An ENS210 Dew Point Sensor e-BoB
with a sample implementation on Arduino UNO

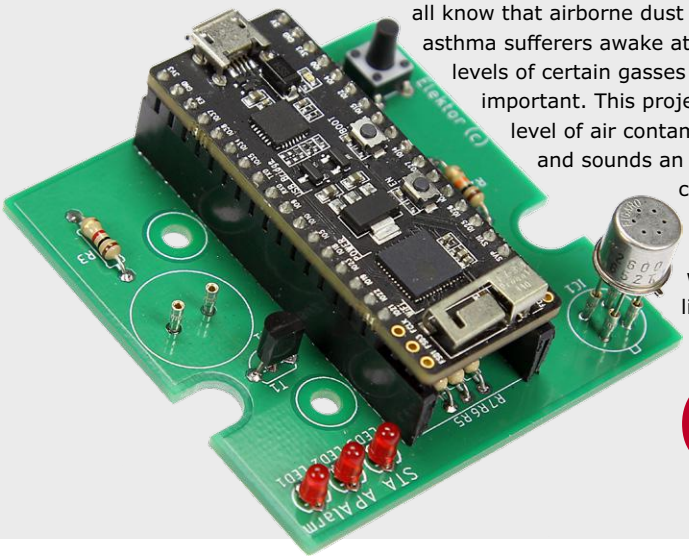
90 Air Pollution Monitor
Any smokers hiding in the toilet?

94 I²C Data Bus Reader
Everything on the PC screen, via USB

Air Pollution Monitor

Any smokers hiding in the toilet?

The quality of the air we breathe should be of deep concern to all of us. We all know that airborne dust particles keep asthma sufferers awake at night, but the levels of certain gasses are equally important. This project measures the level of air contaminant gasses, and sounds an alarm when their concentration exceeds a certain level, for instance when someone lights a cigarette…



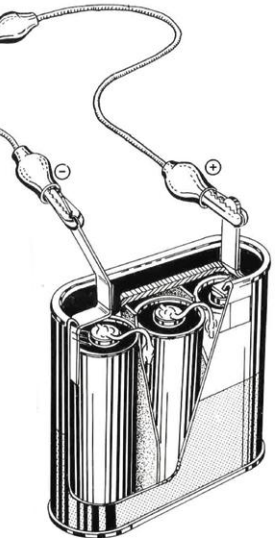
90

Next Editions

ElektorLabs Magazine Edition 3/2019

General Purpose 9-Channel Relay Control Board with PC Interface ● IOTA Cryptocurrency (and an FPGA for the RPi) ● Locating Wayward Wires ● ESP32 Battery Monitor ● Artificial Intelligence ● Simple 7-Band Audio Spectrometers ● Gradient Detector ● SRC Pulse Generator ● Two-anode MOSFET thyristor ● SMD Stencil Alignment ● Weller WT 1014 Review ● USB Current Control Unit ● 'Wolverine' 3-Axis CNC Portal Machine part 2.

ElektorLabs Magazine edition 3/2019 covering May and June 2019 is published on 18 April 2019. Delivery of printed copies to Elektor Gold Members is subject to transport.
Contents and article titles subject to change.



39 RPi Stepper Motor HAT with Anti-resonance
Boost efficiency and calm stepper-motor jitters

44 Rear Radar for Bicycles
a simple project for better bicycle safety

48 The Raspberry Pi Ruler Gadget
Fun with a time-of-flight sensor

Elektor Industry Edition 2/2019

Edition 2/2019 of *Elektor Industry* magazine will cover sensors as well as equipment and methods for test and measurement, with contributions from companies, industry specialists, Elektor editors, and free-lance authors. Edition 2/2019 has a special focus on the **Sensor + Test trade show** in Nuremberg, Germany, from June 25–27, 2019.

Elektor Industry magazine issues are available to ElektorLabs magazine Gold members in print, and Elektor Labs magazine Green members as a PDF download, as well as for purchase by anyone, at www.elektormagazine.com. Publication date: 18 June 2019.

A Beginner's Guide to Microcontroller Development Kits

The first step is the... easiest!

Too many electronics enthusiasts harbour a certain reluctance to enter the world of microcontrollers and programming. It really is not so difficult, as we will show by introducing four low-cost boards. You will see the setup and how some simple demo programs can be run so that early success is guaranteed. To get things going we gave a low-cost Arduino board to Burkhard Kainka who is one of our long-time contributors to Elektor and a specialist in many fields of electronics. His son Fabian will be giving us the lowdown on the NodeMCU board — a platform designed especially for IoT applications.

Too many electronics enthusiasts harbour a certain reluctance to enter the world of microcontrollers and programming. One reason is that there is just too much choice — faced with the jungle of competing development platforms — it is difficult to find the kit that's right for you. Not too complicated, well supported by third-party manufacturers in both hardware and software, and with a supportive community of users who welcome newbies.

We did the searching for you and picked out four controller-boards/kits suitable for beginners and gave them to some electronics wizards to unpack and put through their paces. You can read their impressions; it might help you to discover that your first steps into the world of microcontrollers may not be so daunting after all!

We start off in this issue with the father and son team of Burkhard and Fabian Kainka. Burkhard is already a long time contributor to Elektor, responsible for many popular articles, projects and books ranging in subject from RF technology to microcontrollers. In this issue he is taking a close look at an Arduino nano clone.

His son Fabian is following in his father's footsteps and has specialised in developments and publications on the topic of the Internet of Things. The microcontroller kit he will be looking at is not unrelated to this topic: the NodeMCU Kit.
In the next edition, an ESP32 Wi-Fi/Bluetooth board with an integrated OLED display will come under the spotlight.

JOY-iT Nano V3

By **Burkhard Kainka** (Germany)

The JOY-iT Nano V3 is an Arduino Nano compatible clone from JOY-iT, on sale in the Elektor-Store at just under under 12 Euro. A useful addition to this board is the Arduino Supplement Kit which includes additional components to enable a quick and easy start to microcontroller board experimenting.

These days a spare Arduino board in the lab is an essential resource to quickly prototype new ideas using its powerful processor, versatile connectivity and simple software development environment.
I usually use the Standard Model Arduino Uno, the Nano is much smaller but almost the same hardware based around an ATmega328. This means there's lots of storage space for big projects and more computing power than you are likely to need.

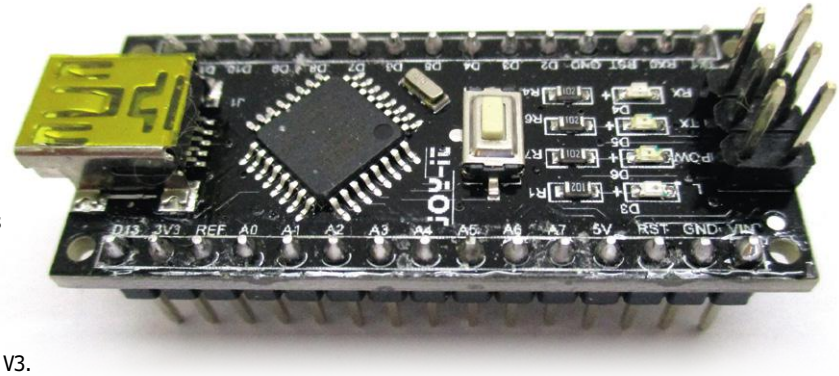


Figure 1. The Nano V3.

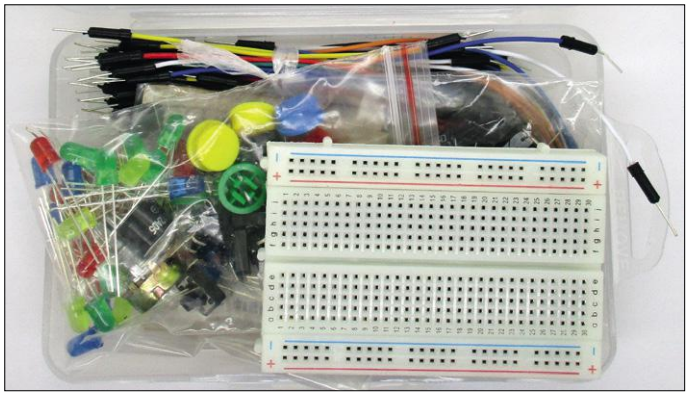


Figure 2. All the goodies in the Supplement Kit.

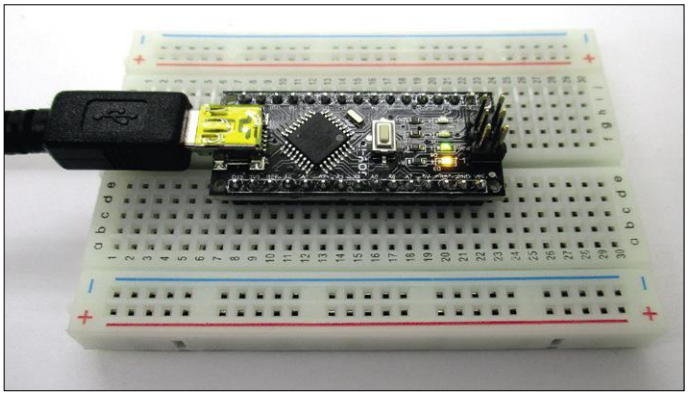


Figure 3. The Arduino Nano in use.

First impressions

At first glance there does not seem to be too much difference between an original Arduino Nano and the Joy-It Nano clone (**Figure 1**), that's before you flip the board over. Underneath you will see an FT232R USB to UART converter chip on the original board and a CH340G on the clone. Both do the same job and convert ATmega serial interface signals into signals to and from the USB port. When plugged into a PC it creates a virtual serial interface so that the board can communicate with the PC. It assigns a COM port number automatically, for example, it uses COM2, COM3, or COM99, depending on how many other devices have previously used a serial interface on the PC. For a FT232R, new COM numbers are constantly being created. The CH340 is different; the next board will appear with the same COM port, which you can rename if you want to use multiple boards simultaneously. If a problem occurs with the USB driver on a computer, you will need to download and manually install the driver [1].

The **Arduino Supplement Kit** (**Figure 2**) contains a prototyping plug-board and many useful components from flying leads to pots. This gives you more LEDs, resistors, capacitors, push buttons and other stuff than you can use in a week.

Now we can get to test the Nano and mount it onto the prototyping plug board (**Figure 3**) this is a stable platform that gives me confidence I can run and test a program without the risk of an accidental short-circuit. A USB cable with a mini-USB plug is all I need now to connect to a PC. The green LED on the board lights up to show the board is powered up and the yellow LED continually emits brief flashes. For Arduino this indicates that no program is loaded yet and the bootloader is waiting to do something.

The Arduino IDE must now be fired up on the PC. Whenever I test a new Arduino, I always first turn to the example program `Blink.ino` (**Figure 4**), which is included in the IDE. Whenever you refer to programs that run on an Arduino they are known as 'sketches'. `Blink.ino` flashes the yellow LED at 2 s intervals. Before the program can be loaded on the Arduino, I have to select the Arduino Nano board as target in the IDE and activate the communication interface (COM2). The last controller board I used with this PC was also fitted with a CH340 and it used the COM2 port also.

Next the upload button was activated and I waited until the upload was completed. No errors were detected and the LED starts to flash slowly. I can't resist fiddling so I just change the delay time in the blink program to 2000 ms and upload the sketch again. It flashes slower.

This all works just as I had expected… so what could we test now? I had a rummage around in the supplement kit and found a buzzer that was just crying out to be tested. It looked like the type of buzzer with internal electronics. I hook it up to the 3.3 V output of the Nano and it bleeps, next I try it at 5 V, woah… that's piercing. The buzzer comes with a label over the sound outlet which I immediately re-attach. Could we try a resistor in series with the buzzer to quieten it a bit? Why not — any value higher than 100 ohms however is too high and stops the buzzer from working. It must work connected directly to a GPIO pin of the Nano board so I connect it to pin 13, which also drives the on-board yellow LED. That works as expected — the buzzer now sounds every time the LED comes on.

I am beginning to suspect that the buzzer is not a piezo-type as I first thought but is instead an electromagnetic buzzer. I wave a bar magnet close to the buzzer and there is a noticeable

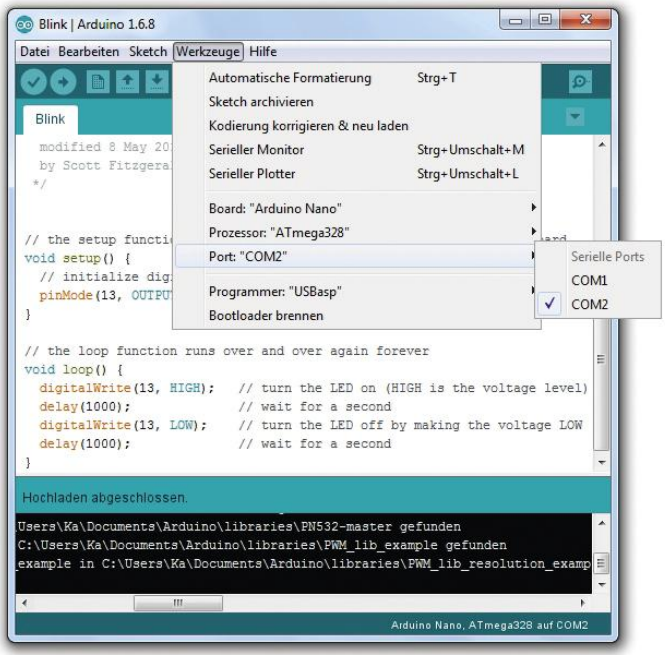


Figure 4. Setting up and loading the Nano in the IDE.

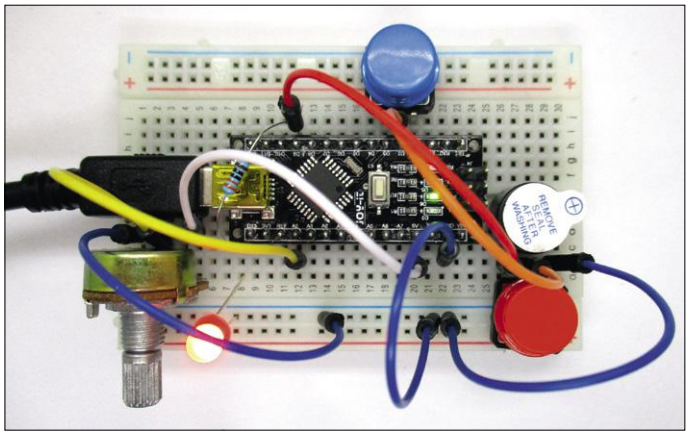


Figure 5. An experimental setup.

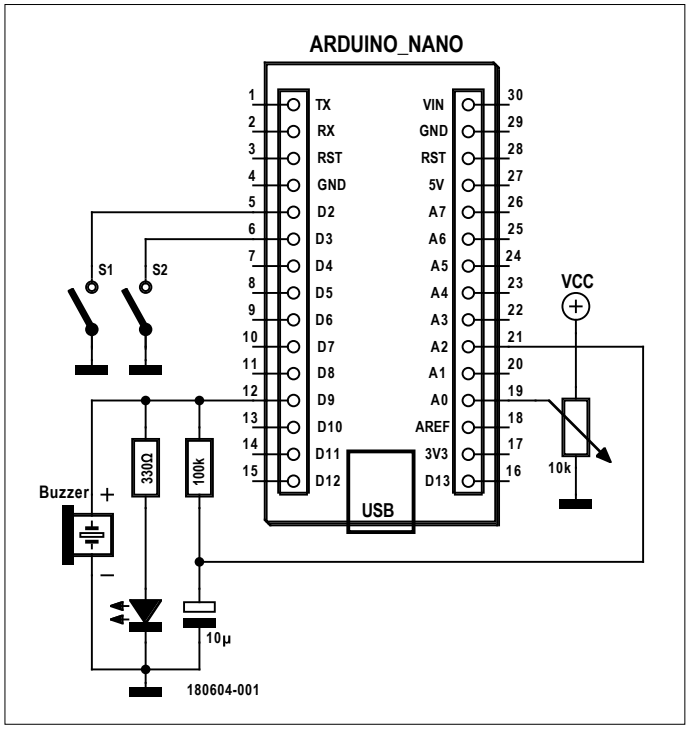


Figure 6. A low-pass filter using 100-kΩ resistor 10-µF capacitor.

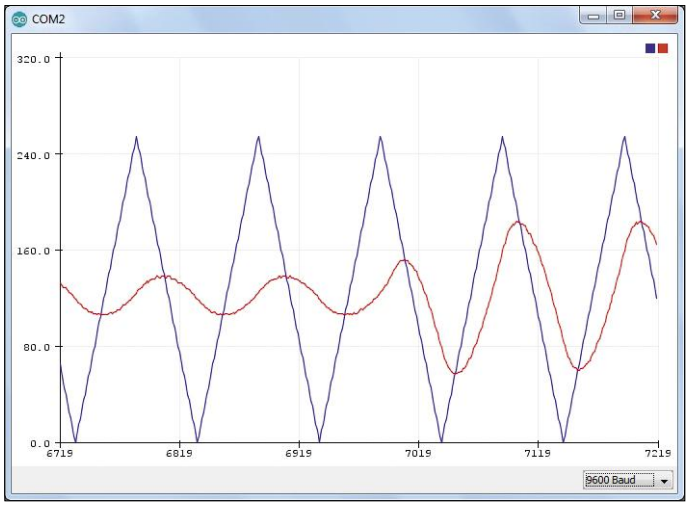


Figure 7. Output waveform and filtered signal.

pull when I get close. This indicates that inside is a magnetic speaker membrane with a solenoid coil driven by a transistor oscillator. This type of buzzer needs a lot more power than a comparable piezo buzzer and is the reason it won't work with a high value resistor in series. An ATmega328 port pin however can provide enough power — it can even drive very low-power DC motors without any additional motor driver stage.

Drive the buzzer with PWM signals

What else could we build using the components in the kit? I tested a red LED with a 330-Ω series resistor to D9 using the example sketch `Fade.ino` from the Arduino IDE. It all worked straight away with the LED continuously glowing brighter then darker. The dimming is all produced by quickly switching the port output on and off.

What would happen if I hooked up the buzzer to the PWM output signal? Actually, that worked as well although it's not really intended to be driven that way. The LED glows brighter and dimmer and the buzzer volume changes—to a lesser extent — at the buzzer frequency. During operation you can hear some additional harmonics produced when the natural buzzer frequency and multiples of the PWM frequency coincide.

What else can we fit on the breadboard? At first glance, there doesn't seem to be much space left but it is possible to squeeze a few more components on the board. Two extra-large push-button, one pot, the buzzer, the LED, the resistor and many cables, all fit well (**Figure 5**). I just connected to two of the push button pins and left the other two unused, hanging over the edge of the board.

Now the Fade example needs to be modified so that the red button stops the output and switches off the sound and the blue button switches it back on. I can use the pot value setting to set the speed. As usual, things don't always go without a hitch and I need to do some background reading to find the solution (I'd rather be working with Bascom). In the end everything worked out as planned. Although the project doesn't really perform anything useful and is a little annoying (hit S1 — the stop button) it's good just to get confidence by using the hardware and software features of Arduino.

I did notice something peculiar when using the board; after downloading and test running a program I unplugged the board and then plugged it into a USB power bank to make the application portable. When I later plugged it back into the PC, the

@ WWW.ELEKTOR.COM

→ JOY-iT Nano V3
www.elektor.com/joy-it-nano-v3

→ Arduino Nano
www.elektor.com/arduino-nano-3

→ Supplement-Kit
www.elektor.com/arduino-supplement-kit

Books:

→ Michael Margolis, „Arduino Cookbook"
www.elektor.com/arduino-cookbook

program was gone! The yellow LED was blinking, indicating an empty Nano. What's going on? A search on the web brought the answer: It is all to do with the version of the Arduino IDE I was using. In Arduino Version 1.6.8. the system will always put a newly plugged-in Arduino into boot mode. Why hadn't I noticed this before? Maybe it's because I usually use the slightly older (1.6.5) version of the IDE; its startup sequence is a bit faster.

Plotting analogue signals

Sometimes version 1.6.8 or higher must be used, especially if you need to make use of the serial plotter display. This allows you to plot varying signal levels. In version 1.8.2, the plotter can even display multiple channels. The program was modified again to plot the value of the variable `brightness`. In addition, a low-pass filter made up with a 100 kΩ resistor and 10 µF capacitor was used to smooth the PWM signal which is measured at the analogue input pin A2 (**Figure 6** and **Listing 1**). The result can be seen in **Figure 7**. The brightness is controlled with a triangular waveform, but the RC filter makes it look closer to a sine wave function. After some time, the pot was rotated to slow the output. The signal period varies relative to the time constant of the RC filter. The amplitude of the 'sine wave signal' increases and the signal becomes distorted. Such experimentation would normally need a bench full of test equipment but here the Nano manages everything on its own.

ISP Programming

By the way, you are not tied to using the Arduino IDE. In principle, you can also use any other development environment for AVR controllers. The hex file can then be flashed to the Nano using the ISP plug. This makes this small board even more universal. As I mentioned earlier, I like to use Bascom. In the microcontroller series ('Microcontroller BootCamp', *Elektor* magazine from 4/2014), an Arduino Uno uses a specially adapted Bascom boot loader and this will also work with the Nano which uses the same ATmega328 controller and runs at the same 16 MHz clock frequency.
I tried to flash the bootloader into the Nano using an STK500 but there was a problem. The Nano (both original and clone) uses a particularly low value (1 kΩ) pull up resistor on the reset pin so the STK500 was not able to pull the pin low enough to trigger a reset. In contrast the Arduino Uno uses a 10 kΩ pullup at this point, which the STK500 can easily handle. For the Nano, I tried a quick

Listing 1. The modified Fade example.

```
/*
 Fade
 This example shows how to fade an LED on pin 9
 using the analogWrite() function.
 This example code is in the public domain.
 Additional: pot + s1 off + s2 on + AD + serial plot
 */

int led = 9;           // the pin that the LED is attached to
int brightness = 0;    // how bright the LED is
int fadeAmount = 5;    // how many points to fade the LED by
int s1 = 2;
int s2 = 3;

// the setup routine runs once when you press reset:
void setup() {
  // declare pin 9 to be an output:
  pinMode(led, OUTPUT);
  pinMode(s1, INPUT_PULLUP);
  pinMode(s2, INPUT_PULLUP);
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // set the brightness of pin 9:
  analogWrite(led, brightness);
  Serial.print(brightness);
  int U = analogRead(A2);
  Serial.print( " ");
  Serial.print(U/4);
  Serial.println ( " ");
  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;

  // reverse the direction of the fading at the ends of the fade:
  if (brightness == 0 || brightness == 255) {
    fadeAmount = -fadeAmount ;
  }
  // wait for … milliseconds to see the dimming effect
  delay(analogRead(A0)/10+10);
  if (digitalRead(s1) == LOW){
    analogWrite(led,0);
    while (digitalRead(s2) == HIGH);
  }
}
```

Web links

[1] CH340G driver: www.wch.cn/download/CH341SER_ZIP.html

[2] The author's Web site: www.b-kainka.de/

and dirty fix by connecting an external 1 kΩ resistor between reset and GND. Now in normal operation the reset voltage level is lower but not low enough to trigger a reset. Now when the SKT500 issues a reset the level falls and a reset occurs. The same could happen if you use another type of programmer, if yours doesn't work, first check the voltage level at the reset input.

NodeMCU

By **Fabian Kainka** (Germany)

NodeMCU is an open-source IoT platform based on the ESP8266 32-bit Wi-Fi SoC microcontroller made by Espressif, a company who specialize in products for the Internet of Things.

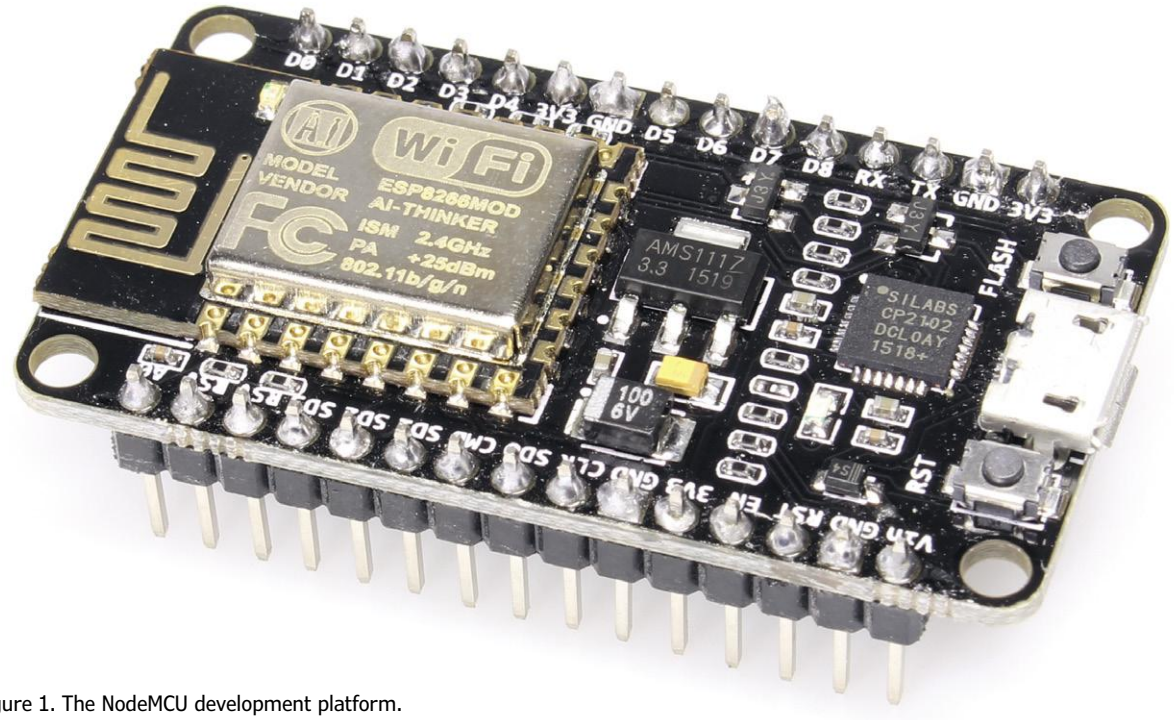


Figure 1. The NodeMCU development platform.

NodeMCU Technical data.

- The ESP12E Module
- 32-Bit Low-Power CPU clocked at 80 MHz
- 4 MB Flash memory for storing the program and files
- Wi-Fi: 802.11 b/g/n at 72.2 MBit/s
- Printed Wi-Fi antenna
- 3.3 V operating voltage
- Programming and power via Micro-USB port.
- DP2102 USB-to-Serial converter
- 13 GPIO pins
- 10-Bit ADC (successive approximation)
- SPI, I²C, I²S, 2 x UART, IRDA, PWM, RTC
- Onboard LED, 2 pushbuttons

As you can see from **Figure 1** the hardware uses the ESP12E Wi-Fi module. It has a total of 13 GPIO user pins available to build a wide range of applications. It uses a DP2102 chip to convert USB to serial communication. An LED and two push buttons (one for reset) are also included on the board.

NodeMCU by default is actually the name of the firmware rather than the hardware kit. It uses the Lua language which is an interpreted language and means that the firmware also contains a Lua interpreter. Lua (Portuguese for 'moon') is a scripting language and is the real reason I'm interested in the NodeMCU board. An ESP8266 can be programmed (like comparable boards) fairly easily using the Arduino IDE. For me the attraction of Lua is that the program is interpreted at runtime and not translated previously by a compiler. This should make software development simpler because the pro-

gram can developed and debugged on the fly.This can best be compared to the console on Linux or the Windows Power Shell, which is able to accept commands at any time, and run script files automatically. Here there is no need for a development environment, just a simple text editor. This can make your job easier as you will see later.

Unboxing

The first step was to take the board out of its box and hook it up to my PC using a micro-USB cable. To establish serial communication I used the Terminal program PuTTY, but any other serial monitor program should work just as well. After I had established the connection at 9600 baud, I was able to receive the first readable characters. They verified that the board is running firmware Version 0.9.6, dated mid 2015 — that seems quite old so I will need to set about finding a more recent version. On my quest I found a lot of other exciting stuff…

Flash the latest firmware

My journey led me to the corresponding GitHub web site [1] (GitHub is a web-based internet site for open-source code). Here all important information was linked, I was just hoping to download the latest firmware edition but to my surprise, as well as the firmware, each user can assemble their own version of the firmware. You can choose from over 65 different different modules which ones you want to integrate into the LUA firmware. There is not enough room to fit all the modules into the 4OO MB flash memory… that would be like installing all the libraries in the Arduino IDE onto an Arduino board. The user therefore needs to decide in advance which functions they are going to need.
Standard Modules such as Timer, UART, WiFi, GPIO and so on should be included in almost all firmware configurations, while modules like HTTP, MQTT, SNTP or specialized sensor modules such as DHT (Temperature and humidity), BME280 (Temperature, humidity and air pressure), HMC5883L (Triple-Axis Compass) or TCS34725 (colour/light sensor) should only be included when they are specifically required.

That all sounds like a lot of work to assemble all the individual modules and compile a running firmware. The whole process would be pretty cumbersome if it were not for the easy-to-use **Cloud Build Service** (**Figure 2**). On the website [2] you can simply ‘personalize’ your own personal firmware environment with all the necessary modules. There are additional options, for example, which of your branches you want to use from the GitHub project flow structure and whether you need TLS/SSL support or FatFs to read SD cards. Once you have made all your choices, you just need to enter your email address in the appropriate field and click **Start your build**. Creating the firmware takes a while (for me me it took about three minutes), then you get an email with the download link which remains valid for 24 hours.

The fully compiled firmware now only needs to be loaded to the board. There is also a practical tool, called the **NodeMCU PyFlasher** (**Figure 3**), which can be downloaded from [3] for Windows and MacOS. Installation is not necessary; the file can be executed directly. Select the correct COM port and downloaded the firmware file and then start the upload process. That was totally unproblematic for me. A detailed description

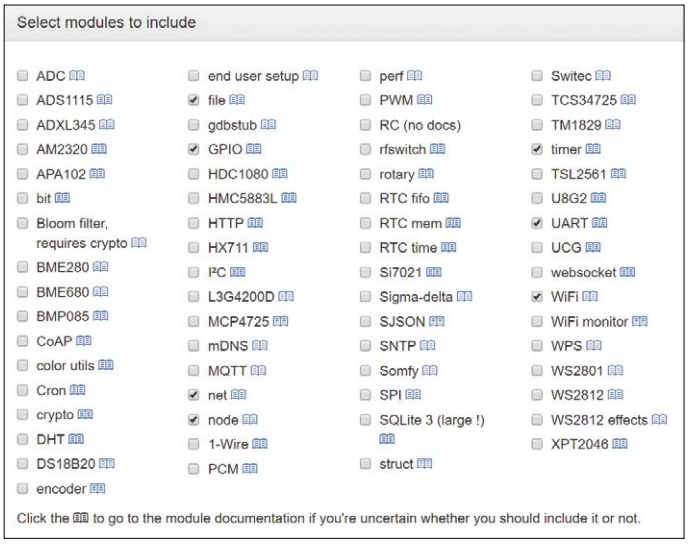


Figure 2. Module selection for the Firmware setup.

of how to create and flash the firmware and also information about all the modules and basic FAQs can be found on the excellent project documentation page [4].

Hello World – Lua-Test

On the documentation page I also found a link [5] to the very useful **ESPlorer** tool. This program (**Figure 4**) is not just a simple serial monitor but a kind of Swiss Army knife for the ESP8266 — regardless of whether you are using MicroPython, NodeMCU or an AT firmware.
After downloading and starting the executable ESPlorer.jar file I established the serial connection to the board. This latest firmware, communicates at 115200 baud. The program now tried to communicate with the board automatically without success. Only when I pressed the reset button on the board

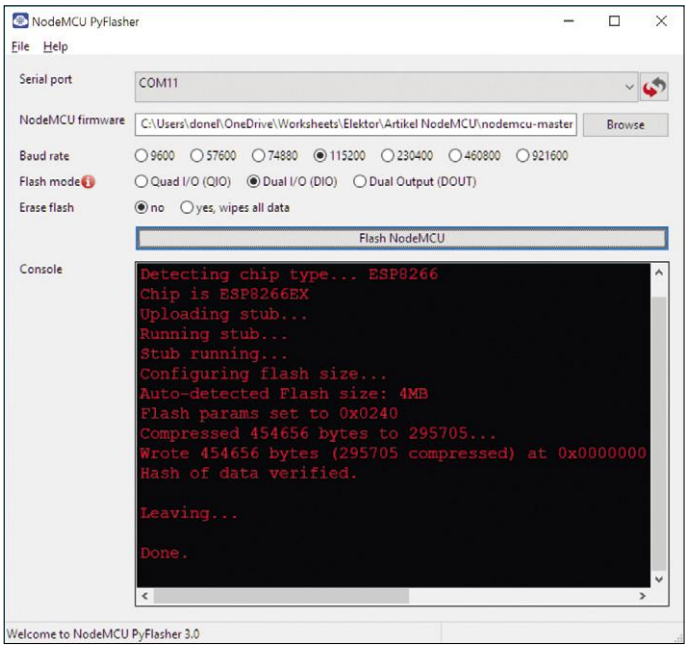


Figure 3. The PyFlasher tool.

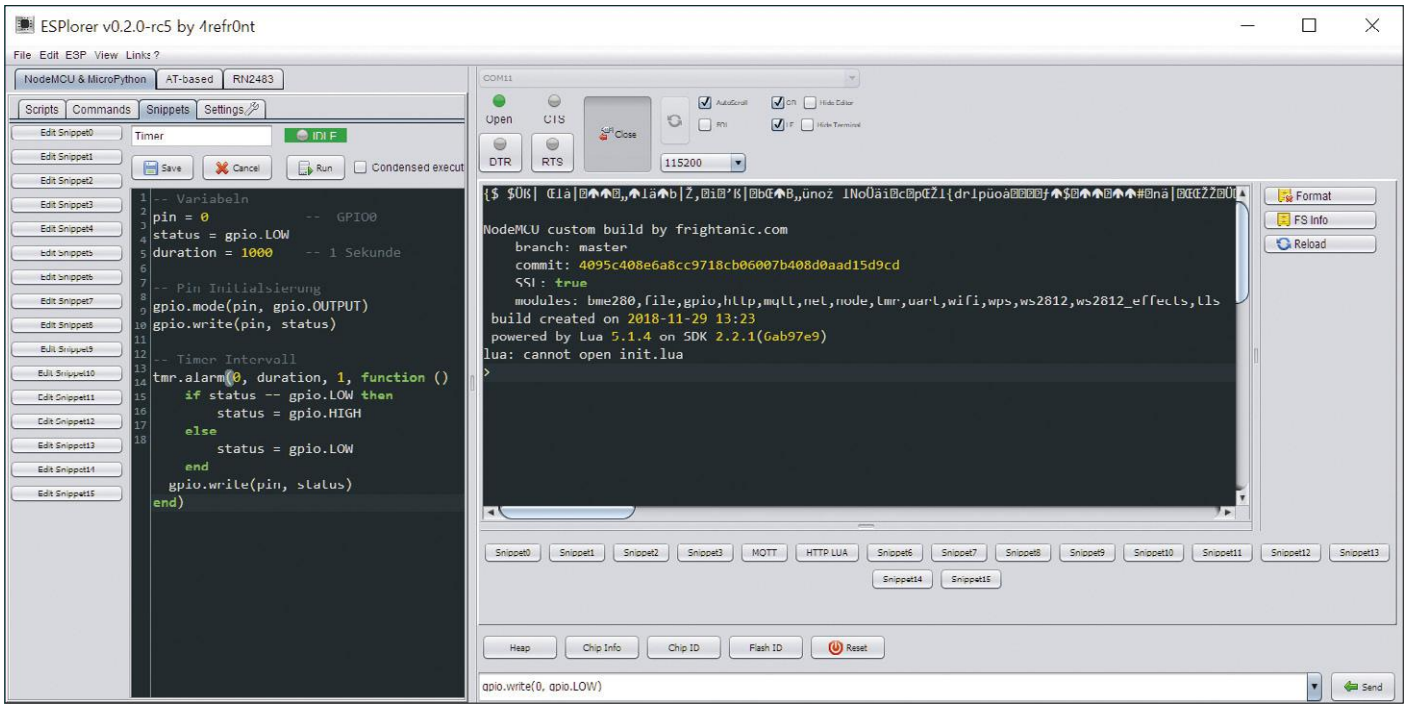


Figure 4. The ESPlorer tool: Terminal on the right, Snippets on the left.

did I get the first message telling me about the firmware version and all the modules, so communication was successful.

The ESPlorer program screen shows a line of buttons along the bottom on the right which can be used to send automatic commands to the board. For example, I can use them to read the chip ID or information on the file system. There is, of course, an input window for direct commands. I first tried entering the following commands:

```
gpio.mode(0, gpio.OUTPUT)
gpio.write(0, gpio.LOW)
```

If you are already familiar with Arduino source code these should be easy to understand. First, GPIO0 is configured as an output and then set to low. A series resistor and LED with its anode connected to VCC is attached to this pin so setting the output low will make the LED light up. That was just the simplest test I could think of at the time to see if the commands really work without compiling.
This manual input window is really handy if you only want to test one or two commands. It keeps track so it makes it simple to execute a command again. Most programs that do anything useful will of course need a whole sequence of commands. I wanted to implement the classic “Hello World” and let the LED flash. These are just a few lines, but fortunately ESPlorer has the ability to create entire code snippets and even save it on the PC. This snippet containing command sequences, called Snippets here, can then, with a click be automatically sent to the board. I built the following blink program by typing 15 editable snippets and then hit run to start:

```
-- Variables
pin = 0           --  GPIO0
status = gpio.LOW
duration = 1000    -- 1 sec
-- Pin Initialization
gpio.mode(pin, gpio.OUTPUT)
gpio.write(pin, status)
-- Timer Interval
tmr.alarm(0, duration, 1, function ()
    if status == gpio.LOW then
        status = gpio.HIGH
    else
        status = gpio.LOW
    end
  gpio.write(pin, status)
end)
```

Interestingly, variables do not need to be declared first. If you have some experience with JavaScript, you’re probably aware of that already. Also interesting is the use of the timer. In this case, the timer will simply pass an entire function defined in the call (called an anonymous function) that will run every second. The flashing LED was therefore no problem and the program was easy to understand. Here we are using an IoT board, so I created the following snippet to connect to my router:

```
wifi.setmode(wifi.STATION)
station_cfg={}
station_cfg.ssid="YourSSID"
station_cfg.pwd="YourPassword"
station_cfg.save=true
station_cfg.auto=true
wifi.sta.config(station_cfg)
wifi.sta.connect()

wifi.eventmon.register(wifi.eventmon.STA_CONNECTED,
```

Web links

[1] Git Repository: https://github.com/nodemcu/nodemcu-firmware

[2] Firmware: https://nodemcu-build.com/

[3] PyFlasher: https://github.com/marcelstoer/nodemcu-pyflasher/releases

[4] Documentation: https://nodemcu.readthedocs.io/

[5] ESPlorer: https://esp8266.ru/esplorer/#download

[6] MQTT example: https://nodemcu.readthedocs.io/en/master/en/modules/mqtt/#example

[7] The author's Web site: https://fkainka.de/

```
function(T)
    print("WiFi connected!")


end)
```

Once a connection has been established, the program puts out the message 'WiFi connected!'. Now I need to create an anonymous function that will be executed when the Wi-Fi connection has been established. This type of event-driven programming is typical in Lua.

The connection setup worked fine. The configuration used here also ensures that the Wi-Fi credentials are stored and maintained after a reset. In addition, `station_cfg.auto=true` ensures that the board will always attempt to connect to the network automatically. Without a corresponding program, no 'Wifi connected' message will appear, that will only occur after a board reset.
Since the connection process has worked so well, I wanted to go on and test the MQTT module. The documentation contains a good example of how to connect the board to a broker as an MQTT client; I copied it directly into a snippet with some slight modifications and then tested it 'live on the board' [6]. Here, too, everything went unusually smoothly and gave me confidence to move on to the next step.
Now I need to write a program that remains resident in the board memory, automatically executing when it reboots. This should be easy to achieve when you consider:

- The firmware provides its own Virtual File System to which files can be uploaded. These files can be script-files executed with th `dofile(file name)` command or web pages, images or other files that you want to store on the board.
- On reboot the board checks whether the *init.lua* file exists in the file system, if it does, it runs automatically.

Specifically, I simply switched from the *Snippets* tab to *Scripts* on ESPlorer and used the text editor. Here I wrote the following lines, saved to my PC as *init.lua*, and then Uploaded them to the NodeMCU:

```
wifi.eventmon.register(wifi.eventmon.STA_CONNECTED,
function(T)
    print("WiFi connected!")
    http.get("https://pastebin.com/raw/k4ccGx3T", nil,
function(code, data)
     if (code == 200) then
          if file.open("run.lua","w+") then
               file.write(data)
               file.close()
          end
      else
          print("HTTP request failed")
      end

      if file.exists("run.lua") then dofile('run.lua')
else print("run.lua not found") end
    end)
 end)
```

This program now does something that usually can only be done in scripting languages. Firstly it waits for the successful setup of a Wi-Fi connection and then in run time it downloads a program from the Internet which it stores with the filename *run.lua* and then runs it. In this example, it just downloaded a simple blink program from a Paste site (Pastebin.com). Any other resources can of course also be used here. Both the HTTP-Module and the TLS/SSL-Option were used in this firmware.
I was impressed with Lua and found it to be a really powerful and compact interpreted language. I imagine there are a whole series of programs on a web server that can be tweaked and improved. You can choose your program via a website or MQTT and download it to the board. It could hardly be more modular or more simple. ⏮

180604-02

@ WWW.ELEKTOR.COM

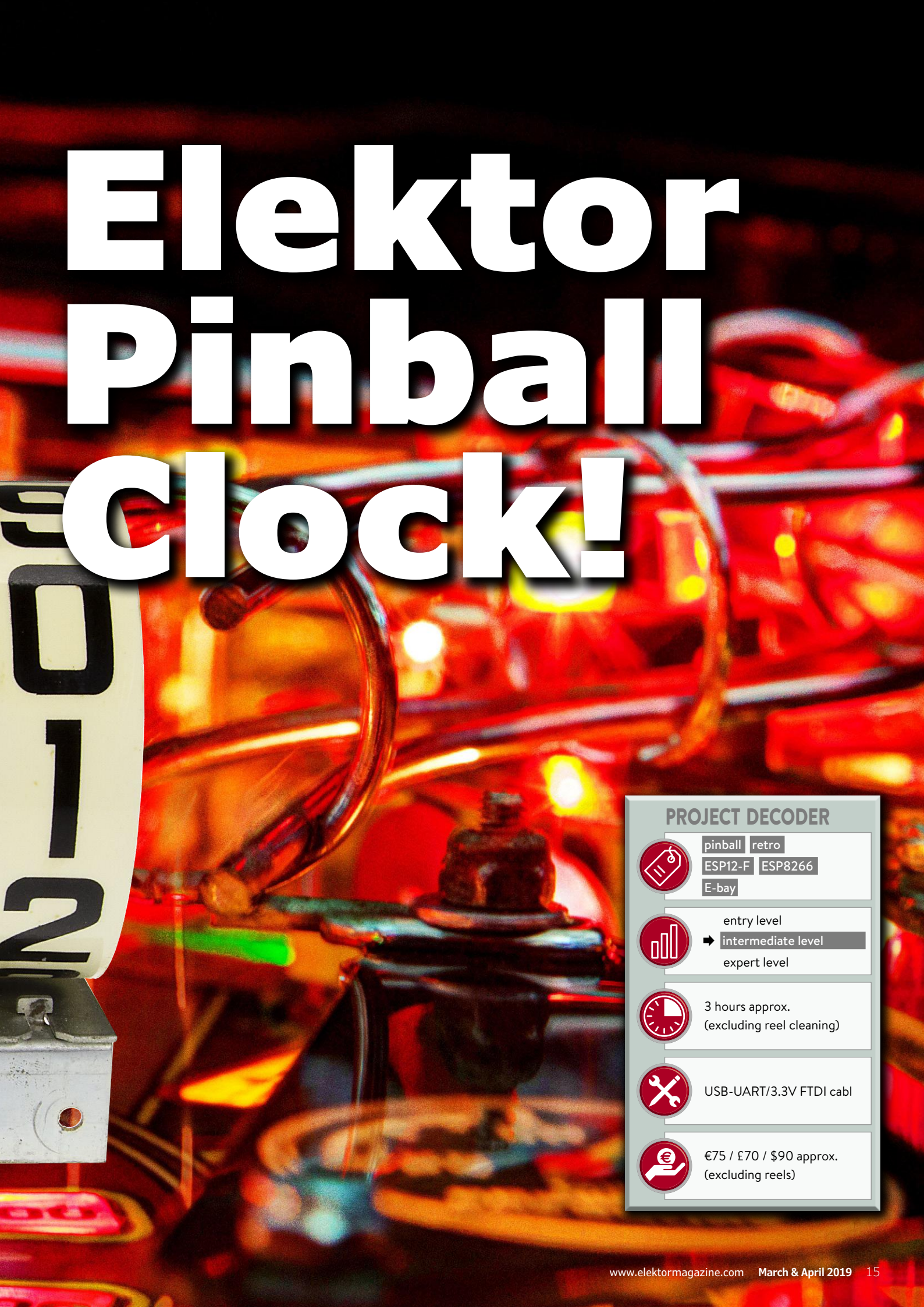
→ NodeMCU Microcontroller board
www.elektor.com/
nodemcu-microcontroller-board-with-esp8266-and-lua

→ Dogan and Ahmet Ibrahim: 'ESP8266 and MicroPython'
www.elektor.com/esp8266-and-micropython

Beat the



Elektor Pinball Clock!



PROJECT DECODER

pinball | retro | ESP12-F | ESP8266 | E-bay

entry level
➡ intermediate level
expert level

3 hours approx.
(excluding reel cleaning)

USB-UART/3.3V FTDI cabl

€75 / £70 / $90 approx.
(excluding reels)

Ah… the sound of them score drums rolling

By **Luc Lemmens** (Elektor Labs)

Vintage displays are sure to attract attention, and not just because they look nice. In this project cheerfully combining 1960s vintage with 2019 Wi-Fi and SMD, they also produce a **racket**, so the passage of time will not go unnoticed…

Sure, Elektor presented numerous projects and Retronics flashbacks on 'vintage' display devices like Nixie tubes, Numitrons, Dekatrons, VFD tubes, but so far just one thermometer project appeared that used electromechanical (EM) pinball score reels units [1], followed by a remake with a pinball credit reel unit to display temperature [2]. High time for a follow-up! This time, let's make a clock with score reels. Not exactly the most original application for recycling old displays, but it's one with the highest practical value and WAF (until noon, but we will discuss that later).

In honour of the displays' original application in bars, cellars and cafes, inside the project hides a simple pinball machine, which can be played on your computer or mobile device. It duly employs the reels for their original application: score keeping!

There's a glut of pinball clock designs around on the Internet, but as far as we know the present one will rank high as one of the more sophisticated and anachronistic, with an ESP8266 connecting the clock to the Internet through Wi-Fi for time and date synchronisation. All clock settings can be accessed and adjusted through Wi-Fi (smartphone, tablet or PC). Time and date can also be set manually and a battery-backed DS3231 Real Time Clock (RTC) IC will keep track of the time with 5 ppm accuracy, even with the power switched off.

You can set a daily (or should we say: nightly) interval during which the clock will remain silent, i.e. the score reels and chimes will not be active then. Alternatively, you can switch the display on and off manually (yep, or pull the power plug).

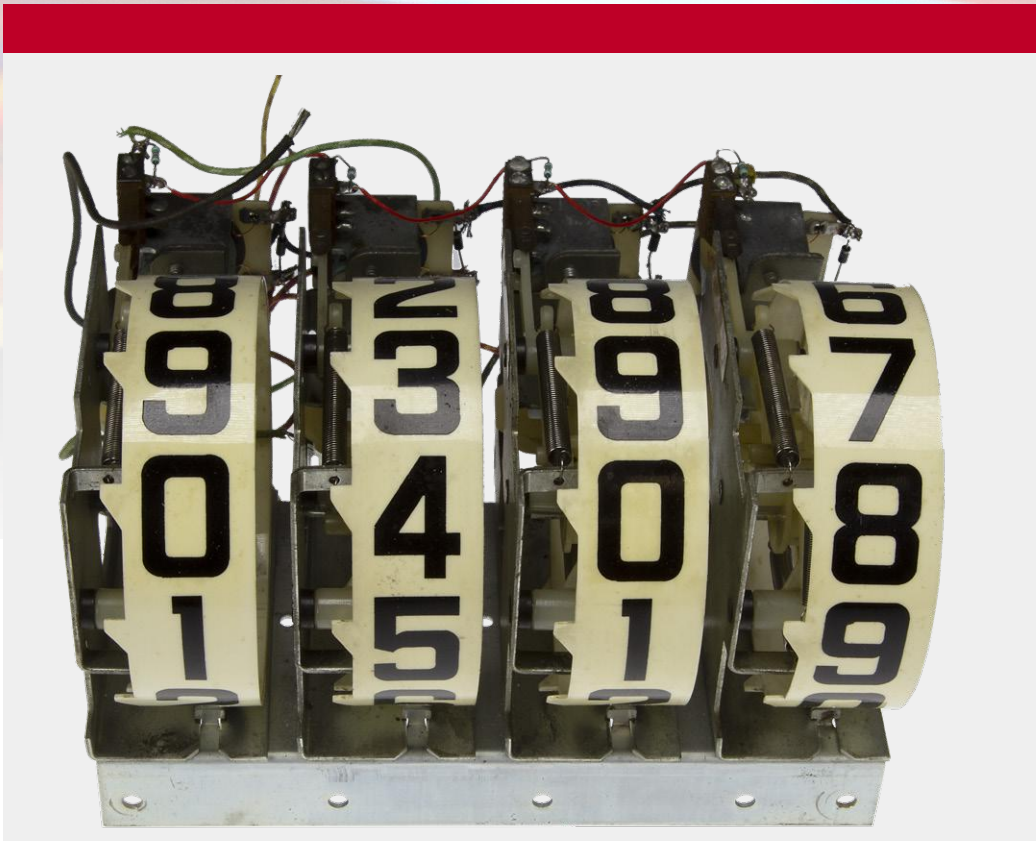
During normal operation time will be displayed either in 12- or 24-hour format, with the date shown every 30 minutes (either in European or US format).

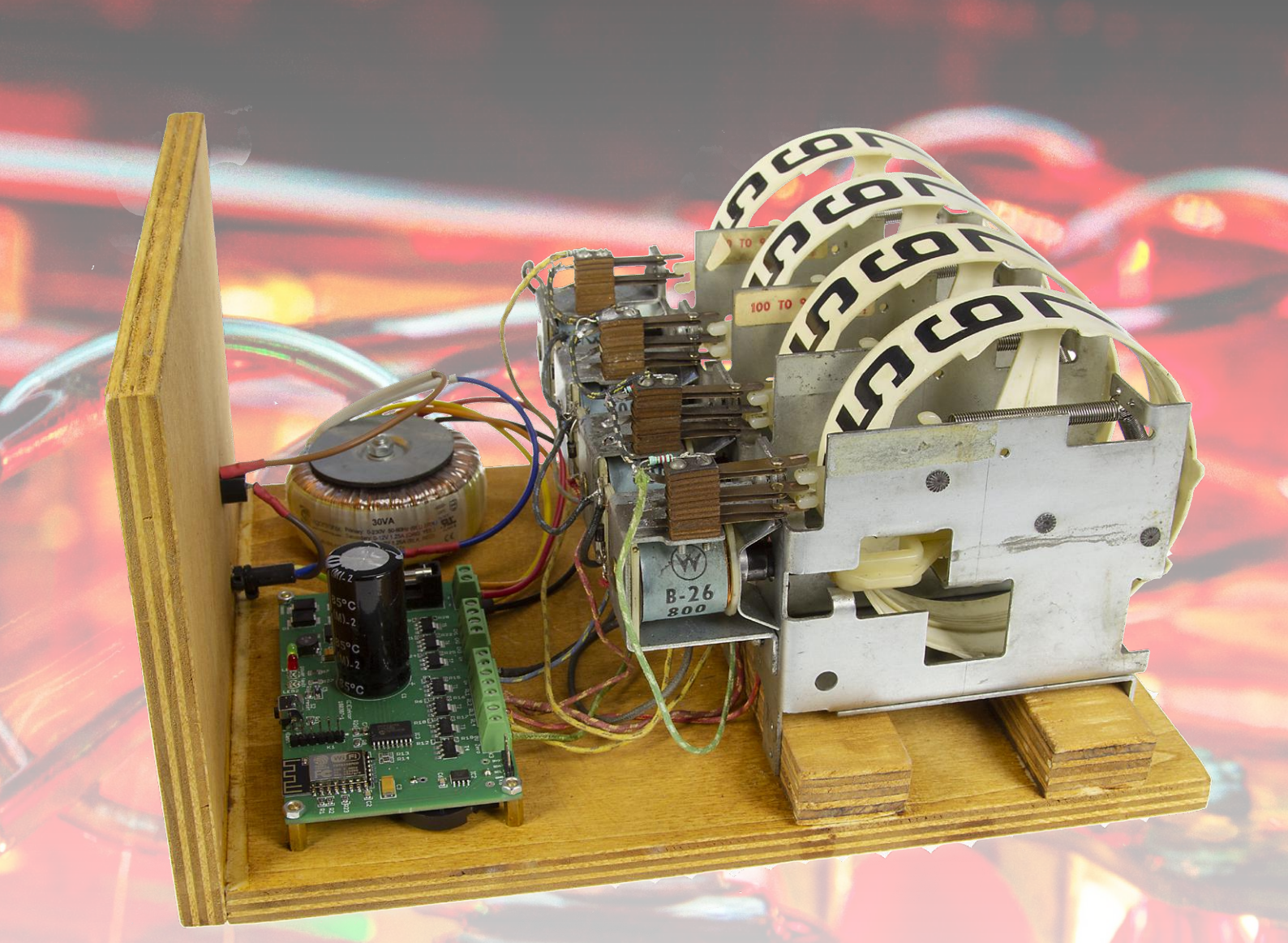
Electromechanical they were

Most pinball machines built from the mid 50's until mid-70's of the last century relied on electromechanical score reels (a.k.a. score drums) for displaying a player's score. The operating principle hasn't changed much over the years: a short pulse is applied to a coil, which causes the reel to increment one step for every pulse. A switch detects the reel's zero position, allowing the pinball machine to reset the score to zero when a new game is started.

In real life things are a bit more complicated. There are other switches on a score reel unit, for example a contact that signals if the reel has completed its step, one that closes in position '9' (carry to higher decade), and some reels can even register every digit position for free play scores and the so-called 'match' feature. But we don't use these contacts in our project, the coil and zero switch are the only electrical parts of interest for the clock.

There was one important change in design of score reels: the diameter of





Where to buy score reels, and which ones?

Any auction or classifieds site will be a good source, but there are also (web-)shops selling pinball scrap and reconditioned parts. All reels from well-known manufacturers like Gottlieb, Bally and Williams will do for our clock, plus you may find lesser known brands that will work too.
Just for the looks: look for four identical ones, there are different reel sizes and different fonts depending on the brand and age of the score drums.

Decarbonising

Once you have four score reels on your workbench, start with the most boring, but also most rewarding part of this project: cleaning! Most likely these things are covered with grime, dust and soot (hence: decarbonizing) and that doesn't only look ugly, but most likely the mechanical parts can use some cleaning too. Do yourself a favour and take the units apart completely (!) and clean every part thoroughly before reassembling. If you only do the outside, the mechanism may not run smooth enough for proper operating. Now, some do's and don'ts from an old hand:

- Most important: be very careful with the reel, the numbering can easily be damaged or even wiped off if you don't handle and clean it with care. For example, early Gottlieb reels were apparently printed with water-based ink…. In general: do not soak, scrub or scratch, and do not use (aggressive) cleaners.
- When reassembling: there is always some kind of notch or marking for correct orientation of the reel on the hub, misalignment can damage the unit or at least shift the zero-position of the drum.
- Moving parts have metal to plastic bearings, if a reel is not running smoothly you didn't clean it well enough OR did something wrong during assembly. Do NOT lubricate!
- Zero switches aren't difficult to identify: look for a blade switch that opens when the reel is in zero position. On more recent Bally and Williams reels this is an easy one: the switch on top of the unit. In that respect Gottlieb machines are far less repair-friendly with the switches hidden inside the unit, it may be a good advice to look for the zero switch when you have disassembled the unit for cleaning. In all cases: clean the contact points with fine sandpaper and adjust the blades if necessary.

A video showing instructions on (dis-) assembling, cleaning and repairing pinball score reels can be found on the Internet. But where did all those chimes go?

the drums was reduced from 5” through the early 60’s, to 4” and even later to 3”. The inertia of the larger counters limited the timing of score handling, and smaller reels could keep up with faster gameplay more easily. That’s why older pinball machines sound slower — less ‘snappy’ than more recent models. To mimic the pace and sound of the real pinball machine (i.e. the games the reels were designed for in the first place), our clock display can also be adjusted to the size of the score drums.

Pinball score drums can only be **incremented**, which means, for instance, that a reel at position ‘1’ can only be reset to ‘0’ by pulsing it nine times. Needless to say, this will definitely be the noisiest clock ever presented by Elektor. To be sure of that, and for the real ‘pinheads’ amongst us: an optional three-tone electromechanical pinball chime can be connected to strike at quarter, half and full hours.

The latter feature could be the most expensive part to add to the clock, since these chimes are becoming increasingly difficult to find and — in other words — increasingly costly. Alternatively, you can use a single bell or chime from an older pinball machine.

Coils in electromechanical pinball machines are AC powered, except for the flippers and so-called pop bumpers in later models. In brief, AC-coils can also be DC-operated, but at a lower voltage than their stated AC rating [3]. Please note that the document covers permanently powering AC-relays with DC. In this clock and in pinball machines (most) solenoids are only powered with short pulses, making it unlikely for the coils to overheat. This could only happen when a coil is permanently energised with the clock malfunctioning. In that case a fuse will blow, cutting the current before the coil fries. In real pinball life, a stuck-closed score switch will result in an evil smelling, burnt score reel coil that blows the 10+ amp fuse. Caused by windings’ isolation having melted down and the coil turned into a full short-circuit. In short, these coils where never designed to be permanently powered!

Except for early Williams games (pre-1963: 50 VAC), the coil voltage of both Williams and Gottlieb pinball machines was 28 VAC during the EM age. All Bally EM games used 50 VAC. All score drums will work fine on approximately 35 VDC, although a lower voltage (some 20 VDC) will also do for Gottlieb and Williams.

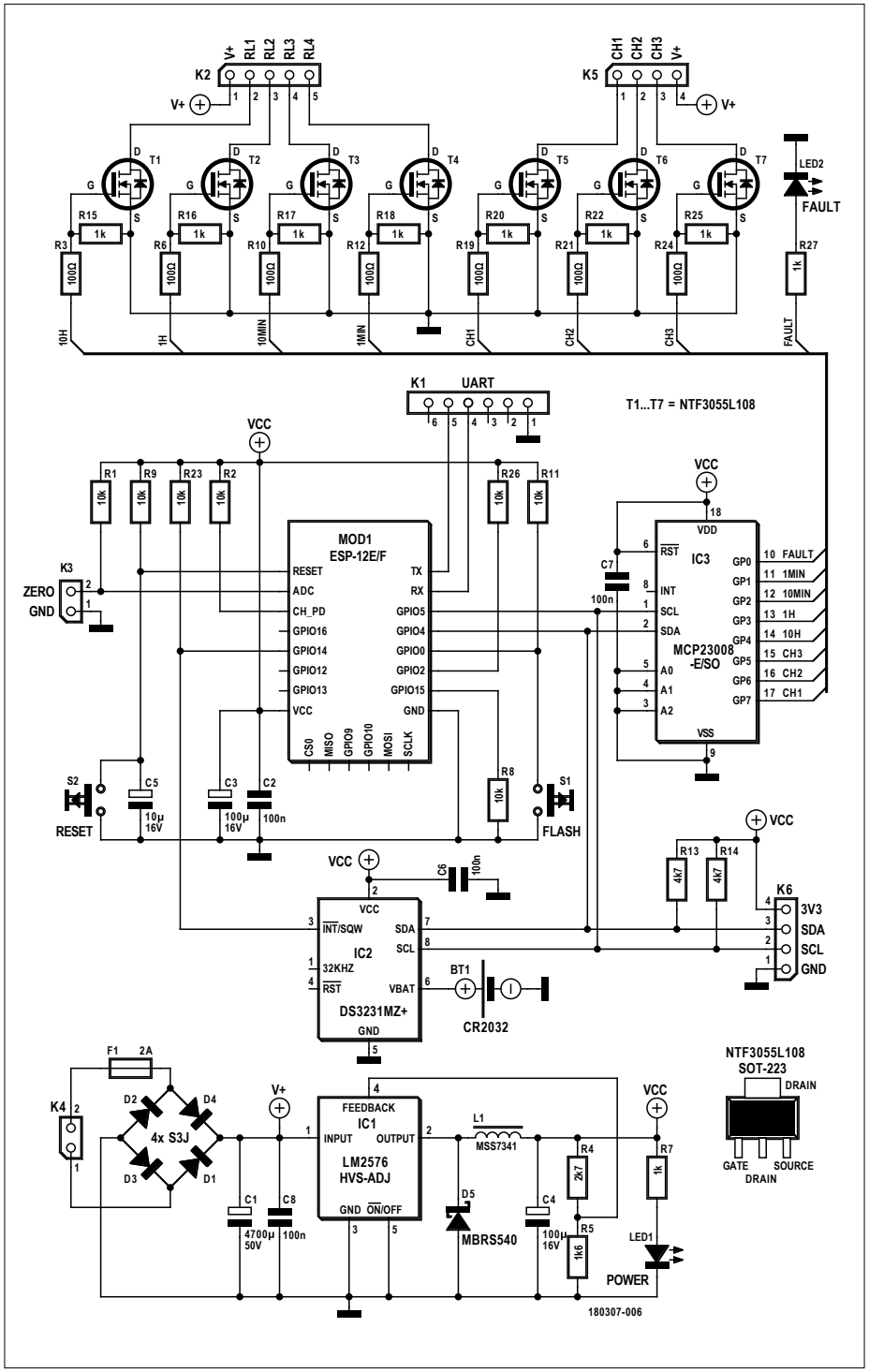


Figure 1: Schematic of the Pinball Clock.
With or without the chimes, this an **acoustically alarming** circuit!

Let’s go electronic

The schematic of the Pinball Clock shown in **Figure 1** isn’t too complicated, but let’s explain all sections in more detail anyway.

The power supply of our clock is rather straightforward. A 2×12 V toroidal power transformer with its secondary windings in series is connected to K4, resulting in about 32 VDC of coil voltage at the positive terminal of bridge rectifier D1–D4. That is… in theory, in practice the unloaded transformer secondaries will present a much higher output voltage, resulting in a coil voltage of over 35 VDC. If you have Williams or Gottlieb score reels, a 2×9 V transformer will do.

IC1 is a LM2576 stepdown converter that supplies the 3.3 V for the digital section of the hardware for the Pinball Clock. To

be on the safe side, we used an HV version with a maximum input voltage of 60 V. 'Regular' LM2576s have an absolute maximum rating of 40 V at the input. LED1 will light when the 3.3 V power supply is switched on.

One note about electrolytic capacitor C1, one of the few through-hole electronic components in this project. It may look a bit odd on the PCB, but this was a deliberate choice. First: it's difficult – if not impossible – to find a large SMD electrolytic capacitor with a voltage rating upwards of 35 V while through-hole versions are all over the place. Second, for the sake of mechanical stability a through-hole capacitor is to be preferred and (most important!) readily available with a maximum voltage rating of 50 V (and even higher).

The coils of the score reels and chimes are controlled by NTF3055 N-channel MOSFETs, which have a sufficiently low Vth to reliably pulse the coils with 3.3 V logic outputs. In our first prototype these FETs were directly controlled by the ESP8266's GPIO pins. This proved to be a bad choice: the ESP8266 doesn't have many IO's and some of them have special functions during power-up, making it very tricky to keep the coils completely switched off during startup. IC3, an MCP2300 I2C I/O expander, made the hardware a lot easier to control and it also offers a kind of protection/buffering between the ESP and the power FETs.

In the first design the four 'zero' switches were connected to the ESP's ADC to save IO pins. Although this ADC is not really accurate, this solution proved to be reliable enough to keep it this way. The ESP will only detect a change in ADC value in this application, it can only determine if a zero switch opens after the coil of a score reel is pulsed, i.e. when this reel has reached its zero position. Consequently, during a complete reset all reels will turn, even if they were already at position zero.

A 1.2-kΩ resistor should be soldered across all four 'zero' switches (refer ahead to Figure 2). All these switches are in series and connected to K3.

If there is no change in ADC value after 12 successive pulses on one reel, 'FAULT' LED2 will light, and pulsing this particular coil will be suspended until the clock is reset, but first check the zero switches and their wiring before you do so. This feature is not only implemented to protect your nerves, but also to protect the coil from heating up by being pulsed to reach a 'zero' position that will never register.

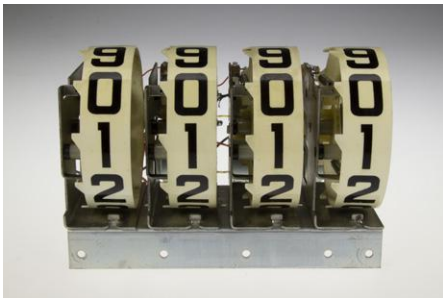
An I²C OLED display can be connected to K6 to display clock status information of the clock. This connector can also be used for future developments, like adding an I2C temperature sensor.

S1 is the Reset pushbutton. If you keep S2 depressed until S1 is released, the ESP8266 will start in bootloader mode, enabling the user to reprogram the clock via UART connector K1. S2 can also revert the clock to default (factory, ha-ha) settings, this can be useful if the clock is to be connected to a different Wi-Fi network. To do so, reset the clock and press S2 while the red FAULT LED is on.

IC2 is the highly accurate (wow, 5 ppm!) DS3231 I2C Real Time Clock. A common CR2032 coin cell battery will maintain the correct time with the pinball clock switched off. So, no need to adjust the clock manually every time after power up if it isn't synchronised via NTP.

Building & wiring the Pinball Clock

First you'll need four score reels, be sure you have them before you buy any



other hardware for this project. Three-tone chimes, or maybe a single pinball bell would be a bonus, but these are not mandatory to build the clock and even if you have them, chances are pretty high you'll disconnect them soon them to keep that noise down.

Although most parts on the PCB are SMDs, soldering will not be too difficult if you use a small soldering iron and thin (preferably <1 mm) soldering wire. Solder one pad of each component first and check if all other connections are well aligned with the PCB and double-check the orientation/polarity of the semiconductor parts and electrolytic capacitors before you solder the other parts.

Now it's time to connect the parts and units that are not on the PCB — refer to the wiring diagram in **Figure 2**.

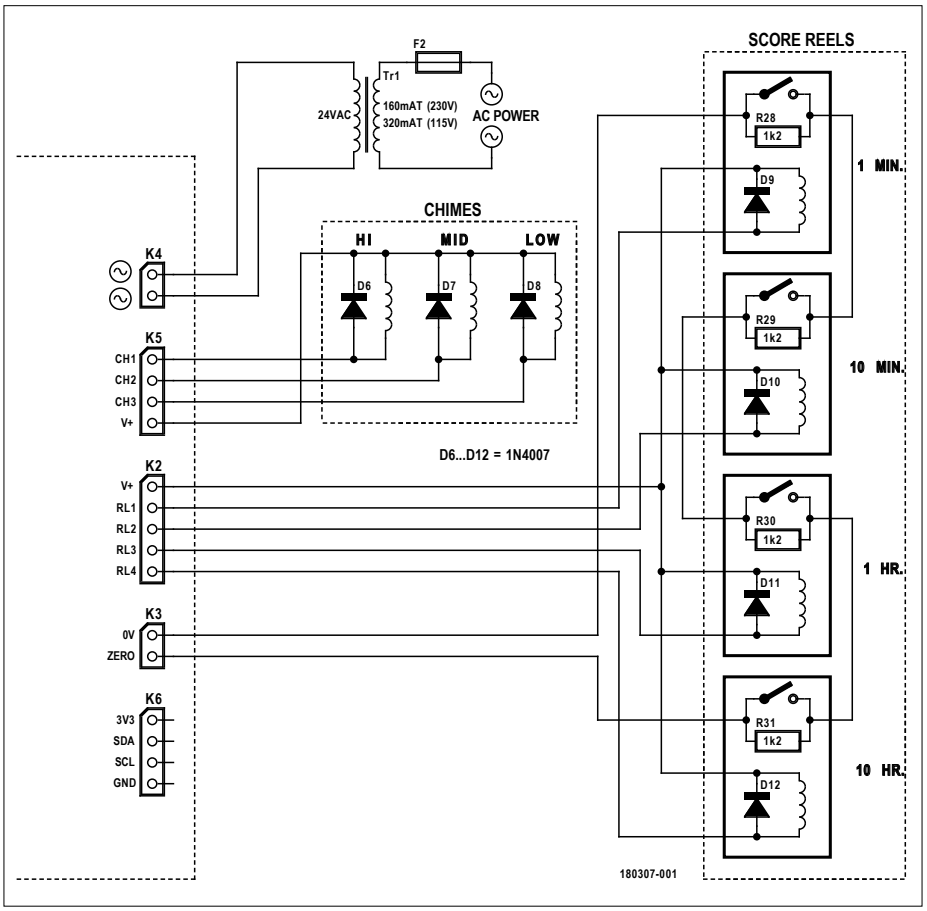


Figure 2: Pinball Clock wiring diagram.

COMPONENT LIST

Order this project at GroupGets.com: www.elektormagazine.com/pinball-clock

Resistors

R1,R2,R8,R9,R11,R23,R26 = 10kΩ, thick film, 5%, 0.1W, 150V
R3,R6,R10,R12,R19,R21,R24 = 100Ω, thick film, 5%, 0.1W, 150V
R4 = 2.7kΩ, thick film, 5%, 0.1W, 150V
R5 = 1k6Ω, thick film, 1%, 0.1W, 150V
R7,R15,R16,R17,R18,R20,R22,R25,R27 = 1kΩ, thick film, 5%, 0.1W, 150V
R13,R14 = 4.7kΩ, thick film, 5%, 0.1W, 150V

Capacitors

C1 = 4700µF, 50V, 10mm pitch, 22x41mm
C2,C6,C7,C8 = 100nF, 50V, X7R, 0805
C3,C4 = 100µF, 16V, 2312
C5 = 10µF, 16V, 1206

Inductor

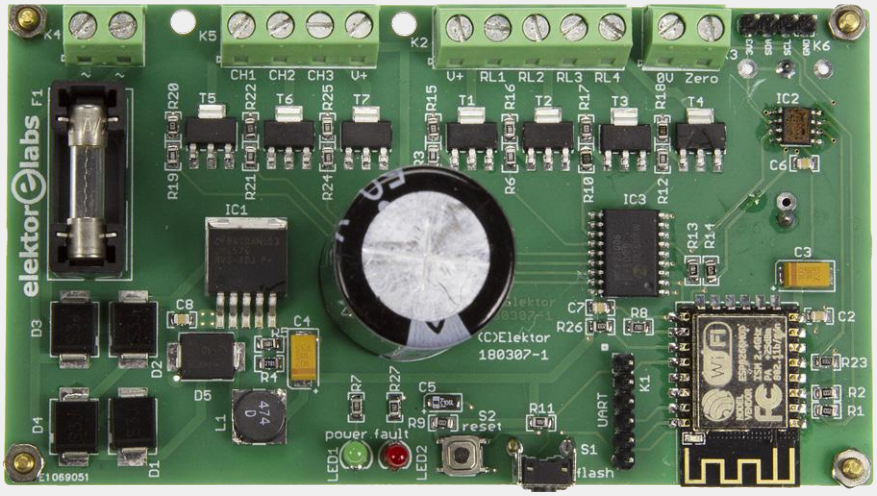
L1 = 470µH power Inductor (SMD), 570mA, shielded, 270mA, MSS7341T series

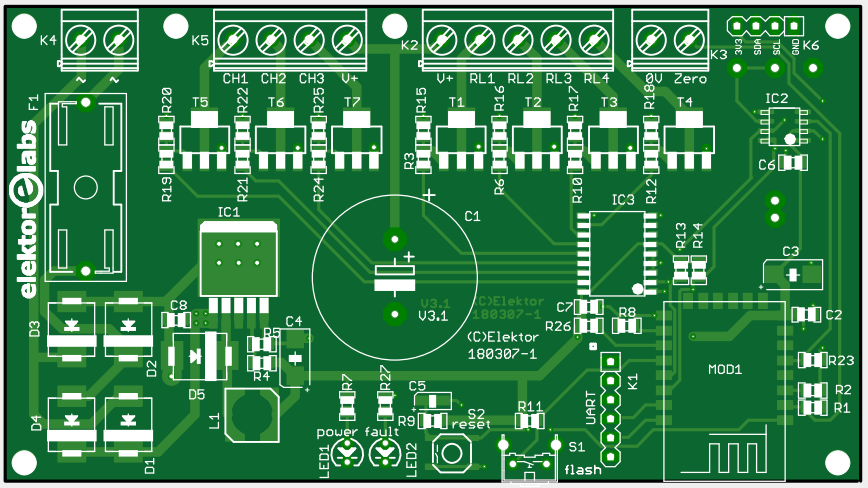
Semiconductors

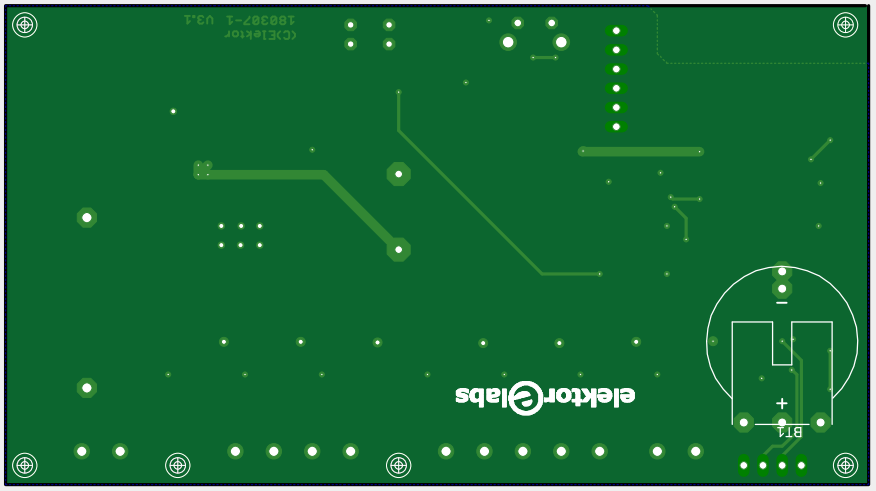
D1,D2,D3,D4 = S3J+, 600V, 3A
D5 = MBRS540, 40V, 5A, V_f=550mV @ I_f=5A
LED1 = green, 3mm
LED2 = red, 3mm
T1,T2,T3,T4,T5,T6,T7 = NTF3055L108T1G MOSFET, n-channel, 3A, 60V, 120mΩ
IC1 = LM2576HVS-ADJ, step-down regulator, 4-60V, 3A
IC2 = DS3231MZ+ RTC, I²C, 5ppm
IC3 = MCP23008-E/SO, 8-bit I²C I/O expander
MOD1 = ESP-12F, ESP8266-based Wi-Fi module

Miscellaneous and/or not on PCB

Bt1 = CR2032 battery, with holder, e.g. Keystone 1066
D6,D7,D8,D9,D10,D11,D12 = 1N4007, 1000V, 1A (not on PCB!)
F1 = 2AT fuse, 5x20mm, with holder, PCB mount, pitch 22.6mm
F2 = 160mAT fuse, 5x20 mm (230V mains) with holder, panel mount, solder terminals
F2 = 320mAT fuse, 5x20 mm (115V mains) with holder, panel mount, solder terminals
K1 = 6-pin pinheader, 0.1" pitch, vertical
K2 = 5-way PCB screw terminal block, 0.2" pitch, 630V
K3,K4 = 2-way PCB screw terminal block, 0.2" pitch, 630V
K5 = 4-way PCB screw terminal block, 0.2" pitch, 630V
K6 = 4-pin pinheader, 0.1" pitch, vertical
LCD1 = 0.96" Inch 128x64 OLED LCD Display for Arduino (optional)
R28,R29,R30,R31 = 1.2 kΩ, carbon film, 5%, 0.25W, 250V (not on PCB!)
S1 = switch, tactile feedback, e.g. SKHHLVA010
S2 = switch, tactile feedback, 12V, 50mA, Multicomp TM series
Tr1 = transformer, toroidal, 2x12V, 30VA, e.g. Multicomp MCTA030/12
Power entry connector, IEC320 C8 + cable
PCB 180307-1 V3.1







Clean the score drums and do not forget to install a type 1N4007 (or equivalent) flyback diode on each coil (D9 to D12), and the same goes for the optional chime solenoid(s). Connect the coil tabs at the cathodes of the diodes to the V+ connection on the PCB (at K2 for the reels, K5 for the chimes). The other side of the rightmost score drum coil (minutes) should be wired to RL1, the second (10 minutes coil) to RL2 on K2 and so on, the highest note of the chimes to CH1 on K5, etcetera.
The resistors for score reel zero detection (R28 to R31) are soldered directly to the tabs of the zero position switches of the score drum units. Next, these switches must be daisy-chained and the ends of this chain wired to K3.
The secondary windings of the power transformer are connected to K4. Don't forget to install an AC powerline fuse at the primary side of the transformer.

Software, and firmware flashing

There's quite some work to do before you can start compiling and uploading the code to the clock's ESP12 module.
First, you need to have the Arduino IDE installed on your computer, plus its ESP8266 boards plugin [4]. Second, you need the arduino-esp8266fs-plugin to upload the clock's web content [5]. Download the firmware for this project from [6]. To compile this code you need to have the following libraries installed:

- RTCLib by Adafruit;
- MCP23008 by Adafruit;
- TimelLib by Michael Margolis;
- Arduino JSON 5.x;
- WebSockets by Markus Sattler;
- NtpClientLib form [7] (use the AsyncUDP branch);
- CRC32 by Christopher Baker.

To make life a bit easier for you, we have included our Arduino 'Portable' folder in the download archive for this project [8], which contains the complete sketch plus all these libraries, including the ESP8266FS-plugin. To keep the size of our download down, we did **not** include the ESP8266 boards plugin! Copy this entire portable folder into the Arduino installation folder on your computer.

- Power up the clock and connect a 3.3-V FTDI cable to K1 (the black GND wire pointing towards IC3) and a USB port on your PC. Keep pushbutton S1 (Flash) depressed, press and release reset switch S2 and then release S1 to start the ESP in bootloader mode.
- In the Arduino IDE, select the COM port that has the FTDI cable attached and choose "nodeMCU 1.0 (ESP-12E Module)" as the target board.
- Within the Arduino IDE, open PinballClock.INO located in this folder:
- ..\arduino\portable\sketchbook\PinballClock
- then compile and upload the firmware to your board.

Last but not least, we must upload the ESP webpage. Inside the Arduino sketch folder resides a subfolder called 'data' which contains all files for this webpage. The upload tool will automatically pick all content from this folder and send it to the ESP's flash memory. First verify that 'Flash size' in the Arduino IDE's Tools menu is set to '4M (1M SPIFFS)'. Repeat the S1/S2 sequence to put the ESP in Bootloader mode and then upload the webpage of the ESP by selecting 'ESP8266 Sketch Data Upload' in the Tools menu. This will take a while to complete, but when the status bar in the IDE shows 'SPIFFS Image Uploaded' the clock is ready for configuration.

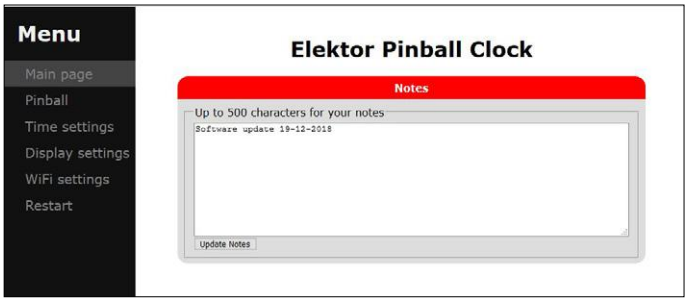


Figure 3: This webpage allows all Pinball Clock parameters and options to be set.

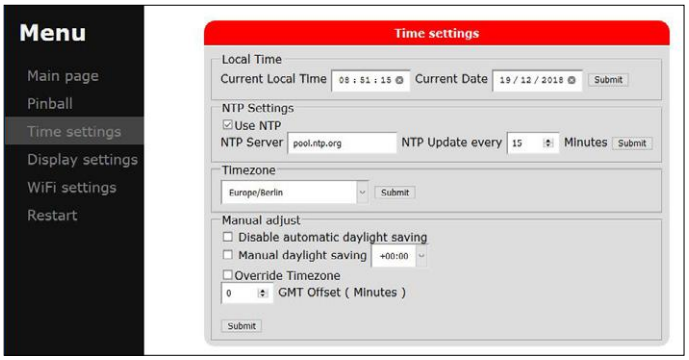


Figure 4: Time settings menu.

JACKPOT! Elektor Pinball Clock hits hackaday: https://bit.ly/2Rw9xln

Operation

After the upload of the webpage the clock will reset again, driving all four reels to their zero positions and then show some random time read from the RTC (which is not synchronised yet). Observe the red Fault LED (LED2) and if it's lit, check and fix the four zero switches on the score reels, the resistors and their wiring and reset the clock by pressing S1.
If the clock can't connect to a Wi-Fi network (like after flashing new firmware, or when the network is down), it will set up an open Wi-Fi access point with the SSID 'Elektor Pinball Clock'. Connect your computer, tablet or smartphone to this network, open your browser and search for IP address 192.168.4.1, which will open the clock's webpage (**Figure 3**). There's a text box that can be used as a notepad, e.g. to enter your High Score of the pinball game, or for special notes if you make your own modifications to the firmware.
You can set the time and date manually under menu item 'Time settings' (**Figure 4**), but you can also have it synchronised

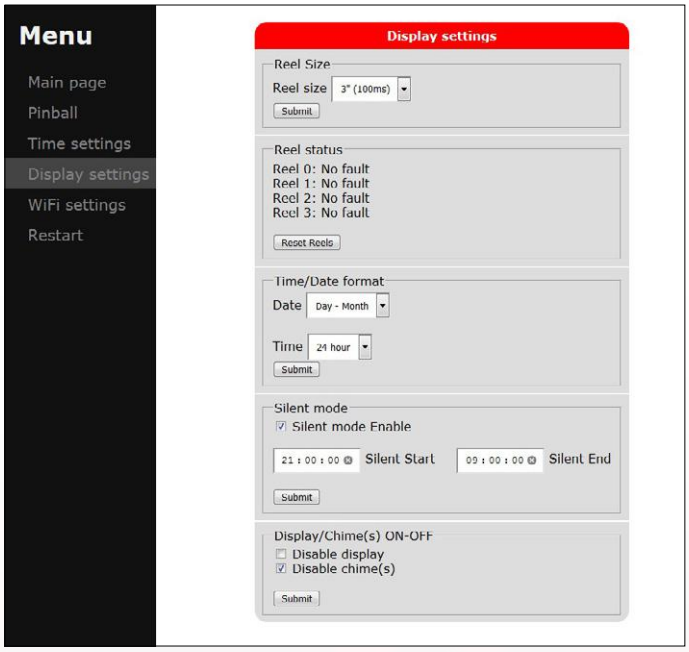


Figure 5: Display settings menu.

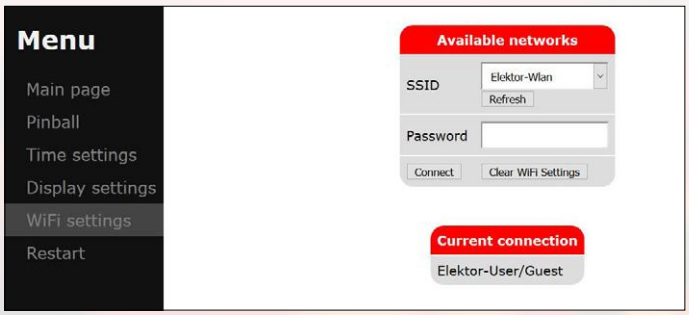


Figure 6: Wi-Fi settings menu.

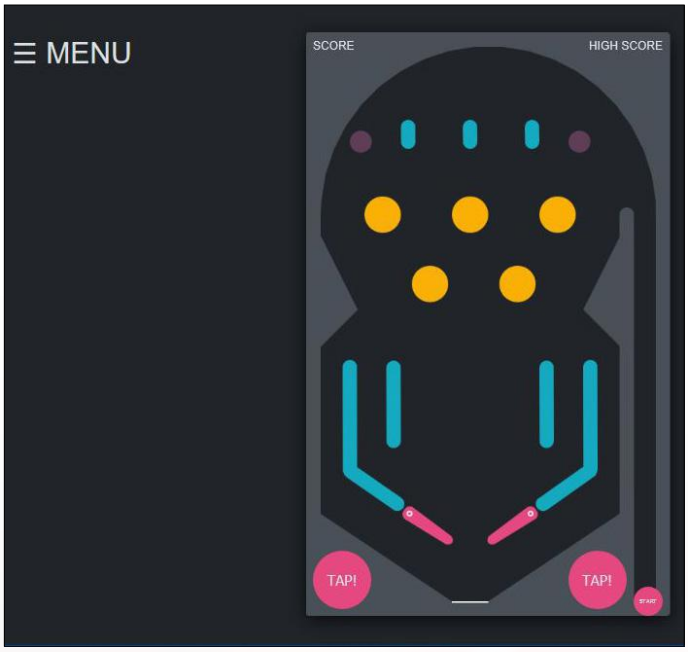


Figure 7: Play the Pinball Game 'Easter Egg' available in the control software and let the clock digits register your score!

TILT!! (.IT) Elektor Pinball Clock hits Italy: https://bit.ly/2CRGm3O

via an NTP server on the Internet. In the latter case, first look up the SSID of your local network in the list in 'WiFi Settings', enter the password and click on 'submit'. The ESP will reset and connect to this Wi-Fi network, receiving an IP address from the network's DHCP server. There are many options to find out what this address is, like:

- with the FTDI still attached between clock and computer, open the serial monitor of the Arduino IDE to see if the logon is successful and what the clock'sIP-address is;
- alternatively, connect the optional OLED-display LCD1 to view the network settings after reboot;
- or just look on your network (e.g. via Windows network map, or tools like Free IP Scanner from) to see which IP address gets added when the clock is switched on.

Now you can access the clock via your local Wi-Fi, using the IP address you just figured out. Don't forget to connect your computer or mobile device to this same network; the AP of the clock will be switched off when it has logged on to your WiFi. Now open 'Time Settings' again and enter 'pool.ntp.org' (or any other server you prefer to use for synchronisation) for the NTP server, and the interval at which the pinball clock is synchronised. Select your time zone and additional time settings and don't forget to click the corresponding 'submit' buttons to update the clock settings.

Settings, don't you love them? Here are few.

- 'Display settings' (**Figure 5**) allows you to adjust the timing (i.e. pace) of the score reel pulses. It's not absolutely necessary to change the default setting as all types and brands will work with the default 5" parameters. However, the smaller 3" ones were designed for faster stepping (keeping up with scoring) and it looks and sounds more natural to speed them up. The larger (older) score drums will not work reliably with faster settings.



@ WWW.ELEKTOR.COM

→ **Pinball Clock PCB, unpopulated: 180307-1**
www.elektor.com/pinball-clock-180307-1

→ **ESP-12F, ESP8266-based Wi-Fi module**
www.elektor.com/wi-fi-module-160100-92

→ **(optional) 0.96" inch 128x64 OLED LCD Display for Arduino**
www.elektor.com/oled-display-for-arduino

EXTRA BALLS!
Free pinball game stashed in the clock control app



- Six-inch reels never really existed (as far as we know) but were added to this menu to give you more options; it's up to you to choose what you like most.
- 'Reel status' shows if the zero switches of the reels are registering and can be used to check (re-) adjustment of these switches. The 'Reset reels' button will spin all four units to '0' as well as clear reel error messages (if present and if solved, naturally).
- 'Time/Date display' allows you to change the format of both time (12h or 24h) and date (mm-dd or dd-mm) display.
- 'Silent mode' can be used to automatically mute the clock (reels and chimes) during a specific period of the day, which will most probably be during the time you normally sleep. The last options in the 'Display settings' allow you to switch the score reels and/or chimes off, no matter what time it is. If the display is switched off, the reels will be set to '9999'.
- 'WiFi settings' we've seen before (**Figure 6**), but now the clock is connected to your Wi-Fi network it shows the available networks and the name of the network the clock is using under 'Current network settings'. The clock will remember your preferred network and its password even with the power switched off.
- 'Restart' does what it says: it restarts the clock, note that this may change the clock's IP address.

And now for the fun part, it's the top item in the clock's menu list but we saved it for desert: the pinball game (**Figure 7**). A simple pinball game will appear on your screen when you select this item, the score reels will reset and do the score-keeping, of course the chimes will be active too! Please note that the game will respect the silent mode of the clock, so you cannot use the game to wake up unsuspecting housemates… ⏮

180307-01

Web links

[1] Thermometer using Giant Gottlieb Displays (Elektor magazine April 2012): www.elektormagazine.com/magazine/elektor-201204/19852

[2] Same PCB Shoots Again! (Elektor magazine July 2012): www.elektormagazine.com/magazine/elektor-201207/19947

[3] AC coils on DC power: www.ecmweb.com/content/using-ac-coils-dc-power

[4] ESP8266 boards plugin: https://github.com/esp8266/Arduino

[5] Arduino esp8266fs plugin: https://github.com/esp8266/arduino-esp8266fs-plugin

[6] Pinball Clock Software bundle for the project (Github): https://github.com/ElektorLabs/180307-PinballClock

[7] NtpClientLib form: https://github.com/gmag11/NtpClient/issues/60

[8] Pinball Clock Software bundle for the project (ElektorLabs magazine site): www.elektormagazine.com/180307-01

[9] Pinball Clock video (Elektor TV): https://youtu.be/lQL2Vk1Hj-o

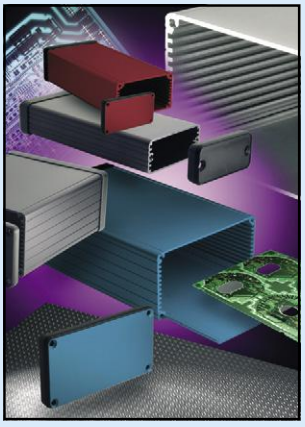
Advertisement

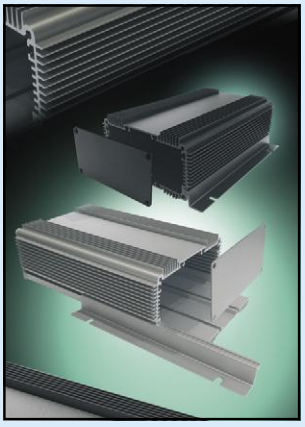
HAMMOND MANUFACTURING®

Extruded enclosures standard and heatsink

www.hammondmfg.com/1455.htm
www.hammondmfg.com/1455NHD.htm

01256 812812
sales@hammond-electronics.co.uk





The Elektor Community

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, ESP32, IoT, 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.elektor.tv

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

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/ElektorLabs

twitter.com/Elektor

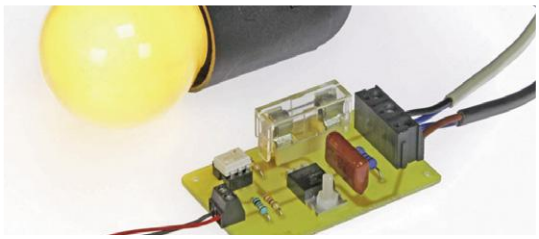
instagram.com/elektorlabs

Elektor Labs Pipeline

Timers and clocks are popular subjects for electronics projects. Everybody needs them and they leave plenty of room for creativity, especially so when it comes to displaying time or a timer.

Get rich & famous, post your projects @ Elektor Labs

Electrically Safe LED-to-Lamp Converter
These days it's common practice to replace incandescent light bulbs with LEDs, but what if you want to do the opposite? With this converter you can. Simply replace the LED by an optocoupler to enable the latter to switch the lamp on and off.



@ Elektor Labs: www.elektor-labs.com/1704

Build a Bedroom Clock Showing Outside Temperature Too
If you're keen on knowing how cold it is outside before getting out of bed... then this DIY clock is for you. A Raspberry Pi running a home automation server, an ESP32 Wi-Fi module with a DHT22 sensor attached to it, and another ESP32 module driving an LED display is all you need. Sounds complicated? Welcome to 2019.



@ Elektor Labs: www.elektor-labs.com/1605

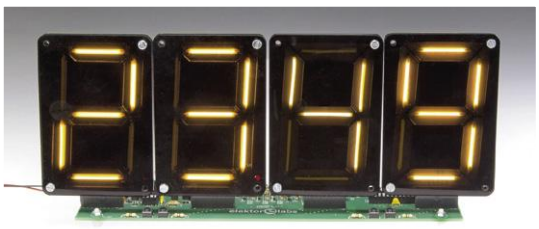
Build a Non-programmable Timer with 0 (zero) Features (a.k.a. hourglass)
'Feature creep' is the phenomenon of adding features to a product or a program that are not necessary for operating the product or program. Examples are colour pickers and font selection options in smartphone apps, and all sorts of other obscure configurable parameters. This little timer does not suffer from feature creep... or does it?



@ Elektor Labs: www.elektor-labs.com/478

Build a Clock/Scoreboard/Timer with XL Digits
If you ever need a large display to keep the score for games, or a counter for sports events, this is the thing to build. With its four tall (10-cm / 4-inch) 7-segment digits it can be read easily from afar. The project is based on LEDitron modules, i.e. 7-segment displays built from LED filaments as found in modern lamps. And sure, it doubles as a clock!

(180571-B-01)



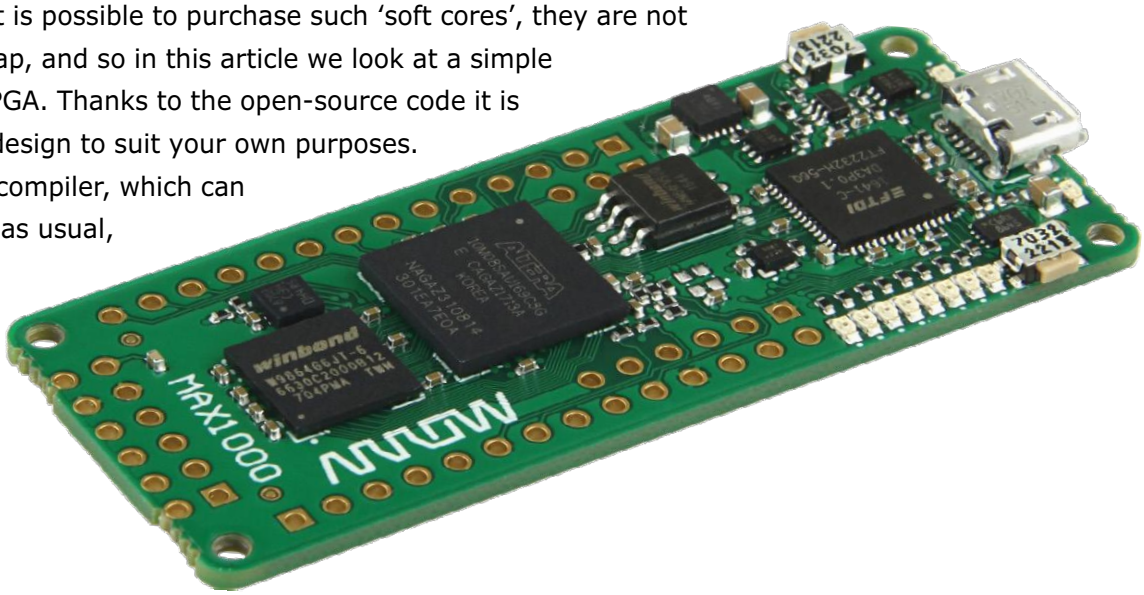
@ Elektor Labs: www.elektor-labs.com/1299

The SCCC Project (1)

Homebrew soft-core processor and C compiler

By **Martin Oßmann** (Germany)

Sometimes within a single project there is a requirement for both an FPGA and a classical CPU, so that complex program flows can be implemented simply. One solution in such situations is a processor realized within an FPGA. While it is possible to purchase such 'soft cores', they are not usually particularly cheap, and so in this article we look at a simple homebrew CPU in an FPGA. Thanks to the open-source code it is possible to modify the design to suit your own purposes. To this we add a DIY C compiler, which can also be extended. And, as usual, the series of articles describing the project will expand on the theory by including as many practical examples as possible.



If you have worked with both FPGAs and normal CPUs you will have come to realize that FPGAs hold the upper hand when a large degree of parallelism is called for, but the flow of control in many FPGA projects is comparatively straightforward. If a more complex flow of control is required, then classical CPUs have the advantage.

In many projects we therefore find both an FPGA and a microcontroller. An alternative to this is to realize a CPU in the FPGA itself. Soft cores like this are often provided by the manufacturers of FPGAs, but generally these take the form of proprietary software that can sometimes be rather expensive.

This is where the Small C Compiler CPU Project (SCCCP) comes in. This provides a simple CPU which, since its design is open source, can be modified to suit particular requirements. Alongside this we require a language in which the CPU can be programmed, and here we have a compiler that can understand a subset of the C programming language based on integer variables called 'Small-C'. The compiler can also be expanded.

The MAX1000 board.

Intel MAX®10 10M08SAU169C8G FPGA

- Arrow USB Programmer2
- 64 Mbit SDRAM (16 bit data bus)
- 64 Mbit flash
- 12 MHz MEMS oscillator
- optional MEMS oscillator
- 8 red user LEDs
- 2 indicator LEDs
- 2 user buttons
- 3-axis accelerometer
- footprint for 12-pin PMOD header
- footprint for Arduino MKR header
- footprint for JTAG header
- footprint for 3-pin I/O header

Hardware

The hardware for this project takes the form of the Arrow MAX1000 board, available for around US$30 (£25) [1]. The board comes with an Intel/Altera MAX10 series FPGA, a USB programming interface, two user buttons, eight user LEDs and a range of expansion connectors (see text box).

In order to gain familiarity with the board and the Intel/Altera

Quartus software, we can try a first experiment. The document at [2] is the MAX1000 User Guide: this includes a description of how to install the Quartus development environment. The development environment is used to create a programming file (a process called 'synthesis') which can be uploaded to the FPGA to configure the behaviour of the component's internal circuitry. The hardware description language that we shall be using in this project is Verilog: the previous issue of Elektor included an article covering the basics of this language [3].

Experiment 1: Test

The main purpose of experiment 1 is to test the Quartus installation. Download the software archive from the web page accompanying this article [4], which includes various demonstrations of the SCCC project. Unpack the archive and place the results in a local directory C:/SCCCP created specially for this project. Now simply use Quartus to open the QPF (Quartus Project File) for experiment 1: *C:/SCCCP/experiment1/experiment1/experiment1.qpf*. Then synthesize the project and upload the programming file thus created to the MAX100 board. If all works as it should you will see a running light pattern on the user LEDs. We have just synthesized our homebrew processor for the first time and made it run on the FPGA: the C program is sitting in the experiment 1 directory in ready-compiled form. The corresponding source code can be seen in **Listing 1**.

To take a peek behind the curtain to see what is going on, look at the contents of the experiment 1 directory. In there you will find files called codemem.txt and datamem.txt, which were created by the author's C compiler. When the programming file is uploaded to the FPGA the contents of these files are also transferred into the memory of the FPGA. The contents of codemem.txt define the initial state of the processor's code memory (which contains instructions in machine language), while datamem.txt contains the initial state of the processor's data memory (see below).

Looking inside these files it will become apparent that they include their own source code at the start, which is convenient if you subsequently wish to inspect the source that gave rise to them.

The working directory also contains the file `sCCCPcpu1v01.v`. This is the Verilog source for our CPU, running to around a thousand lines of code.

Figure 1 illustrates the work flow, with the various tools and files involved.

On the author's computer the synthesis process takes around four minutes. That means that debugging is a rather tedious and time-consuming activity: there must indeed be a better way, as we will shortly see.

A bit of history

Designing a CPU along with a suitable compiler is not exactly the kind of project you are likely to knock off in a spare afternoon. It is therefore a good idea to start with something reasonably manageable. The inspiration for the SCCCP comes from the 'Small-C' project by Ron Cain (Ron Cain, 'A Small C Compiler for the 8080's', *Dr. Dobb's Journal*, April–May 1980, pp. 5–19). Cain developed a small but nevertheless very powerful subset of the C language, and implemented code generation for both the 8080 and the Z80. Only a small number of different 8080 instructions were generated, and a corresponding 'cut down' CPU is therefore reasonably straightforward to implement as

Listing 1. Running Lights.

```
setLeds(int pattern){   // output pattern to LED
 pattern ;
#asm
 OUTA 5         // LEDs are attached to channel 5
#endasm
}

delay(int time){
 int k ;
 for(k=0 ; k<time ; k++){
  }
}


main(){
 int k ;
 while(1){
  for(k=0 ; k<8 ; k++){
    setLeds(1 << k) ;
    delay(500000) ;
  }
 }
}
```

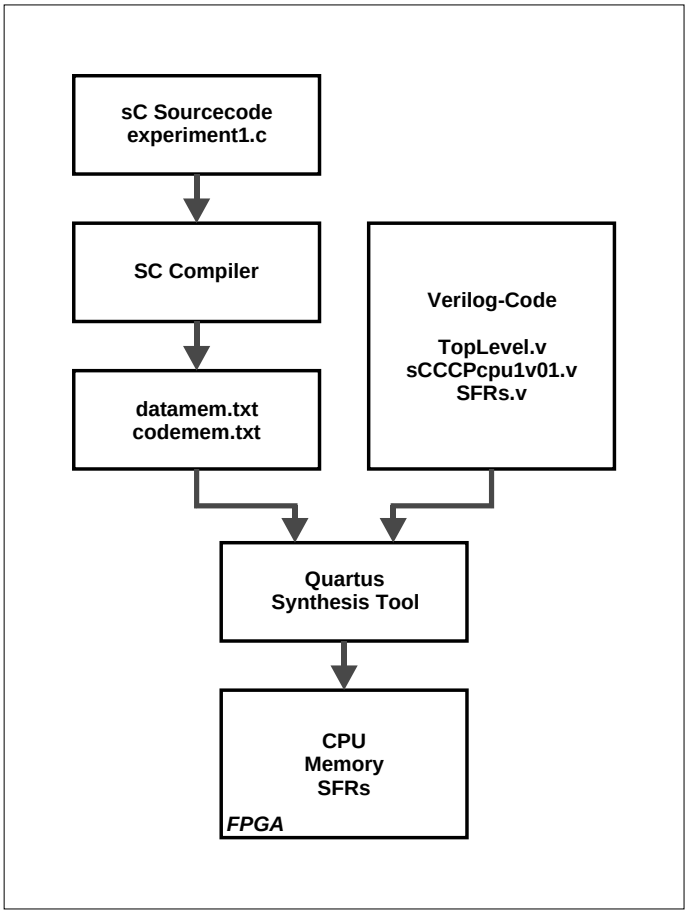


Figure 1. The C compiler converts the C source code into machine language and generates the file codemem.txt. This image file is loaded into the RAM of the FPGA, from where it can be executed by the synthesized CPU.

Listing 2. Running Lights with interrupt handler.

```
#asm
RTCintVector        EQU   8
CPUflagsChannel     EQU   6
LEDsChannel         EQU   5
 ORG       RTCintVector
 PUSH      R0
 PUSH      R1
 CALL      RTCinterrupt
 POP       R1
 POP       R0
 RET
#endasm

setCPUflags(int x) {
 x ;
#asm
 OUTA   CPUflagsChannel
#endasm
}

setLeds(int pattern){
 pattern ;
#asm
 OUTA   LEDsChannel
#endasm
}

RTCinterrupt(){
 setLeds(0xFF) ;
}


main(){
 int k ;
 setCPUflags(1) ;
 while(1){
  for(k=0 ; k<8 ; k++){
     setLeds(1 << k) ;
  }
 }
}
```

a soft core. Our SCCCP CPU has been designed in such a way that it can handle the complete set of machine code instructions that can result from compiling Small-C code. If you can still remember the good old 8080, you will immediately see similarities between it and our CPU.

CPU architecture

Our SCCCP CPU uses what is called a modified Harvard architecture, which means that program and data memories are separate. In its standard configuration there is room for 4096 instruction words and 1024 data words. The CPU also has a program counter (or PC) and a stack pointer register (SP). These two registers are sufficiently wide that they can address all memory words available to them: the program counter is thus 12 bits wide, the stack pointer 10 bits.
There are two further registers: R0 is the primary register through which most data are transferred. Register R1 is the secondary register, which is used whenever an operation requires two operands. R0 corresponds to the HL register pair in Ron Cain's Small-C implementation, and R1 to the DE register pair. These two registers are 32 bits wide.
The data memory is also 32 bits wide, while the program memory is 40 bits wide. The latter allows us to have an instruction word with 8 bits encoding the opcode and 32 bits to encode a data word.
Now we can look at one of the CPU's instructions. The instruction

```
LDI 1234 ; LoaD Immediate
```

loads the value 1234 into register R0.
Our machine understands only 32-bit words and all operations are carried out in 32-bit arithmetic. If we wish to deal with char types, then only the bottom 8 bits are used. This may seem at first glance wasteful, but it makes a huge difference to the simplicity of developing a compiler.

Peripherals

The CPU also has the capability to access values from other parts of the hardware design, which in each case will be 32 bits wide.
If, for example, the CPU is extended by adding a real-time counter (RTC), then we would want to be able to read and write the counter value, and possibly change the RTC period. In this case there are two values (RTCtimer, RTCtop) that can be written from the CPU, and one value (the current counter value RTCtimer) that can be read by the CPU.
Each parameter in the peripherals that is to be readable or writable is assigned a number called the SFR (special function register) address. To read a value we have the instruction

```
INPA nn
```

which reads a value from special function register nn and writes it to register R0. Likewise, the command

```
OUTA nn
```

sends the contents of R0 to special function register nn.
In our experiments we will expand the bare CPU with a range of additional functions, including a serial interface, analogue-to-digital converter, pulsewidth modulator, RTC and interrupts. The CPU can control these peripherals via the corresponding SFRs using the INPA and OUTA instructions.
Suppose for example we wish to access the RTC to set its value or read its current state: in this case the SFR address is 4. The period setting RTCtop is available at SFR address 7. So, to set the RTC to 256 and the period to 500, we can write the following assembler code.

```
LDI 256 // load constant into register R0...
OUTA 4 // ... and write it to SFR address 4 (RTCtimer)
LDI 500 // load constant into register R0...
OUTA 7 // ... and write it to SFR address 7 (RTCtop)
```

Other SFR addresses allow access to the I/O pins of the FPGA, which in turn are connected to peripherals on the board such as the user LEDs.

Debugging the CPU design

Debugging a CPU in Verilog takes a long time, as each debug cycle requires resynthesizing the design completely from scratch. To hunt down bugs in the Verilog code for the CPU (and any extensions you make to it) it is a good idea to use a Verilog simulator instead. Here we will use the free simulator Icarus Verilog [5]. A guide to installing this simulator can be found at [6].

Before we can progress to experiment 2, we need to modify the batch files c.bat and r.bat in the experiment 2 directory so that the invocation of the Icarus Verilog simulator is correct. On the author's system the Icarus .EXE files can be found in the directory C:/iverilog/bin/. The corresponding line in the file c.bat (here the 'c' stands for 'compile') is then

```
c:/iverilog/bin/iverilog.exe -o Vobject TestBench1v01.v
```

and likewise the batch file r.bat ('r' for 'run') is as follows.

```
c:/iverilog/bin/vvp.exe Vobject
```

Both c.bat and r.bat must contain the correct path to the Icarus Verilog executables.

The SCCCP environment

In the experiment 1 directory you will find the application program for our CPU in ready-compiled form in the file codemem.txt: this file is an image of the program memory. However, in general users will want to be able to write their own programs in C, compile them, and then execute them on the CPU. For this purpose the author has produced a simple development environment written in the Java programming language. It can be invoked from the command line as follows.

```
java -jar c:/SCCCP/CompilerAsJar/sCCCP.jar -s experiment2.c -i c:/SCCCP/experiment2/experiment2/
```

With the parameters shown above the C source code file experiment2.c will be loaded into the development environment at start-up.

To save readers too much error-prone typing, the invocation is implemented in the form of a batch file, which can be found in the experiment 2 directory under the name c_compile.bat. The GUI of the simple development environment is illustrated in **Figure 2**. The 'MAKE' button causes the C program to be compiled. In addition there are options for debugging and simulation, to which we shall return later in this article.

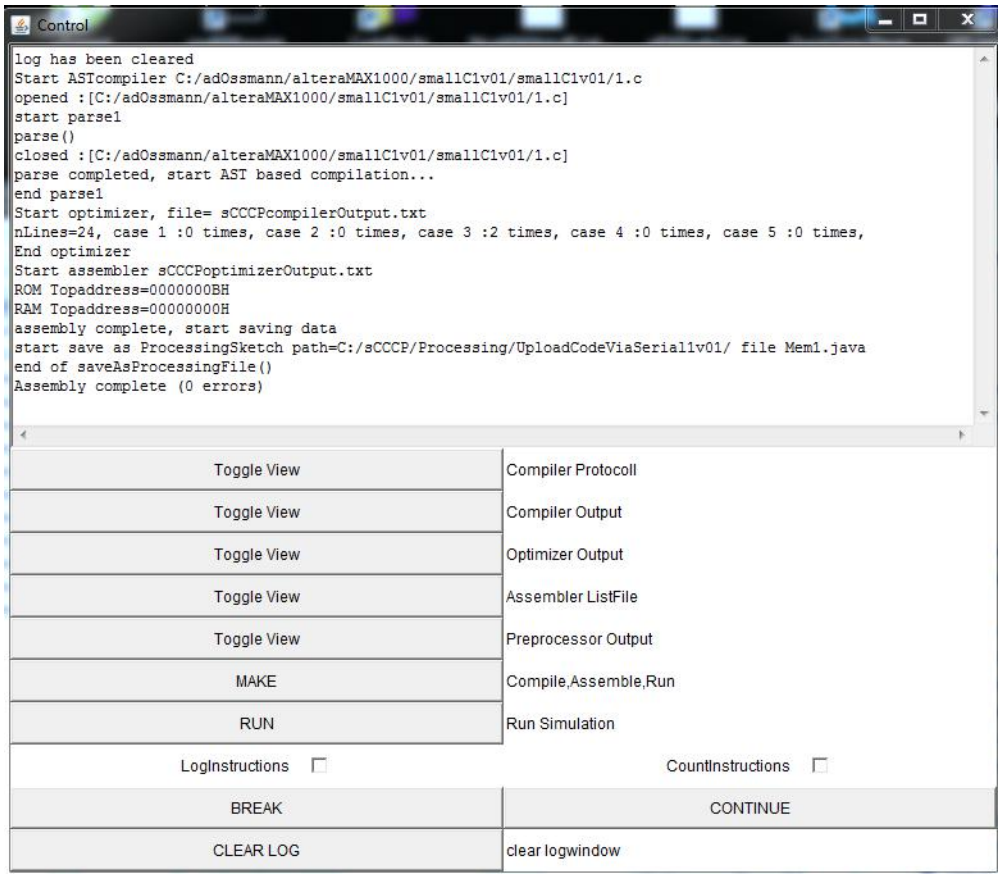


Figure 2. The simple development environment, including compiler and simulator, is written in Java.

Listing 3. Debug output.

```
VCD info: dumpfile test1.vcd opened for output.
sC reset
sC reset
 -- set CPUflags 00000001H
 -- out channel 00000005H data = 00000001H
 -- out channel 00000005H data = 00000002H
 -- out channel 00000005H data = 00000004H
 -- out channel 00000005H data = 00000008H
 -- out channel 00000005H data = 00000010H
//
Interrupt irq0 CPUflags=00000001H
***** interrupt ACK           selectedIRQ=0001H
   serviceCode at 0008H  simTimer=0000012eH save
   PC=00000028H

 -- out channel 00000005H data = 000000ffH
 -- out channel 00000005H data = 00000020H
 -- out channel 00000005H data = 00000040H
 -- out channel 00000005H data = 00000080H
 -- out channel 00000005H data = 00000001H
 -- out channel 00000005H data = 00000002H
 -- out channel 00000005H data = 00000004H
```

Experiment 2: Simulation using Icarus

For experiment 2 we will again use the running light code from experiment 1, but we will extend it by adding an interrupt handler for interrupt 1 (see **Listing 2**). First bring up the SCCCP environment using c_compile.bat and compile the code using 'MAKE': the result will be placed in the C:SCCCP/experiment/experiment2 working directory.

Then use c.bat to convert the code into the internal format used by Icarus Verilog: the simulation can now be run using r.bat. **Listing 3** shows the output from this process.

You will see that the values 01H, 02H, 04H and so on are output, and these correspond to the LED patterns. The LEDs are accessed at SFR address 5. You can also see where execution breaks off to process an interrupt. Now this is not a lot more information than we obtained in experiment 1, and this is because the definitions to enable the debugging modes are still commented out in the Verilog source code for the CPU. Enable the following definitions in the first few lines of sCCCPcpu1v01.v by removing the '//' prefixes.

```
//`define VERBOSE2
//`define VERBOSE1
//`define VERBOSE3
//`define VERBOSE_INPA1
//`define VERBOSE_OUTAchrOut
```

Now, after recompilation, we will be able to see in detail what is happening inside the CPU. The output resulting from the execution of the first four instructions is shown in **Listing 4**.

Listing 4. Detailed debug output.

```
FETCH PC=0000H Time=000002  SP=03ffH OPC=2500000018H  R0=00000000H R1=00000000H
decode: CALL    00000018H
pending write, store adr=000003feH val=00000001H

FETCH PC=0018H Time=000004  SP=03feH OPC=2affffffffH  R0=00000000H R1=00000000H
decode: ADDSP   ffffffffH

FETCH PC=0019H Time=000006  SP=03fdH OPC=4000000001H  R0=00000000H R1=00000000H
decode: PUSHI   00000001H
pending write, store adr=000003fcH val=00000001H

FETCH PC=001aH Time=000008  SP=03fcH OPC=250000000eH  R0=00000001H R1=00000000H
decode: CALL    0000000eH
pending write, store adr=000003fbH val=0000001bH
```

Listing 5. The NOP and LDI instructions in Verilog.

```
if( sCstate==sCstateDecode1 ) begin
     if (sCopcode==sCopcNOP) begin
        sCstate<=sCstateFetch ;          // NOP instruction: nothing to do
        end                             // goto fetch phase of next instruction
      else if (sCopcode==sCopcLDI) begin
        sCregR0 <= sCoperand1 ;          // Data fom instruction goes to R0
        sCstate<=sCstateFetch ;          // goto fetch of next instruction
        end
      else
     ...
```

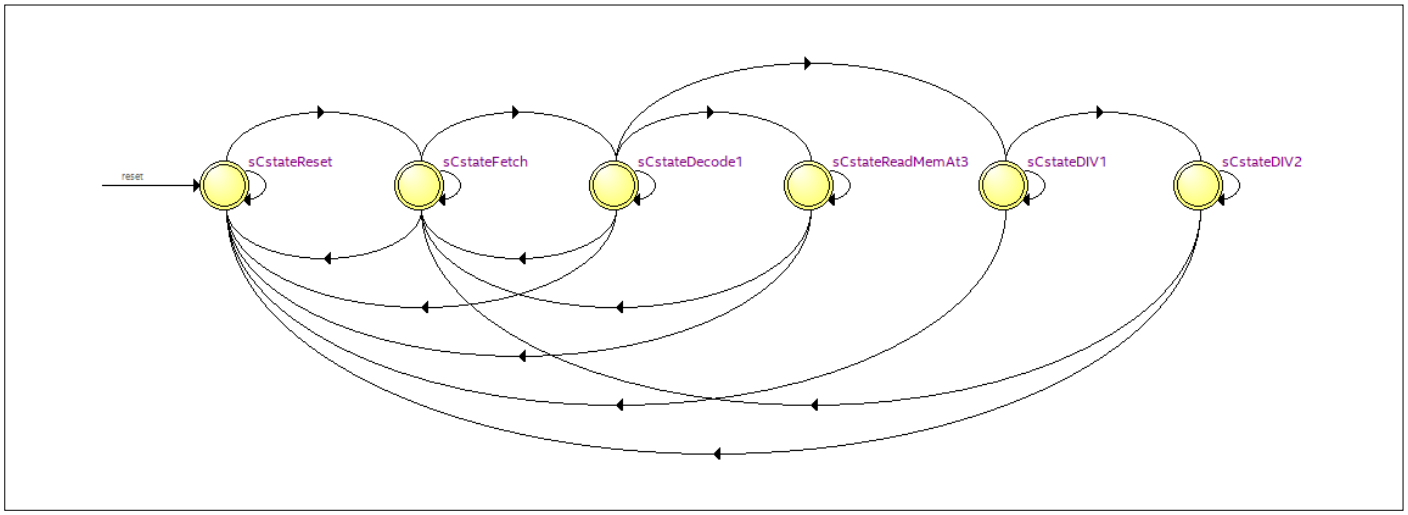


Figure 3. State diagram for the finite state machine that processes machine code instructions.

As you can see, we have more detailed information for each instruction that is executed: the contents of all registers, as well as the mnemonic for the instruction itself. Sometimes, indeed, it is possible to be somewhat overwhelmed by the sheer quantity of information, and then it is necessary to be creative in order to display only the information that is useful; but still there are many cases where the full level of detail is required to get to the bottom of a problem. One common issue is also that the simulation cannot reflect every aspect of a program's execution because it would be too complex to model every corner of a real design. Consider, for example, the case of analyzing radio signals in a real design. For this absolute accuracy in the simulation of timings is required, which can be too time-consuming.
However, when hunting down, for example, bugs in the CPU simulation, the Icarus Verilog simulator is a truly powerful tool, and indeed it has been used to detect and ultimately eradicate various bugs in the design.

Programming the CPU in Verilog

Now, with the help of a few selected instructions, we will explain how the SCCCP CPU was implemented in Verilog. Instructions are processed using a finite state machine: **Figure 3** shows its state diagram, as generated by Quartus. The first state is called sCstateFetch, and it is here that the instruction is loaded and the program counter (sCregPC) is incremented. The machine then moves to the next state, called sCstateDecode1. In Verilog that is all implemented as follows.

```
if( sCstate==sCstateFetch ) begin
 sCinstruction <= sCcodeMem[sCregPC] ;
 sCregPC <= sCregPC + 1'b1 ;
 sCstate <= sCstateDecode1 ;
 end
```

In the state sCstateDecode1 the instruction is decoded, and then we can begin with the execution proper. For many instructions this is practically the end of the story; for example, see the Verilog code in **Listing 5** for the NOP and LDI instructions. In each of these cases the next state is once again sCstateFetch, which begins the processing of the next instruction.
There are of course also more complicated instructions. The instruction LDPUSH aa loads the value stored at address aa, and writes it both to the stack and to R0. In the sCstateDecode1 state the value of aa is copied to the readAddress buffer register. The next state is sCstateReadMemAt3, in which the data memory is read, and the value fetched is written to the R0 register. Simultaneously the CPU prepares to write to the stack by storing the value of the stack pointer sCregSP minus one to the writeAddress buffer register: this new value is also written to sCregSP. The data value to be written, writeValue, is copied from the value just read from memory. And now, in the fourth state, we are ready to perform the write operation to memory. However, since our CPU employs a modified Har-

Web links

[1] MAX1000 board: www.arrow.com/en/products/max1000/arrow-development-tools

[2] MAX1000 board manual with guide to installing Quartus: www.arrow.com/en/products/max1000/arrow-development-tools/~/media/f19115ca0929486c9e09fc138b8c7c81.ashx

[3] 'Verilog Basics', ElektorLabs magazine 1/2019: www.elektormagazine.com/180562-02

[4] Project page with software download: www.elektormagazine.com/180394-02

[5] Icarus Verilog simulator: http://iverilog.icarus.com

[6] Icarus installation guide: http://iverilog.wikia.com/wiki/Installation_Guide

Instruction set overview of the homebrew CPU.			
instruction set (all operations 32 bit wide):			
Name	**cycles**	**meaning**	**explanation**
NOP	2	no Operation	
IDIV	37	signed integer division	R0=R1 / R0
IMOD	37	modulus operation	R0=R1 % R0 for positive operands
HALT	2	halts execution	
LDIND	3	load indirect	R0=memory[R0]
LDI n	2	load immediate	R0=n
ADDI n	2	add immediate	R0=R0+n
ADDSP n	2	add to stack pointer	SP=SP+n
JMP adr	2	jump	PC=adr
PCR0	2	load PC with R0	PC=R0
JFALSE adr	2	jump if HL==0	if(HL==0) PC=adr
LD @adr	3	load	R0=memory[adr]
PUSH R0	2+	save R0 on stack	memory[--SP]=R0
PUSH R1	2+	save R1 on stack	memory[--SP]=R1
CALL adr	2+	call subroutine	memory[SP--]=PC , PC=adr
POP R1	3	fetch R1 from stack	R1=memory[SP++]
POP R0	3	fetch R0 from stack	R0=memory[SP++]
XTR0	3	xchange top of stack and R0	R0=memory[SP], memory[SP]=R0
RET	3	return from subroutine or interrupt	PC=memory[SP++]
IADD	2	integer addition	R0=R1 + R0
IADD SP	2	add stackpointer to R0	R0=R0 + SP
IXOR	2	bitwise XOR	R0=R1 ^ R0
IOR	2	bitwise OR	R0=R1 | R0
IAND	2	bitwise AND	R0=R1 & R0
ICOM	2	bitwise complement	R0= ~ R0
INEG	2	signed negate	R0= - R0
IMUL	2	signed multiply	R0=R1 * R0
IASR	2	arithmetic shift right	R0=R1 >> R0
IASL	2	arithmetic shift left	R0=R1 << R0
ST @adr	2+	store	memory[adr]=R0
STIND @R1	2+	store indirect	memory[R1]=R0
OUTA ch	2	output R0 to channel ch	channel n=R0
INPA ch	2	input R0 from channel ch	R0=channel n
CMPEQ	2	compare equal	R0=(R1==R0)
CMPNE	2	compare not equal	R0=(R1!=R0)
CMPLT	2	compare less than	R0=(R1<R0)
CMPLE	2	compare less equal	R0=(R1<=R0)
CMPGT	2	compare greater than	R0=(R1>R0)
CMPGE	2	compare greater equal	R0=(R1>=R0)
CMPULT	2	compare unsigned less than	R0=(R1<R0)
CMPULE	2	compare unsigned less equal	R0=(R1<=R0)
CMPUGT	2	compare unsigned greater than	R0=(R1>R0)
CMPUGE	2	compare unsigned greater equal	R0=(R1>=R0)
The following instructions were added later to increase efficiency (higher speed, smaller code). These instructions are not necessary. They are used by a simple peephole-optimizer. Switching the optimizer off eliminates usage of these instructions.			
PUSHI n	2+	push immediate value	R0=n ; SP-- ; memory[SP]=R0
TWOOP n	2	prepare two operands	R1=R0 ; R0=n ;
LDPUSH adr	3+	load var and push	R0=memory[adr] ; SP-- ; memory[SP]=R0
ADDSPI n	2	load R0 by SP+offset	R0=SP+n
GETLCL n	3	get local variable	R0=SP+n ; R0=memory[R0]

Listing 6. The LDPUSH instruction in Verilog.

```
if( sCstate==sCstateDecode1 )
              readAddress <= sCoperand1 ;
              sCstate<=sCstateReadMemAt3 ;

if( sCstate== sCstateReadMemAt3 )
              sCregR0 <=sCdataMem[readAddress] ;
              writeAddress <= sCregSP-1'b1 ;
              writeValue   <= sCdataMem[readAddress] ;
              writePending <= 1 ;
              sCregSP <= sCregSP-1'b1 ;
              sCstate<=sCstateFetch ;

if( sCstate== sCstateFetch )
    if(writePending) begin
    sCdataMem[writeAddress[sCdataMemAddressWidth-1:0]]<=writeValue ;
    end
```

Listing 7. The LDPUSH instruction in the Java simulator.

```
instruction=ram[regPC] ;
opcode=(instruction>>32) & 0xFF ;     // 8 Bit opcode
operand1=instruction & 0xFFFFFFFF ;   // 32 Bit immediate operand1

switch(opcode){
  case(ccNOP) :    { return ;}
  case(ccLDI) :    { regR0=operand1 ; return ;}
  case(.ccLDPUSH) : { regR0=readDataMemX(operand1) ;
                      regSP-- ;
                      writeDataMemX(regSP,regR0) ;
                      return ;}
```

vard architecture, we can simultaneously access data memory and code memory. Thus the outstanding write operation to the data memory can be overlapped with the fetch state of the subsequent instruction, and this is indicated to the state machine by setting the writePending flag. The overall effect is to reduce the execution time of the instruction from four to three cycles.

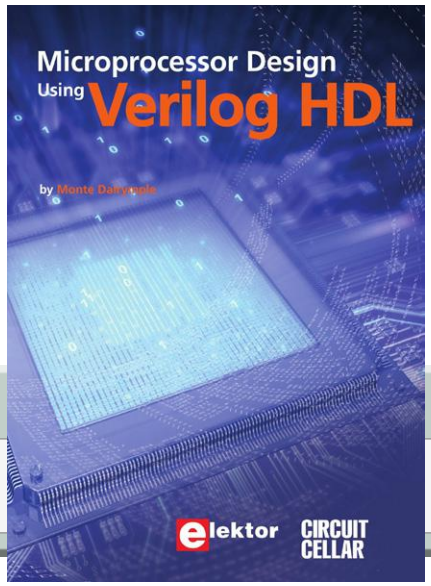
Listing 6 shows the implementation of the LDPUSH instruction. Code along the same lines must be written for each possible instruction: see the text box 'Instruction set overview'. This is not a huge task, but getting it all finished and exterminating all the bugs does take a couple of days of programming effort. The author has included a simple simulator for the CPU in the SCCCP development environment. The Java code corresponding to the LDPUSH instruction is much more straightforward: see **Listing 7**.

Simulating the CPU will be the subject of experiment 3, which we will describe in the second installment of this series. ⏮

180394-02

@ WWW.ELEKTOR.COM

→ E-book 'Microprocessor Design Using Verilog HDL'
www.elektor.com/microprocessor-design-using-verilog-hdl-e-book



Fan Controller

Your choice of Arduino or analogue electronics

By Jac Hettema (Netherlands)



In circuits that dissipate a lot of power, active cooling is generally the only solution. In other words: with a fan. To prevent wear and, more importantly, limit the noise, it is desirable to adapt the speed of the fan to the power to be dissipated. This requires some kind of controller. In this article we describe two versions: 'old-school' analogue and 'fashionable modern', using an Arduino with software.

Electronic circuits often struggle with excessive heat. When the amount of heat to be dissipated is small, it usually suffices to use only passive cooling, that is, with a simple heatsink. But when that power increases that heatsink needs to increase in size as well. This will eventually result in insurmountable problems and the solution can then be to use active cooling — with a fan. How you could control such a fan, we will show here.

Analogue electronics or software?

The circuit we use here is frugal with energy. For the controller we propose two variants: the first using old, familiar, analogue electronics, the second using an Arduino (Pro Mini) with appropriate software (firmware). The controller uses a PWM signal to drive a control element, for which we use a transistor. By adding an inductor, diode and capacitor in the output circuit, the circuit acquires the character of a switching power supply, with the corresponding characteristics. We begin with the analogue version, see **Figure 1**.

Analogue

For the analogue version we use a transistor as the temperature sensor. We make use of the temperature coefficient of a base-emitter junction (about –2.2 mV/K). We connect the base to a voltage divider between the collector and emitter (R4 and R5), with which we increase the change in voltage. This voltage is then compared by U1A to the voltage at the wiper of potentiometer R1. With R1 we can therefore adjust the fan controller. R1 obtains its power supply from zener diode D2, which limits the voltage in the first instance to 7.5 V. The current through R1 also flows through D1 (a blue or green LED), which reduces the voltage further. This design has a few (minor) advantages

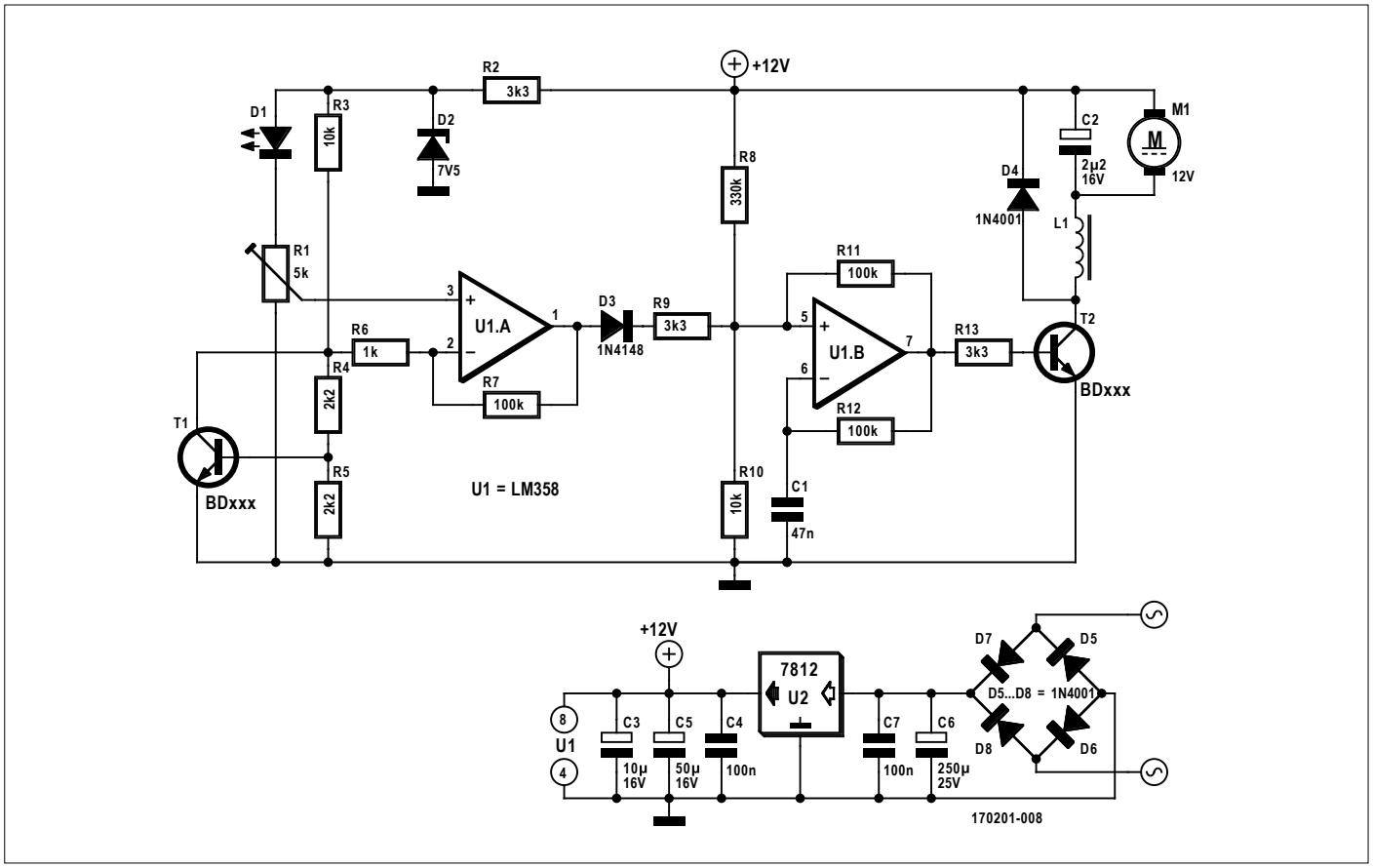


Figure 1. In the analogue version of the fan controller we see the LM358 as an astable multivibrator.

- The zener diode has a value at which the temperature coefficient is minimal, the LED has a negative temperature coefficient, so the voltage across the potentiometer now has the positive temperature coefficient of the series resistor of the LED (R2), which is used to 'measure' the ambient temperature.
- T1 has a negative temperature coefficient and measures the temperature of the object to be cooled. On balance, the voltage difference depends both on the temperature of the object to be cooled and the ambient temperature, such that at a higher ambient temperature the control voltage increases and the fan will spin faster.
- The LED also gives an indication that the circuit is powered on.

We now feed the sensor voltage to opamp U1A, an LM358, which has a voltage gain of 100 times. this is quite a lot, so that small temperature variations have a large effect. If that is not necessary, then the gain can be changed appropriately with the ratio of the resistors R6 and R7. When we take a smaller value for R7, the gain is also smaller. This gain can, of course, also be made adjustable by replacing R7 with a potentiometer that has its wiper connected to one of its other pins. With a 100k potentiometer you can adjust the gain from (approximately) 1 to 100.

The second output of the LM358, U1B, is configured as an astable multivibrator. The non-inverting input is connected to the output of U1A, via a voltage divider (R8 and R10), R9 and D3. For a more thorough technical background on the operation of the opamp multivibrator you can rely on the internet, for example at [1].

The duty-cycle of the multivibrator is determined by the voltage at the non-inverting input of the opamp. When this voltage is equal to half of the supply voltage, then the duty-cycle will be 50%. When the voltage at the non-inverting input is lower, the output voltage will have to be high for a longer time before the opamp changes state. This therefore lets you influence the

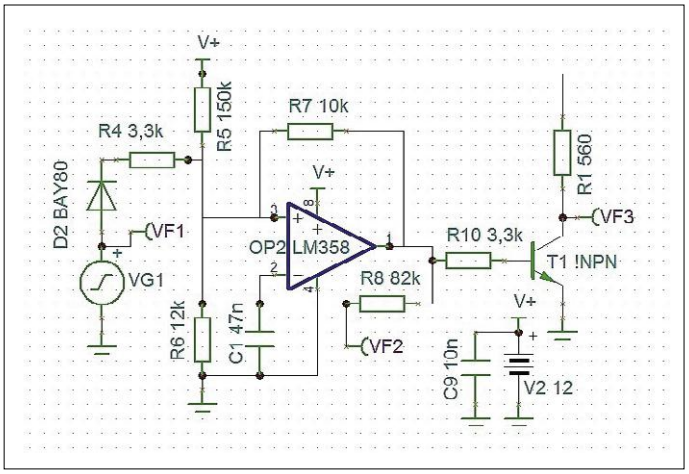
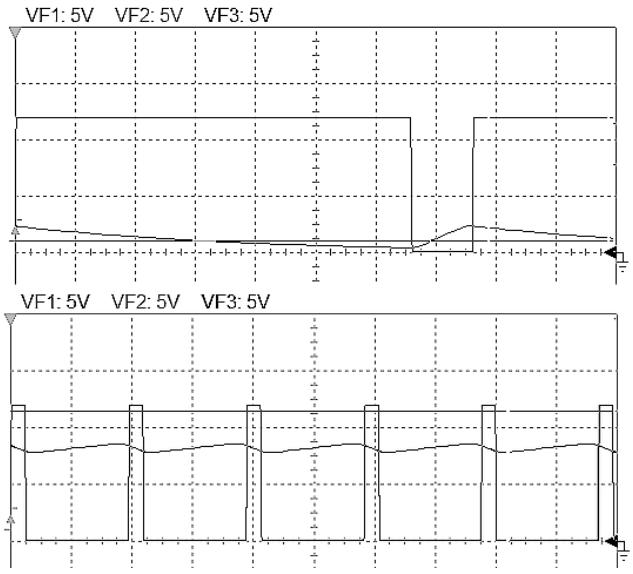


Figure 2. The analogue circuit is easily simulated in Tina TI.



Vf1 (V)	t (msec)	th (msec)	f (Hz)	duty cycle
1	12	9	83	0.8
11.5	4.5	1	222	0.2

Figure 3. From the simulation results we can clearly see that not only the duty cycle changes, but also the frequency.

duty-cycle. A lower input voltage means a higher duty-cycle and the other way around.
We can simulate this nicely in Tina TI V9 [2]. **Figure 2** shows the schematic that we used for that. In **Figure 3** we show the results from that simulation. Here you can easily see what effect the input voltage has on the duty-cycle.
As the alert reader has probably already observed, not only the duty-cycle changes, but also the frequency! In this circuit, however, that is not a problem.
The schematic in **Figure 1** indicates an NPN transistor for T1, but this could just as easily be a PNP transistor, in which case you need to swap the collector and emitter around. An advantage could be that a transistor of the BD-type then has its collector connected to ground, so that when mounting the sensing transistor on the heatsink there is, in principle, no isolation required, which improves the heat transfer.
For L1 there is deliberately no value indicated in the schematic.

@ WWW.ELEKTOR.COM

→ JOY-iT Uno R3
www.elektor.com/joy-it-uno-r3

→ Elektor Sensor Kit
www.elektor.com/arduino-sensor-kit

→ Book: Arduino in Control
www.elektor.com/arduino-in-control-3de-versie

That is because this has to be determined experimentally: just take a look at what you have on hand, try it and observe how the circuit behaves. Its presence acts as a low-pass filter for the PWM signal and gives a quieter operation of the fan (for which no specific type is indicated either).

Digital

For the microcontroller controlled version we started with an Arduino Pro Mini. These boards can be found for not much money on Ebay and the like, and are perfectly suitable for our purpose. It is possible to make the circuit even smaller by using a Digispark with the ATtiny85, but we haven't gone this far (yet). In principle it could work with any Arduino, but it will be easiest to use one with the same microcontroller. On the website for Arduino itself [3], there is a handy overview of the specifications.
This circuit is much simpler than the analogue version, refer to the schematic in **Figure 4**. What is immediately obvious, is that for the temperature sensor an ordinary signal diode has been used, and not a transistor as in the analogue version. This has to do with the choice of reference voltage for the Arduino. Here we used the internal reference voltage of 1100 mV of the Arduino. If we had used a transistor as thermal sensor then the input voltage would be more than the reference voltage. And the microcontroller cannot deal with that.
The resolution of the internal A/D-converter is 10 bit, which corresponds to voltage steps of 1.1 mV. This step size and using a single diode then results in a temperature resolution of about 0.5 K, which is sufficiently accurate for this application. Because both the reference voltage of the Arduino as well as the characteristics of the diode used varies with each individual part, it is recommended to perform a few calibration steps. You do this by measuring the voltage across the diode at 0 °C and at 100 °C, using the Arduino and reading out the result. Once the voltage at these two points is known, you can simply calculate the temperature for all the voltage values in between. You can then also calculate which value the PWM signal has to have for each measured temperature.
You could, of course, also choose to measure the temperature with a digital sensor, such as the 18B20. But this one is not always on hand in the parts drawer and then it is good to know that it can also be done with just an ordinary diode. The 1820B can be connected in place of the diode without any other changes. It will then operate in the 'one-wire-mode', which is not a problem for short wire lengths.
By the way, the software still has 'some room for improvement'. The temperature readout works as it should, but the changing of the PWM signal as a consequence of the temperature, for example, has not been thoroughly tested.

Motor control and power supply

In our circuit we assumed the use a simple 12-V fan with not a lot of power. For the control of this we use a BD135 or similar (T2). The collector is connected via a noise suppression choke (L1) and related components to the fan motor.
The circuit is, of course, not limited to the use of a BD135. With a more powerful transistor, much bigger loads can be controlled (also note whether the Arduino or the LM358 are able to supply the higher drive current requirement of a bigger transistor).
For the power supply for both versions of the circuit we reverted to the old, familiar, 7812 regulator. Diode network D5 through

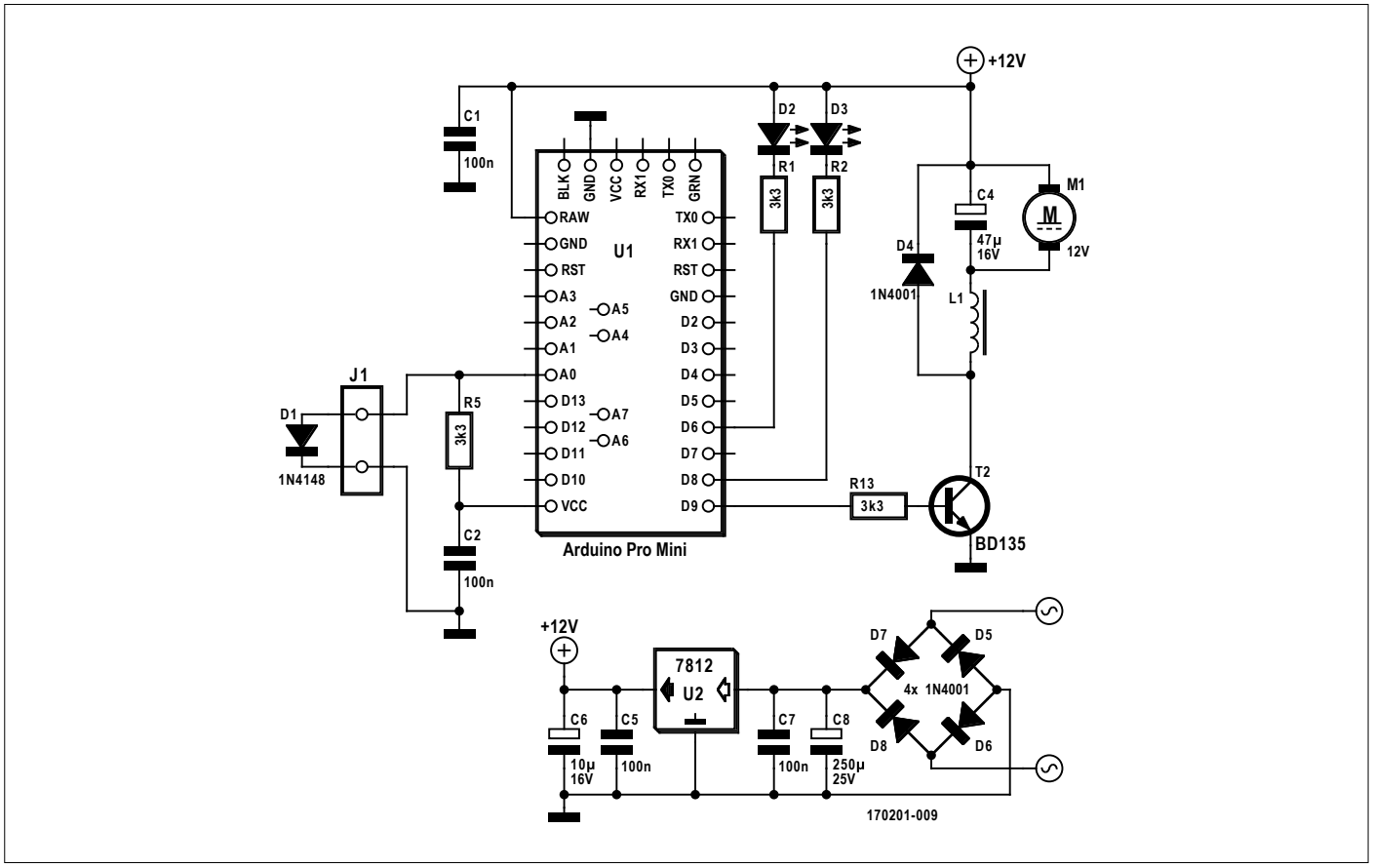


Figure 4. With the digital version we can see that the first part is replaced by the Pro mini and that the second part is practically unchanged.

D8 ensures that both DC as well as AC (up to a maximum of about 18 V_{rms}, depending on the cooling and load on the 7812) can be connected to the power supply. Additionally, only a few buffer capacitors are necessary for the correct operation of the 7812. Most of the current will be going to the fan, so if you allow for a surplus of about 100 mA above the nominal current rating of the fan for the power supply, then you should have no problems.

We powered the Arduino Pro Mini from the 12-V regulator. The Arduino has its own power supply regulator on board that reduces the voltage to 5 V for the microcontroller. This 5 V is also available at the VCC pin of the module. We gladly make use of that for powering the sensor diode.

Printed circuit board design

For these two circuits the author designed two circuit boards, which are available for download from the web page for this project [4]. The board for the digital version has a few additional features, compared to the analogue version:

- It is has two additional outputs for, for example, an LED or an optocoupler. The switching thresholds for these are easily added to the software.
- There is the build option for a header which makes +5V, GND, A4 and A5 available. To this you could, for example, connect a module that can communicate via I^2C.
- It is possible to power the fan from an external power supply. For this there is a header with a jumper. If the voltage of the external power supply is higher than 12 V, then you need to take note whether the BD135 can handle that. Potentially another transistor (and/or cooling) is required. Also the voltage rating of C4 needs to be sufficiently high.

Get cracking

Both schematics have been drawn in Ulticap 5.72 and the PCBs have been designed in Sprint 5. The software, in the form of an Arduino sketch, has been made available by the author. For programming, the freely available Arduino IDE is required [5]. The Ulticap schematics, Sprint layouts and Arduino sketch can all be found at [4]. With all this you can get cracking!

170201-02

Web links

[1] Operation of the opamp multivibrator: www.electronics-tutorials.ws/opamp/op-amp-multivibrator.html

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

[3] Specification overview Arduino: www.arduino.cc/en/products.compare

[4] Web page for this project: www.elektormagazine.com/170201

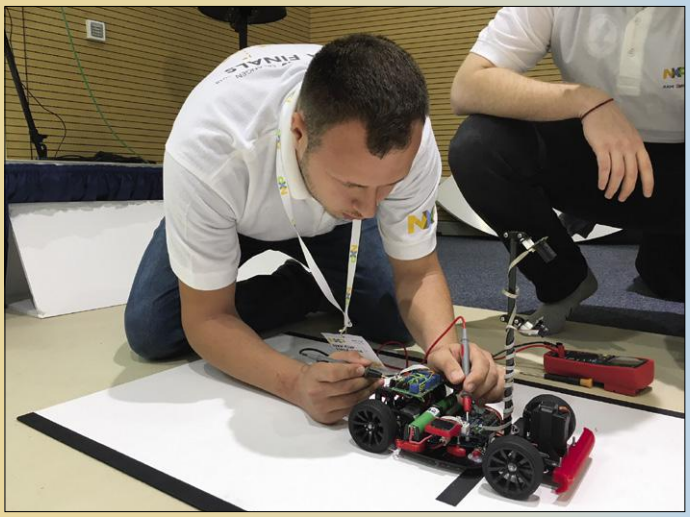
[5] Arduino-IDE: www.arduino.cc/en/Main/Software

The NXP Cup 2019
Is Bigger and Better Than Ever!

By **Clemens Valens** (Elektor Labs)

The NXP Cup is an annual global competition for student teams and robotics clubs aspiring to build, program, and race a fully autonomous model smart car as fast as possible around a prescribed track. And not just for fun, as the overall winning team will be winged to Silicon Valley (and back)!

The objective of the NXP Cup is to build and race pro-level autonomous cars without spending pro-level money. To accomplish this, well designed and highly optimised hardware and software are needed — clearly not an easy task.



The NXP Cup is a great way for engineering students to become familiar with the skills they will need in their professional career. Or, as Lars Reger, CTO of NXP Automotive put it: "*The technology, university support, and the incredible partner ecosystem give students opportunities to learn key collaborative teamwork skills, and become familiar with core technologies to build autonomous robots.*"

Creating an autonomous car capable of participating, and, if possible, winning, requires:

- embedded software programming and basic circuit design skills;
- optimizing motor control hard- and software for highest speed and best steering precision;
- embedding a camera to assist the car to remain on track by following a guide trace.



To help the participating teams to achieve their goals, the 2019 edition of the NXP Cup comes with revised rules that allow for more freedom:

- unlimited sensors;
- free choice & unlimited number of NXP processors;
- more challenges to earn extra points with: Timed Race, Figure-of-8, Obstacle Avoidance and Speed Limit Zone.

Besides more lenient rules and extra challenges, the track itself has been improved as well and offers better grip, allowing for higher speeds. ⏮

180692-01

Web link

[1] NXP Cup EMEA: https://community.nxp.com/groups/tfc-emea



HOMELAB PROJECT

RPi Stepper Motor HAT with Anti-resonance

Boost efficiency and calm stepper-motor jitters

By **René Merz** (Magnetron.ch)

Stepper motors have become indispensable in the design of modern electromechanical equipment. You can find them used everywhere from 3D printers to hard drives and robotics. Despite their popularity, stepper motors also have a dark side; this includes resonant behaviour and poor efficiency. This article will show you how you can combat both these problems.

Stepper motors are available in all sizes, offering a wide range of step size, frequency range and physical outline to suit the respective application. Hybrid stepper motors and claw-pole variants are increasingly used in 3D printers to provide linear motion using spindles. Often they will be operating below optimal conditions, i.e. at high frequencies and low loads, where effects such as mechanical resonance and vibrations will be evident. To achieve optimized stepper motor operation, it's important to take these phenomena into account in the motor control method and, if possible, counteract them. This project builds a complete dedicated stepper-motor controller HAT for the Raspberry Pi which combines advanced anti-resonance motor-driver techniques together with a stepper-motor power output stage.

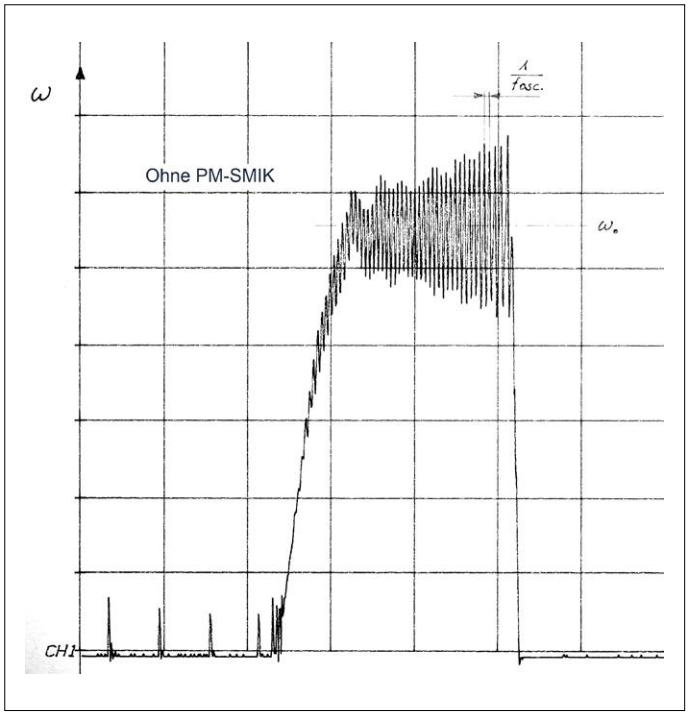


Figure 1. Typical rotor oscillations in small stepper motors versus frequency measured at the shunt. X = 500 ms/division and Y = 200 mV/division. Without anti-resonance measures the rotor oscillation frequency is 34 Hz. After uncontrolled oscillation the motor stalls.

Resonances

For stepper motors, a distinction is made between self-resonances and the so-called parametric or mid-band instability range. Self-resonances occur at frequencies in the range of 1 to 300 Hz and can be controlled by mechanical countermeasures (gear selection, damping etc.). Operating in this range, a stepper motor typically achieves an efficiency of <10%.
Above 300 Hz (or steps/s) parametric resonances occur in certain frequency bands. Oscillations of the rotor rotation around its respective step position result in resonant behaviour (see **Figure 1**). These rotor vibrations can build up and reach large amplitudes due to resonance thereby making the rotor no longer synchronous with the control frequency; it can lose steps and ultimately stall. Without suitable countermeasures, the usual solution to this situation is to operate the motor with a restricted drive current below its maximum rating so that the full power range cannot be fully utilized (see **Figure 2**). The solid line in **Figure 3** shows ideal torque against operating frequency and the dashed line shows without countermeasures both the torque and control frequency remain below that which could be achieved.

A solution

It is possible to counteract mid-band resonances using an electrical feedback loop. To measure the resonance phenomena you can tap the signals produced by the oscillations using

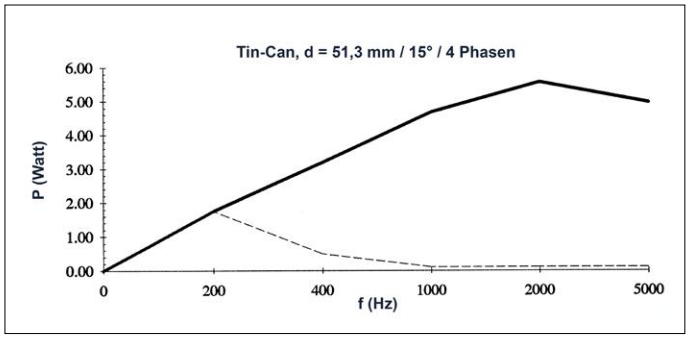


Figure 2. Mechanical engine power in watt at I = 0.4 A/Ph and ED = 30%. The dashed curve shows what is achievable without anti-resonance circuit. This results in three times the possible power plus significantly higher drive frequencies.

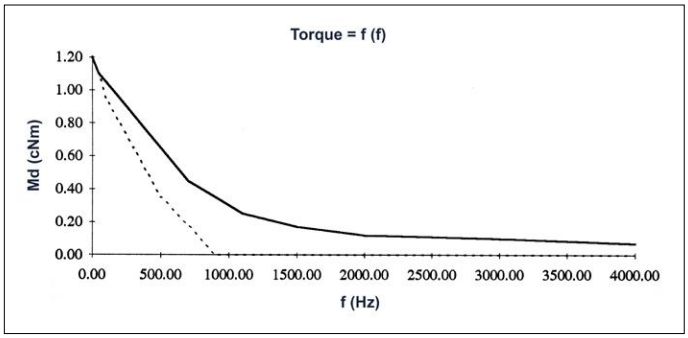


Figure 3. Torque against frequency. The solid line shows the enormous improvement with the anti-resonance circuit.

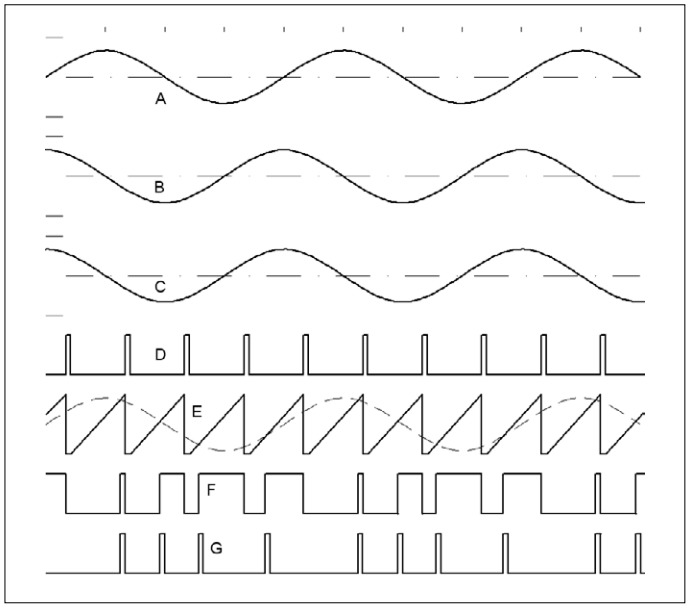


Figure 4. Phase relationship of the motor control signals. A: rotor speed ω; B: load angle rotor γ; C: U = $f_{(IT)}$, in phase with signal B; D: Clock-In (f_0), from the frequency generator; E: sawtooth signal and U' after filter, 90° out of phase; F: PWM; G: Clock PM, feeds the sequencer.

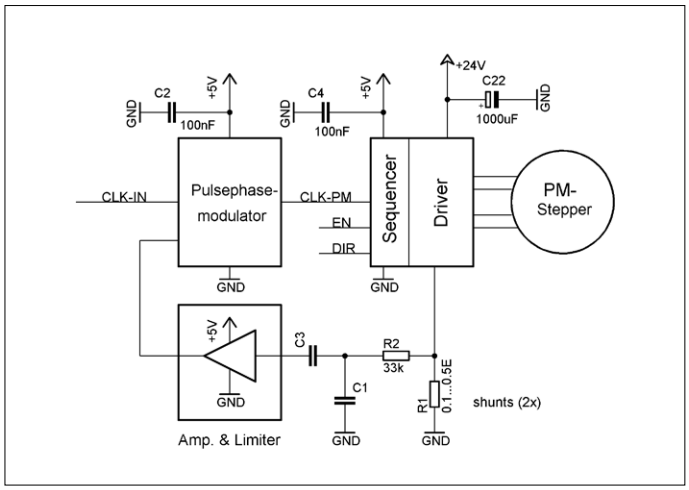


Figure 5. Principle of the anti-resonance circuit. The feedback signal at the shunt is used to phase shift the control pulses.

a shunt resistor (the voltage drop produced is proportional to the current through the rotor winding). Figure 4 illustrates the idealized phase relationships of the rotor and control signals. **Figure 5** shows the principle of a possible countermeasure, developed and described in 1986 by Pierre Boillat [1], whose patent protection has in the meantime lapsed.

For smaller claw-pole stepper motors (Tin-Can motors), the oscillations typically occur in the range from 5 to 80 Hz. After filtering and amplifying the shunt signal (below), it is used to control a pulse-phase modulator. The output signal CLK-PM controls the sequencer input of motor-driver stage with a corrected phase ratio. Such a pulse-phase modulator can be used if the actual clock generator (usually a microcontroller) is not able to or not designed to carry out the phase modulation process. Using this hardware approach it's possible to achieve very high step speeds up to 15,000 steps/s, which also allows the stepper motor to operate at higher efficiency. Another advantage of this approach is that you do not have to comply with special acceleration ramps to drive the motor through the region of instability. Its worth mentioning that this method of stabilization is not related to chopper power controllers that you sometimes find integrated into some stepper-motor driver chips available today.

The anti-resonance circuit

As a practical demonstration of the principle, the author has developed the circuit in **Figure 6**. The PCB is in the form of an expansion board (HAT) which plugs onto a Raspberry Pi A+ board. The clock signal from an external microcontroller is connected to CLK-IN at pin 3 of the screw terminal strip X2. The signal is shaped with one of the Schmitt-input inverters of IC7 and triggers the monoflop IC3A on the negative edges. The output signal of IC3A at Q consists of a series of pulses with the constant width of about 30 µs. These pulses are used to discharge capacitor C2 via MOSFET T2 of the square-wave/saw-tooth converter built with IC1A, T1, T2 and T3. The constant current source made up from T1 causes C2 to be charged to its maximum value (after discharge via T2) producing a rising linear voltage ramp waveform. The result is a fairly clean saw-tooth signal at the positive input (pin 3) of comparator IC2A. Together with the modulation signal **U'** at the negative input (pin 2), we can see pulses at the output (pin 1) with the desired (linear) phase modulation.

To ensure a wide frequency range of the modulator the saw-

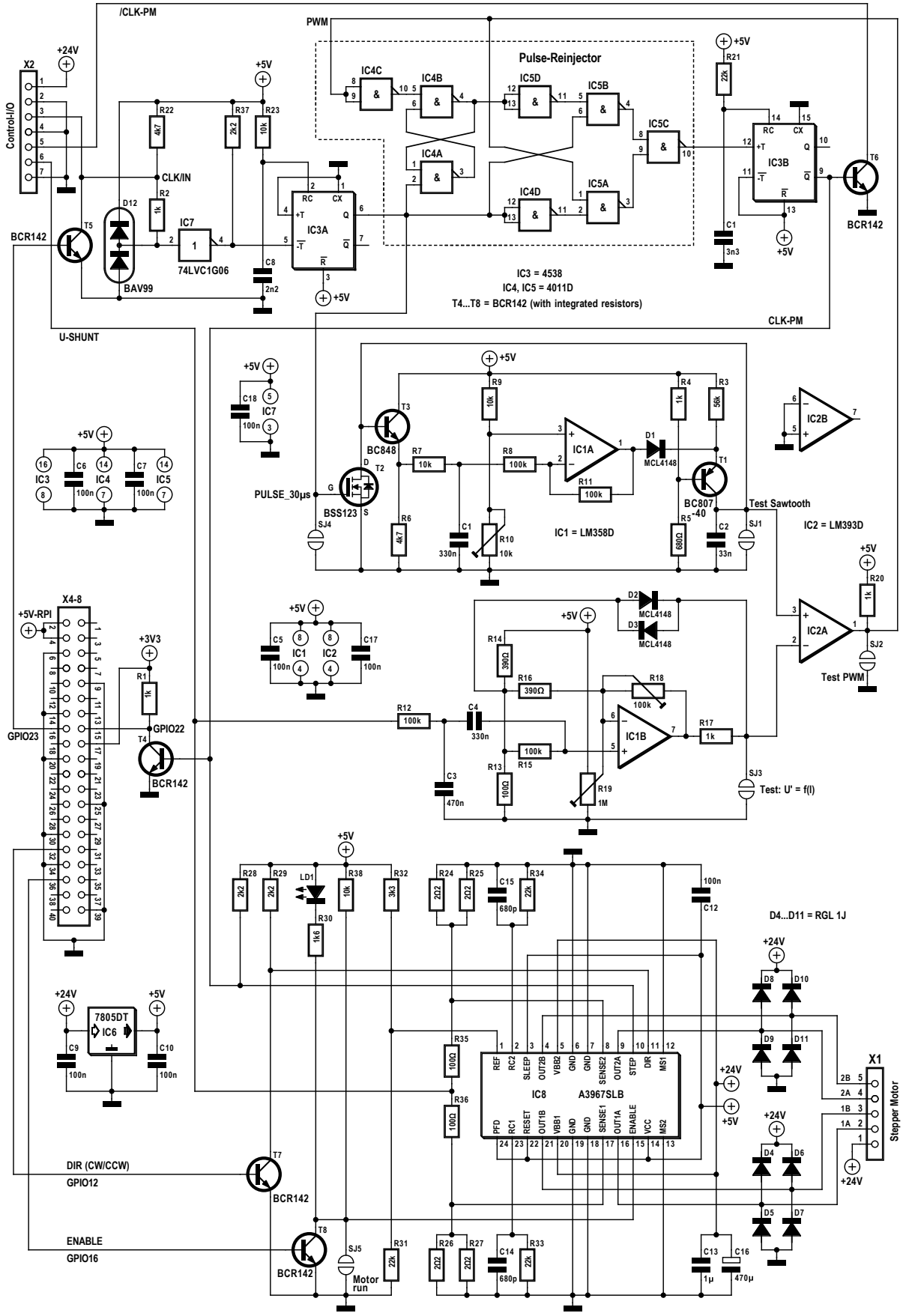


Figure 6. The anti-resonance circuit implemented as an RPi-HAT.



Figure 7. Signals of the anti-resonance circuit with a step angle of 15° at 901.75 Hz. C1: 30 µs pulses on SJ4; C2: saw-tooth on SJ1; C3: PWM signal to SJ2; C4: shunt signal, U = $f_{(IT)}$ at X2-6.

tooth amplitude must be independent of frequency. To achieve this the current source formed by T1 is controlled through IC1A. The amplitude is kept constant by using the mean value (the DC component) of the sawtooth by feeding the signal back to T3 and low-pass filtering with R7 and C1 to give a feedback voltage applied to the inverting input of IC1A. The output signal is then fed via D1 to control the current source.

With a very fast sweep (for example, where only a few pulses occur in a ramp from 0 Hz to 4 kHz in only 200 ms), the control loop is too slow due to the time constant of R7 / C1. In this case the saw-tooth amplitude becomes clipped which leads to missing pulses at the end of the sweep output from comparator IC2A. That can then cause motor-actuator positioning errors. In order to prevent a lack of pulses, the signal is looped around to the Pulse-Reinjector circuit made up of NAND gates IC4 and IC5.

The monoflop IC3B is needed because only the positive signal edges are modulated, some commercial sequencer drivers will trigger on the negative edge. The output signal /CLK-PM is driven by the open-collector configuration of T6 so that an external circuit just needs to supply a pull-up resistor to the voltage used by the external circuit in order to achieve any necessary signal level shifting.

The modulation voltage level **U'** is produced from the signal at the driver-shunt resistors R24 to R27 filtered by a bandpass filter formed by R12, C3, C4, R15 and amplified by IC1B. In rare cases the bandpass may need to be adjusted (depending on the motor's rotor inertia). The gain can be adjusted between 1 and 250 times using preset resistor R18. Practice has shown that a gain setting of around 50 is usually about right. The pads SJ1 to SJ4 on the board serve as test points for an oscilloscope probe to study the waveforms. The waveforms shown in **Figure 7** can be adjusted using preset resistors R10, R18 and R19. Preset resistor R19 is used to adjust the DC level at the output of IC1B. The level measured at SJ3 should be half of the peak value of saw-tooth SJ1 and in the range 1 to 1.3 V.

To power the stepper-motor an integrated driver chip from Allegro (IC8) is used [2]. The IC A3967SLB contains a sequencer, a PWM chopper current control for microstepping and a 0.75 A/Ph bipolar transistor power amplifier stage. The reference voltage at Pin 1 of IC8 can be adjusted with the voltage divider network R31/R32 to set the output drive current. This should be adjusted according the current rating of the motor in use.

The RPi HAT

The RPi HAT PCB for this project is shown in Figure 8. It plugs directly onto an RPi A + board via the GPIO 2 x 20-pin header connector. This can then be used to generate the necessary control signal for the motor for all kinds of practical applications. To do this, the 3.3V level clock pulses generated by the RPi are routed through Port GPIO23 and T5 to the anti-resonant circuit on the HAT which operates using 5V logic levels.

The CLK-IN signal uses a pull-up resistor (R22) to convert the 3.3 V input signal level to a 5 V signal. The two other control signals /ENABLE (motor enable, active Low) and DIR CW/CCW (direction of rotation) use the same level-shifting

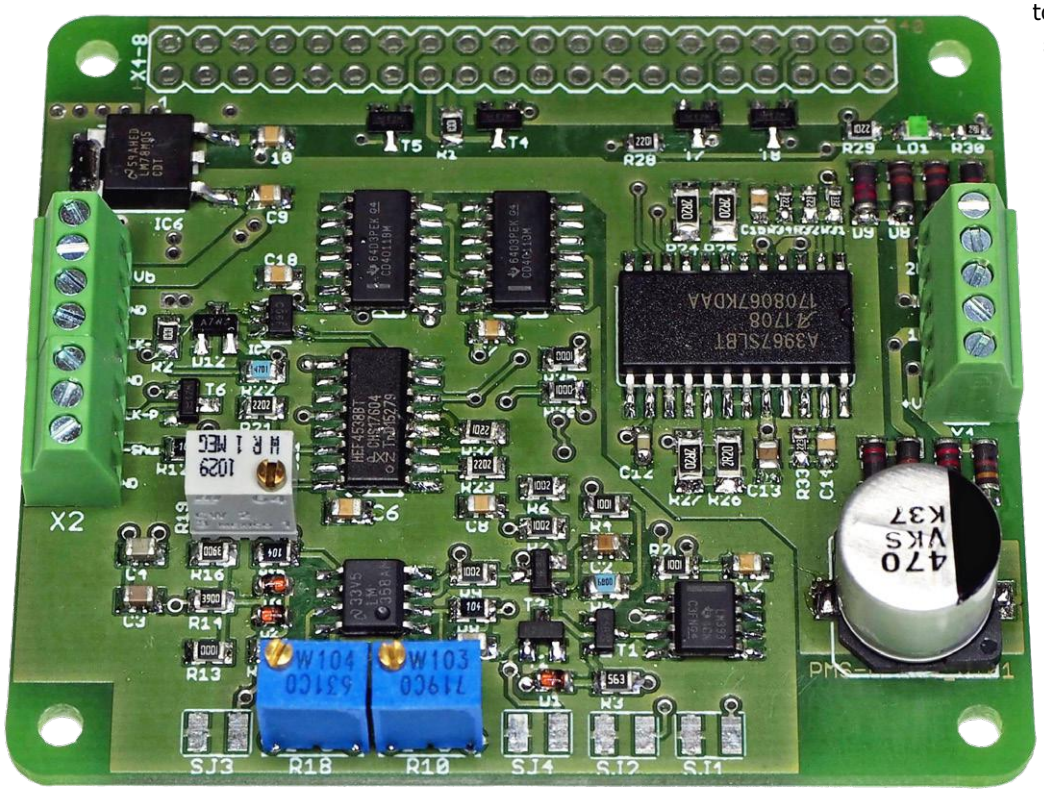


Figure 8. The prototype anti-resonant stepper-motor controller in the form of an RPi-HAT for the model A+.

configuration via transistors T7 and T8. The motor ENABLE input at GPIO16 is therefore active high. If you are using the board as a HAT with an RPi you will need four spacer bolts and screws for mechanical support. The motor-control HAT should be powered independently from the RPi (use two separate power supplies). Any small 2-phase stepper motor, regardless of whether they are claw-pole or hybrid types (NEMA17), bipolar or unipolar, will be suitable for testing. They should have a winding resistance in the range 5 to 100 Ω/Ph with a power rating of 1 to 15 watts. For resistances ≤ 5 Ω/Ph, the driver current setting should be made very carefully. The motor cables should never be disconnected while the motor is in operation.

Although IC8 is designed for bipolar stepper motors, unipolar versions can also be connected at screw terminal strip X1. In this case the centre tap for the coils is connected to position 1 (+24 V) of X1.
If you want to make the HAT board yourself, you will find all the circuit and board files on the Elektor website for this article [3]. The **Technical Data** can be found in the adjacent table. Finally, now all the necessary motor control hardware is sorted, all you need to do is to program a pulse generator suitable for test purposes or for your application, together with a corresponding application for the RPi. Have fun...

Technical Data	
DC operating voltage	7 to 24 V
Control frequency burst	1 mHz...25 kHz
Time delay CLK-In/Out	approx. 5 ms at 1 kHz
Phase current max.	0.75 A/Ph

Things to consider and project background

Back in 2014 the author was using a 3D printer; several of its drive shafts were equipped with bipolar, hybrid stepper motors (four-wire, NEMA17) with the windings connected in series. The control frequencies used to drive the motors were unfortunately in frequency ranges where the printer housing exhibited strong mechanical resonances. The large hybrid stepper motors could have been replaced by smaller versions if an anti-resonance circuit and an adapted transmission were used in the control system. Many users often turn to microstepping mode to reduce vibrations in the self-resonance range and also to achieve a higher degree of resolution. A finer resolution alone does not necessarily mean greater positional accuracy.
The reason for this lies in errors in the geometry of the motor stator, flux leakage in the soft-magnetic parts used in the stator and deviations of the desired drive current value (PWM chopper, 1/8 and 1/16 step) with low phase currents!
Using the same drive frequency for the sequencer with eight microsteps, reduces the motor shaft speed by a factor of 8. In the formula for the mechanical power output ...

$$P_{mech} = Md_x \, \omega$$

... the angular speed ω of the motor shaft and the torque (*Md*) both play an important role: The load must be driven to its position as quickly as possible. To help achieve this, the microcontroller should provide one or more pulse generators. It is important that the controller has enough memory (NV-RAM or Flash) to store tables of values to generate frequencies, number of burst counts and saw-tooth generation. This is not a problem for the SoCs used in the RPi.
To make stepper motors more easy to use in the future, manufacturers should clearly document the efficiency and/or the ratio between electrical power input and mechanical power delivered as a function of the control frequency. In order to get the most out of stepper motors the use of complex algorithms to define acceleration and deceleration ramping profiles are central. Of course that requires extra programming effort. Avoid driving the motor at frequencies which correspond to its natural resonances. This also applies to the mechanical installation of the motor: ensure the motor mount has high level of integral damping. With regard to the use of a backlash-free drive (harmonic drive) or a suitably designed toothed belt transmission, it is important to consider the relative cost-benefit ratio for each application. Maybe in the future we will see the emergence of low cost driver ICs featuring a class D output stage that would also output filtered sine and cosine signals for the motor phases. As for the motors it would be advantageous if the stator materials had lower magnetization and hysteresis losses, nowadays it is common to see stepper motors operated with chopper frequencies as high as 18 kHz. ⏮

180279-02

About the Author

René worked for SAIA AG (now Johnson Electric International AG, Murten, CH) from 1985 to 1990. During his time there he developed new magnet rotor designs for stepper and synchronous motors. Since then, he has also been involved in the design of stepper motor controllers. In 1992 he started up his own business Magnetron Labs Merz.

Web links

[1] Pierre Boillat, 'Restraining the instability of a stepper motor', US-Patent US4673855 and US4683409.

[2] Allegro A3967: https://bit.ly/2UoRnk1

[3] Article support page: www.elektormagazine.com/180279-01

@ WWW.ELEKTOR.COM

→ Book: 'Motors for Makers'
www.elektor.com/motors-for-makers

→ Mendocino-Solar-Motor X-8:
www.elektor.com/mendocino-motor-x-8

HOMELAB PROJECT

Rear Radar for Bicycles

a simple project for better bicycle safety

By **Laurent Labbe** (France)



Having lived in Asia for a long time (Shenzhen, China, and currently Hong Kong), I regularly ride my bicycle to work, or to train for triathlons. That's why I conceived this project around an 8-pin microcontroller; it allows me to avoid a stiff neck by monitoring the vehicles around me.

When I used to ride to school on my bicycle, a (very) long time ago, and had to go across town, I had a small rear-view mirror left and right to be aware of the vehicles that were passing me. Nowadays, that kind of accessory is difficult to find. That's why I thought of a system that could alert me without having to turn my head to left and right, and thus concentrate on the road in front of me. In China, the cars and buses go every-which way, often without any warning, so you have to be very careful. It's even worse in Hong Kong. The roads are very narrow, so buses, taxis and minibuses are always brushing against you as they pass.

The system proposed here is a sort of radar, composed of two ultrasonic sensors placed at 90°. They measure the distance between the seat of the bicycle (where the device is installed) and vehicles approaching from the rear or from the left (in the case where one drives on the left as in Hong Kong). It is possible to add a third sensor. The prototype presented here (**Figure 1**) doesn't have one, but the downloadable software [1] can handle three sensors. The final position of the module depends on the driving convention (on the left or the right) hence on the country. The module should be installed more to the left (in the case of driving on the right) or more to the right (if driving is on the left as in China). If the module is aligned with the axis of the bike, with the two sensors each at 45° to each side of the direction of travel, I found that stationary or parked vehicles (which are on the left in Hong Kong or on the right in China) cause false alarms.

How to alert the cyclist, without his eyes leaving the road? I chose to use 2 small vibrator motors, placed under the seat, to left and right to indicate the direction from whence the vehicle

is approaching. The number of vibrations (1, 2 or 3) gives an idea of the distance. One can also imagine putting the vibrators on or in the handlebars, on the left and the right, by running longer wires. The vibrators are small motors with an eccentric flyweight, recovered from old mobile phones.
Finally there remains the question of power supply. the consumption must be as low as possible, so the device is limited to one function only: radar. It is powered by a Lithium battery, recharged by a dedicated integrated circuit (here the MAX1555). I did not use a regulator. Recharging from solar energy remains an option, maybe in the next version.

The circuit

As I had the idea of being able to measure the battery voltage, I needed a microcontroller with an integrated voltage reference but small (8 pins) to minimise the power consumption. I chose the 12F1512 which supports a power supply voltage range between 3 V and 5 V.
For the ultrasonic sensors, there is a plethora of choices. However, few of them may be powered with a voltage between 3.3 V and 5 V. The Y401 and US100 models do allow it. The sensors are configured to measure the distance by the width of the return pulse. The other option (not used here) is to get the distance on a serial type link.
The circuit diagram is very simple (**Figure 2**): two outputs GP1 and GP2 of the µC activate the triggers of the two ultrasonic sensors and a diode OR gate allows measurement of the time until the return pulse. As the activation and distance measure are sequential in the software, an OR gate is sufficient, with a pull-down resistor.
Measurement is effected with the aide of Timer1 which counts as a function of the state of pin 4 (GP3, `T1GVAL`). The register of this timer is incremented when the pin is at '1' ; it does nothing when the pin is at '0'. The software makes a preliminary delay (while the signal is at '0') then measures the time during which the signal is at '1' or stops the measurement if the set time limit (`TIME_LIMIT_WAIT`) is attained (in the case where no obstacle is found by the ultrasonic sensors). The diode OR gate is thus connected to this input which starts Timer1 counting. The time is read from the register of this timer when the signal from the OR gate returns to '0'.
The distance d is calculated with the following formula:
$d = t \times 34 / 200$ [cm]
Where t is the duration of 'high' signal in µs. The vibrators are activated by two BS170 FET transistors.
As stated previously. the module can be powered from any lith-



Figure 1. The author's prototype *in situ*, with a sensor to the right and one to the rear. The wires connecting the vibrators on both sides of the frame of the seat can be seen.

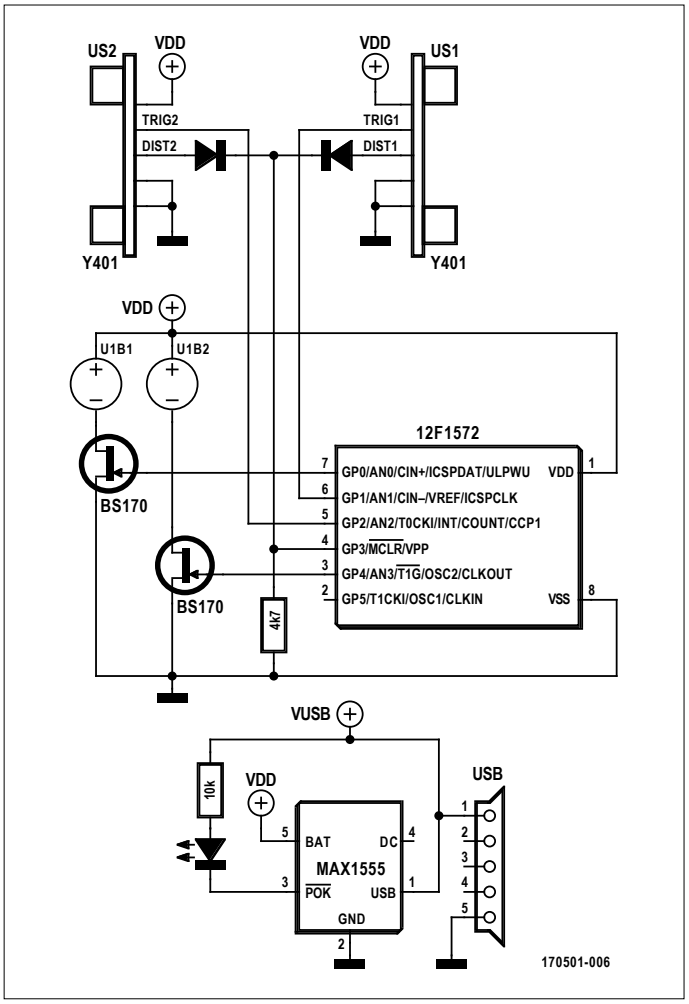


Figure 2. Circuit diagram of the radar, simple and effective.

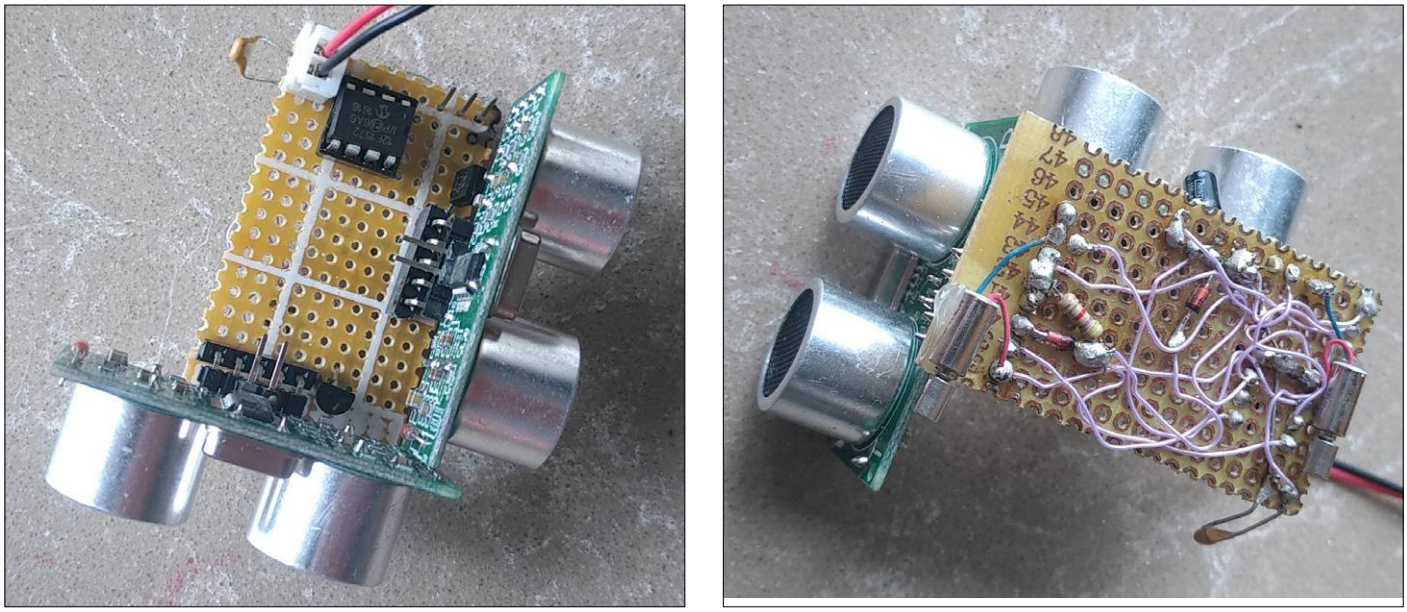


Figure 3. DIY wiring on both sides of a matrix board.

ium battery; a MAX1555 Integrated circuit controls charging from a USB port.

The software

The software is very simple. Every two seconds, it activates the right hand sensor, measures the distance, activates the right hand vibrator as a function of the obstacle detected, then does the same operation with the left hand sensor.
If a third sensor is wired for the centre (output GP5) the software alerts by activating both vibrators at once.
The distances for the alerts are predefined in the software.
The software was developed with the latest version of MikroC, the only one which recognises the 12F1512 µC. The free version is adequate because the final program size is small. I used the PICkit3 for programming the µC.

Mounting on the bike

This is not the simplest task. The radar board and the vibrators must be installed:
The radar is placed under the seat. I fixed it with cable ties. Care must be taken that it is exactly horizontal to avoid 'sensing' the road.

- As explained previously, the radar should be oriented suitably for the traffic direction in the country of use.
- For the vibrators, find the simplest position under the seat.
- The two vibrators are clad in thermal sheath and fixed with a cable tie on the two metal rods of the frame of the seat. after many tests I found that this is where there is the least movement. On the front of the seat, with the movement of the legs, the vibrations are not felt. If adding wiring to the bike is no problem, mounting them on the handlebars is the best solution.
- This little safety device could be further developed (sensors, power...).

Up to you to have a play!

170501-03



Web link

[1] Article support page
www.elektormagazine.com/170501-03

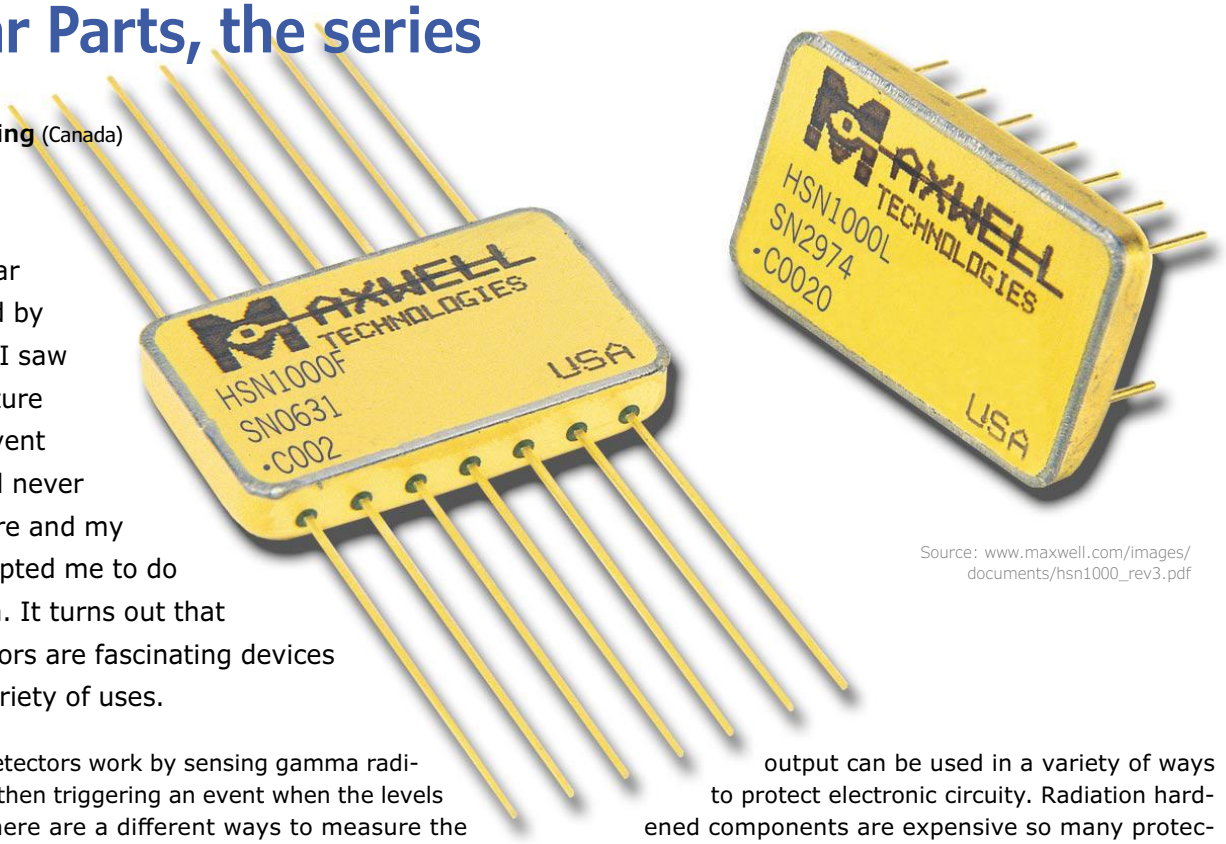
@ WWW.ELEKTOR.COM

→ Book: "Les microcontrôleurs PIC pour les débutants", By Rémy Mallard, (in French only)
www.elektor.fr/les-microcontroleurs-pic-pour-les-debutants

Nuclear Event Detector

Peculiar Parts, the series

By **Neil Gruending** (Canada)



Source: www.maxwell.com/images/documents/hsn1000_rev3.pdf

Today's Peculiar Part is inspired by a Twitter post I saw that had a picture of a nuclear event detector. I had never seen one before and my curiosity prompted me to do some research. It turns out that nuclear detectors are fascinating devices that have a variety of uses.

Nuclear event detectors work by sensing gamma radiation levels and then triggering an event when the levels get too high. There are a different ways to measure the radiation levels (like Geiger counters) but conveniently gamma radiation will introduce a proportional photocurrent in semiconductor PN junctions, which is what we will focus on here.

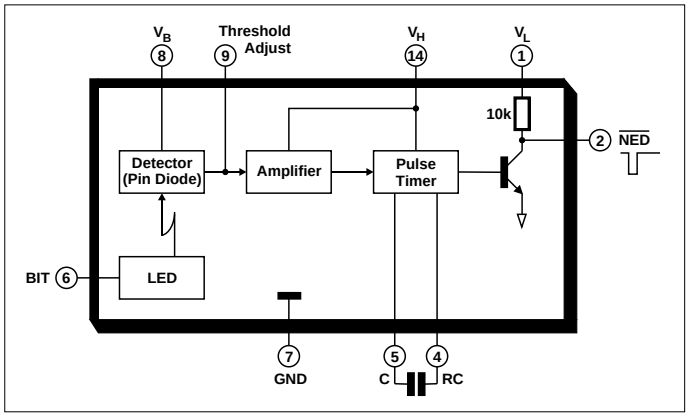


Figure 1. HSN1000 block diagram [1].

The HSN1000 by Maxwell Technologies (block diagram in **Figure 1**) is one such device that uses a PIN diode to detect gamma radiation levels as shown in. A PIN diode is similar to a regular PN diode except that it has an extra undoped region called the intrinsic region sandwiched between the P and N regions which helps to increase the diode's sensitivity and speed when detecting nuclear events. The PIN diode output is then fed through an amplifier which triggers a pulse timer to generate an active-low event output called NED.

And this is where it really gets interesting because the NED output can be used in a variety of ways to protect electronic circuity. Radiation hardened components are expensive so many protection circuits will put the system into a safe state to minimise the number radiation hardened components or will just shut everything off for the duration of the event. Otherwise the semiconductors could experience soft errors like erratic CPU operation or even latchup and junction burnout. Latchup is a serious concern because the IO logic pin becomes stuck in a state which may not clear after a reset. Junction burnout is caused by the types of transistors used as well as the amount of photocurrent generated by the radiation.

The main use of nuclear event detectors is for military applications, but their gamma sensing abilities also make them useful for spacecraft and satellites. Historically spacecraft have used various shielding materials to protect their critical systems but that increases their weight and the launch costs. Today, it's more common for the spacecraft to use radiation hardened components for the typical radiation exposure and then use nuclear event detectors to protect the systems against large radiation events. This method reduces the amount of shielding which lowers the weight and reduces the launch costs.

Unfortunately, these types of parts are very expensive and difficult to find 'surplus' but that's probably a good thing since you would need a gamma radiation source to trigger it. But it's always interesting to see how simple parts like a PN diode can be used in unusual ways. ⏮

(180575-B-01)

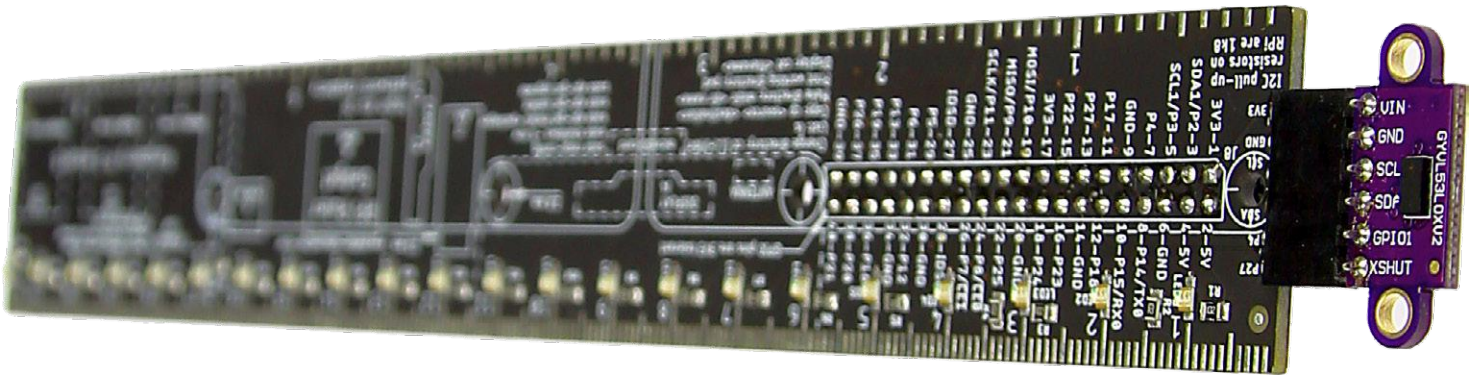
Web Link

[1] Maxwell Technologies HSN1000 datasheet: www.maxwell.com/images/documents/hsn1000_rev3.pdf

eLABS PROJECT

The Raspberry Pi Ruler Gadget

fun with a time-of-flight sensor



By **Clemens Valens** (Elektor Labs)

Although designed as a giveaway gadget for trade shows and similar events where Elektor is attending, this ruler can electronically measure distances of up to two metres with the help of a VL53L0X time-of-flight (ToF) sensor.

Quick Features

- Many practical RPi mechanical details
- VL53L0X Time-of-Flight (ToF) sensor
- 22 LEDs
- I²C connector
- Basic ruler free to Elektor booth visitors

If you happen to have one of these rulers because you visited the Elektor booth at electronica in Munich, Germany last November, or at the WoTS in Utrecht, the Netherlands, you will have noticed that it has solder pads on it as well as space for two connectors. That's because the ruler can be equipped with 22 LEDs (in an 0805 SMT package) with current limiting resistors (0805 package; 1 kΩ will do), a 40-way Raspberry Pi extension socket (16 mm high minimum if used with a 'normal' Pi), and a 6-pin I²C header/socket (or wires, whatever you prefer) — see **Figure 1**.

Admittedly the RPi ruler by itself has limited usefulness at best. Still, it does have lots of practical mechanical and electrical information about the Raspberry Pi on it, and it does work fine as a ruler. Also, it can be turned into a real high-tech gadget if you mount all the parts, connect a VL53L0X time-of-flight (ToF) sensor to the I²C connector, and of course plug the ruler onto your RPi.

The VL53L0X time-of-flight (ToF) sensor

The VL53L0X sensor used in this project is pretty amazing. Manufactured by ST, its datasheet claims that the laser-ranging module is housed in the smallest package on the market (this was in 2018). It provides accurate distance measurement irrespective of the target reflectance, and can measure absolute distances up to two metres. The sensor does not estimate the distance to an object by measuring the amount of light reflected back from it — this amount depends highly on the object's colour and surface — but instead precisely measures the time it takes for the light to travel to the object and back (the time of flight, so to speak).

Single-photon avalanche diode

The reflected light is detected by a single-photon avalanche diode or SPAD. This is a kind of diode similar to the 'normal' avalanche photodiode (APD), except that it needs a single incident photon only to make it break down (avalanche) and produce a current of a few milliampères. When the avalanche is quenched, the SPAD can recover and detect a second photon. The main difference between an SPAD and an APD is that the latter is a linear amplifier for optical signals whereas the first is a trigger device.

Two metres is the best the VL53L0X can do, and in real-life applications the maximum distance depends on ambient light conditions and the reflecting surface it is pointed at. Indoors up to one metre is

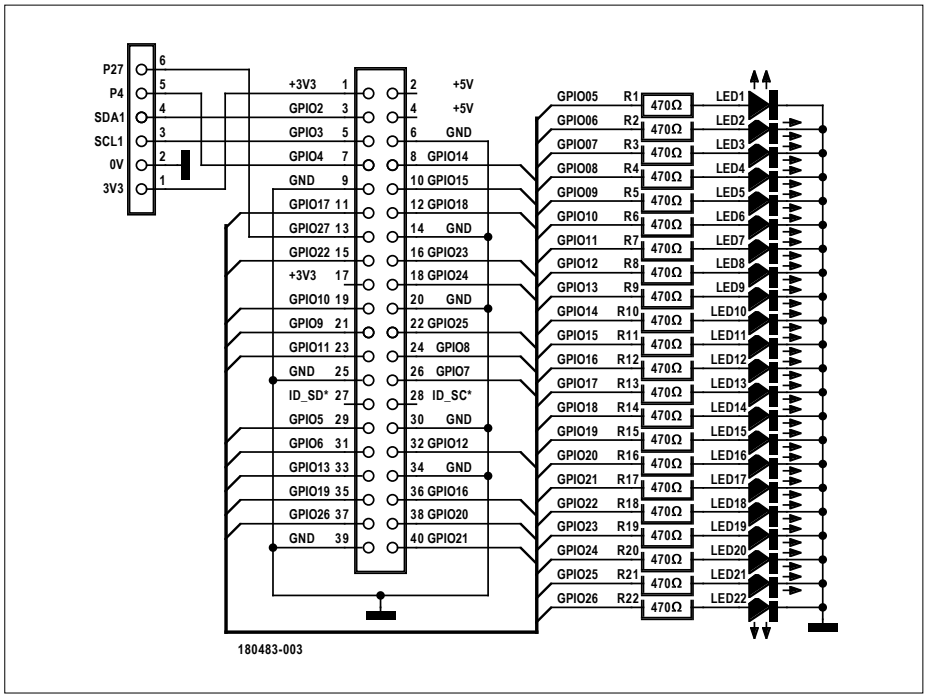
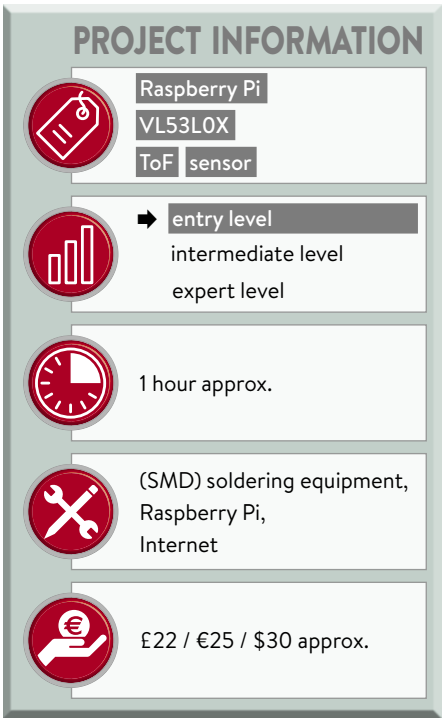


Figure 1: A mere 22 LEDs with current limiting resistors together with two connectors make up the schematic of the RPi ruler.



easily achievable; outdoors it may be a bit less. The sensor's accuracy is better than 10% in most cases, but again, this depends on usage conditions.
The data produced by the VL53L0X can be accessed over a convenient I²C interface.

Convenient break-out board

The interesting module was quickly singled out by (Chinese?) breakout board providers and is easily found on the Internet in a practical form (**Figure 2**). The manufacturer has put a lot of effort in creating software for the sensor, which together with the breakout boards has resulted in several open source driver libraries for Arduino and Raspberry Pi.

An electronic RPi ruler

In this project we will use the LEDs on our RPi ruler to display distance. Since there are only 22 LEDs on the ruler the readout is limited to 22 centimetres. Fortunately, more elaborate display schemes can be conceived using LED multiplexing. For this reason I mounted red LEDs for the values 10 and 20, and green LEDs for the rest. Note that the distance in millimetres is also printed to a display attached to the Pi's HDMI port. Also note that this project can be built without the ruler; just connect the VL53L0X to the I²C port of the Pi.

The software was written on a Raspberry Pi 3 model B V1.2 but should also work on other Pis with a 40-pin extension connector. Plug the ruler on the Pi before powering it up.

Installation of the software on the RPi

The VL53L0X driver was written in C++, which is why the software for this project is in also in C++. Python libraries for the sensor exist as well, so if you prefer to continue in Python, you can. Just to be clear, the following steps are done on your Raspberry Pi, not on a Windows/Linux/macOS computer:

- Download and install wiringPi [1]. This library is used to access the Pi's GPIO pins in C++ so we can control the LEDs.
- Download and unpack the VL53L0X driver STSW-IMG005 from ST, get it from the 'Get Software' section at [2].
- Download and unpack the Raspberry Pi adapter VL53L0X_rasp by Cassou (Mathieu Cassard) [3]
- Download the Makefile from [4] and copy it into the folder VL53L0X_rasp (overwrite the existing file).
- Download the file rpi_ruler.c from [4] and place it in the folder VL53L0X_rasp/examples

Supposing that you followed the steps above and that you work from your home directory, do this:

```
cd VL53L0X_rasp
make API_DIR=~/VL53L0X_1.0.2/
make examples
   API_DIR=~/VL53L0X_1.0.2/
```

(the last two lines must be entered as one line).
There should be no warnings or errors. To run the RPi ruler program type:

```
./bin/rpi_ruler
```

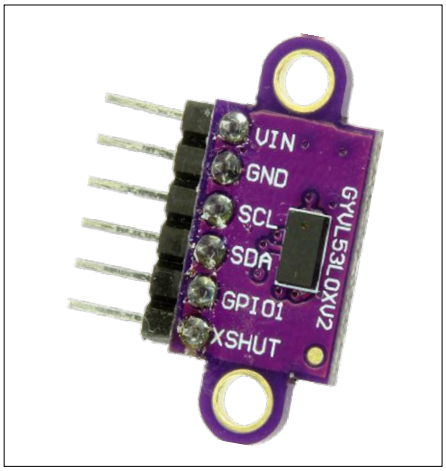


Figure 2: A particularly popular breakout board (BoB) for the VL53L0X Time-of-Flight sensor.

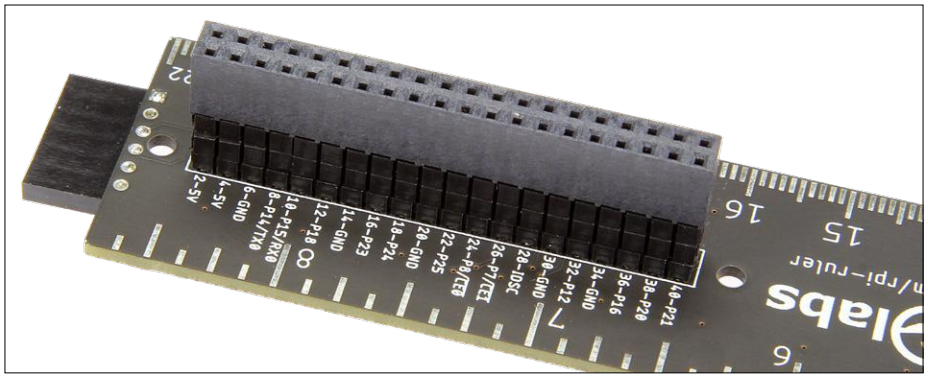


Figure 3: If the ruler is to be plugged on a 'normal' Raspberry Pi, this connector must be 16 mm or taller.

Displaying results

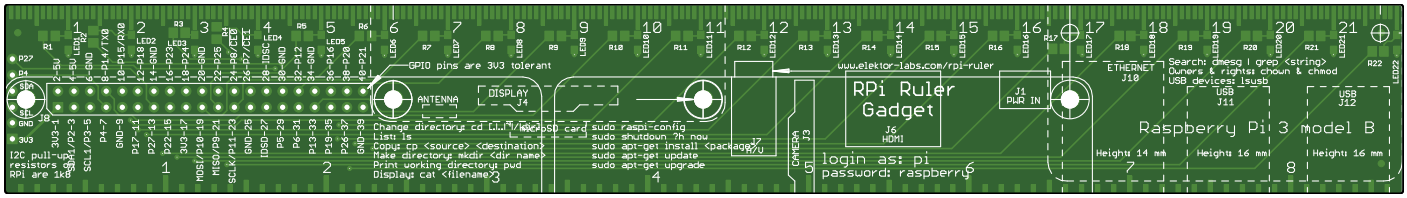
Having connected a display to the RPi you should see measurements scrolling past. Waving your hand or an object in front of the sensor should result in a plausible distance in millimetres. Nothing in front of the sensor will produce the value 8192 (i.e. out of range). If an object is within 22 cm from the sensor, the distance is shown by the LEDs. Because in this case the distance will often be a non-integer centimetre value, two LEDs may light up at the same time. Their brightness is determined by the decimal value with the LED closest to the decimal value shining brightest. Example: when the measured distance is 12.4 cm then LED12 will have a brightness of 60% and LED13 will light at 40% (sure, for optimum readability this should use a logarithmic-law scale instead of linear). Brightness is controlled by the function `softPwmWrite` available in wiringPi.

Notes

For 'best results and user experience', please carefully read the following:

- Use a 16-mm high (or taller) socket for the Raspberry Pi extension connector because the ruler has to pass over the Pi's USB connector (**Figure 3**). This remark does not concern the Raspberry Pi Zero because it doesn't have tall connectors.
- The VL53L0X sensor can be powered from a 2.6 to 3.5 V supply. However, VL53L0X break-out modules tend to have an on-board low drop-out (LDO) voltage regulator to allow powering from the old 5 V. The I²C connector on the ruler only supplies 3.3 V. That's not a problem since many LDOs simply pass the voltage when it drops below their minimum input voltage. If for some reason you really need 5 volts, simply steal it from the RPi's extension connector.
- Your VL53L0X sensor may have an (often yellowish) protective film stuck on it. If it does, remove it before using the sensor. ⏮

(180483-B-01)



COMPONENT LIST

Resistors

R1-R22 = 1kΩ, 0805

Semiconductors

LED1-LED22 = green, 0805

Miscellaneous

40-way (2x20) pinheader socket, 0.1" pitch, 16 mm tall (e.g. Samtec ESQ-120-33-T-D)
6-pin SIL pinheader, 0.1" pitch
6-way SIL pinheader socket, 0.1" pitch
VL53L0X ToF sensor module
PCB # 180483-1 from the Elektor Store

@ WWW.ELEKTOR.COM

→ **180483-1: RPi Ruler Gadget, bare PCB**
www.elektor.com/elektor-raspberry-pi-ruler

→ **180483-71: Kit of parts**
www.elektor.com/rpi-kit-180483-71

→ **VL53L0X Time-of-Flight (ToF) sensor**
www.elektor.com/tof-distance-sensor

Web Links

[1] wiringPi: http://wiringpi.com/

[2] Official VL53L0X driver: www.st.com/en/embedded-software/stsw-img005.html

[3] RPi VL53L0X driver: https://github.com/cassou/VL53L0X_rasp

[4] Makefile and main program: www.elektormagazine.com/labs/rpi-ruler

productronica
fast forward
the start-up platform
powered by Elektor

COMPETE TO LAUNCH YOUR STARTUP @ PRODUCTRONICA 2019

p-ffwd 2019 – Participate Now!

November 12-15, 2019
Munich



Further information at:
www.elektormagazine.com/p-ffwd

Productronica Fast Forward is brought to you by

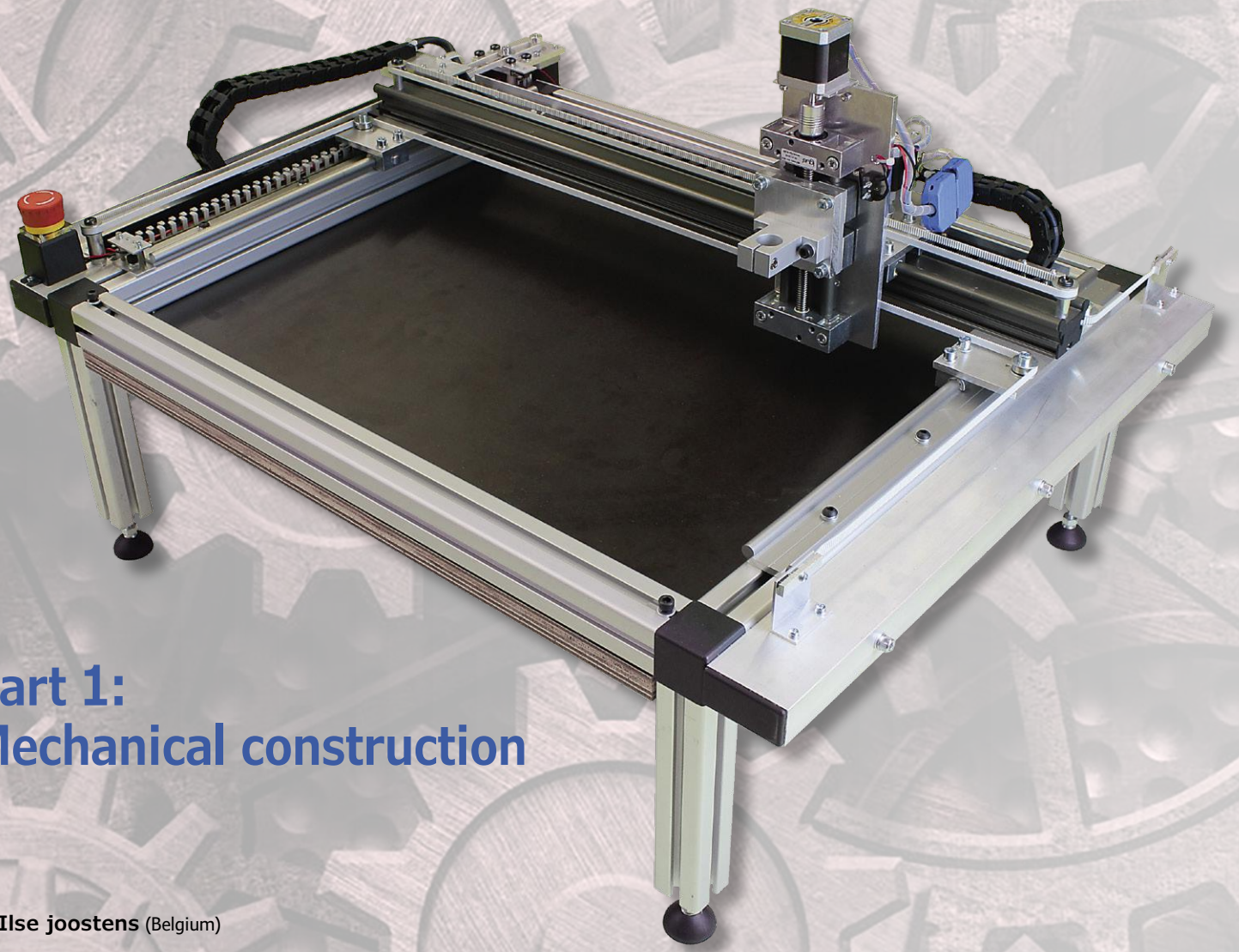
productronica

elektor
INNOVATION STARTUP TRADE

eLABS PROJECT

‘Wolverine’ CNC Gantry Robot

Multifunctional Powerhouse



Part 1: Mechanical construction

By **Ilse joostens** (Belgium)

Many hobbyists, fanatics and makers struggle with the mechanical aspects of their projects. Designing a circuit board and assembling it or programming a microcontroller is generally not a problem for a dyed-in-the-wool electronics enthusiast, but once the circuit has to be housed in something, the enthusiasm typically cools considerably. And then we are not even considering making relatively simple mechanical parts, such as a gear or a cover plate, yourself. With the rise of 3D printers, affordable CNC machines and online service providers, mechanical design has become a lot more accessible. With the CNC gantry robot presented here, we would like to help you further along the path of this fascinating world.

Specifications

- work surface area about 50 × 30 cm
- linear glides with polymer bearings, insensitive to contamination
- maximum speed of X and Y axes: 15000 mm/min, Z-axis: 750 mm/min
- repeatability: about 0,2 mm
- nema-17 stepping motors
- driven via reinforced GT2 toothed belts (X and Y axes) and screw spindle (Z-axis extension)
- Z-axis extension suitable for drilling, milling and engraving in wood, plastics and non-ferrous metals (aluminium, bronze). Using Proxxon LBS/E spindle (5,000 – 20,000 rpm, 100 W)
- professional drivers and software from Eding CNC (cnc310 controller board)
- stepping motor drivers based on Trinamic TMC2160 controller IC's for silent operation
- 2.5D software extendible to full 3D
- multi-functional design, can be extended for other applications
- can be extended with a pendant (= console with buttons and wheel for manual control)
- maximum power consumption without drill/grinder: about 100 W
- total mass of the machine including Z-axis extension and Proxxon LBS/E drill/grinder: 24 kg

If you have ever contemplated designing mechanical parts yourself, then the chance is great that you initially considered using a 3D printer. 3D printers have been around for a while, but thanks to the availability of affordable machines, this technology has become hugely popular in a short time. The vast majority of these machines use FDM technology (Fused Deposition Modelling), where a plastic filament is melted and extruded in layers. Machines based on stereo lithography do also appear in the hobby lab.

Disadvantages of 3D printers

Despite their great popularity, 3D printers do have a number of disadvantages. Not only are the materials (filament, UV resin, …) relatively expensive, it is also a slow process that is not really suitable for the production of larger quantities. Furthermore, the FDM 3D printers, in particular, have quite a rough surface finish, that sometimes requires final processing. It is absolutely not advisable to 3D-print objects that will come into contact with food, because of their rough surface finish they cannot be cleaned sufficiently well after use.

Compared to 3D printers and their sophisticated technology, the simple CNC machines look more like a primitive drill press on steroids. They are nevertheless capable of producing parts much quicker than 3D printers, with a better finish and a greater choice of materials. To give you an idea, the typical small job on a CNC machine takes some 5 minutes, while with a 3D printer you would have to think more in hours. It is obviously not our intention to criticise 3D printers, for complicated 3D shapes or prototyping they continue to be a good choice.

The popularity of 3D printers is a great advantage for our project: these days it is much easier to obtain mechanical parts such as glides, bearings, toothed belts and couplings, compared to some 10 years ago. Our CNC gantry robot is therefore built from such easily obtainable components that you can simply order them online (via eBay, for example). In this first part we examine the mechanical construction of the machine.

Frame and general construction

Compared to a standard desktop CNC gantry milling machine, our gantry robot has a remarkable appearance. While gantry milling machines are normally built around a fixed work surface, we have selected a sturdy aluminium frame. The 40 x 40 mm aluminium profiles not only contribute to the general stiffness of the machine, but also increase the usage options and offer more opportunities for future extensions.

The work surface of our prototype consists of 20 mm thick concrete-forming plywood that thanks to the frame construction is easy to remove. Not only can an appropriate work surface be selected as a function of the application, but the work surface can also be mounted at a different height. It is even possible to position the machine without a work surface over the work piece, for instance

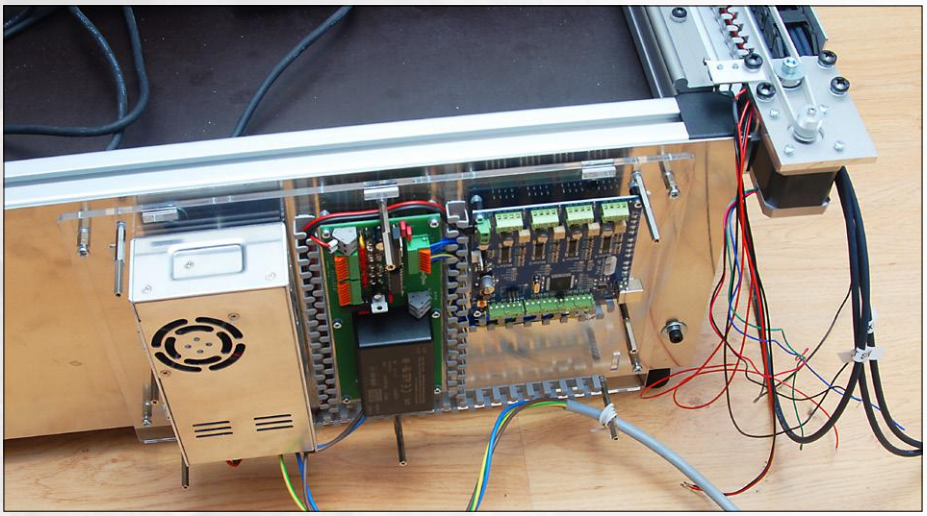


Figure 1. The electronics find a nice home at the back of the machine.

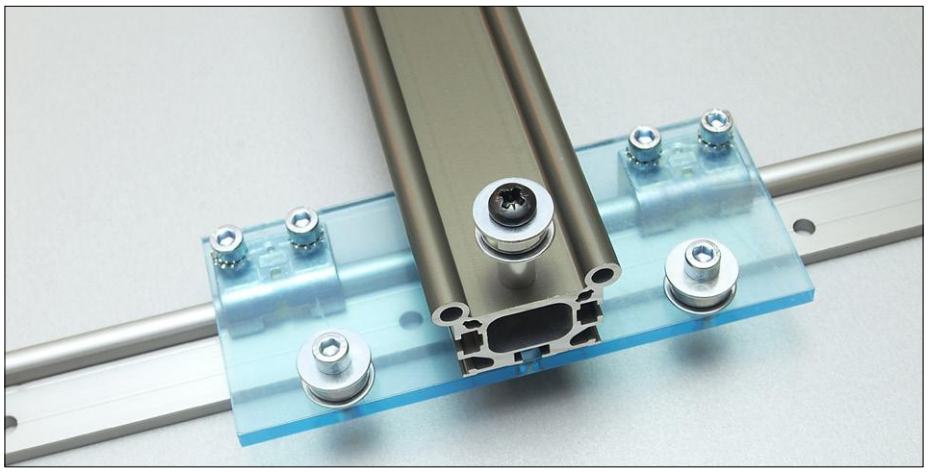


Figure 2. For the Y-axis we employ two single rails type WS-10.

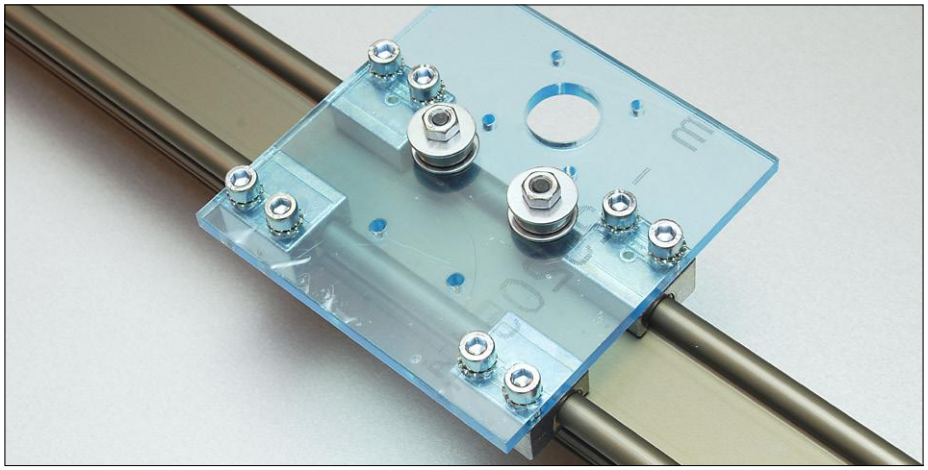


Figure 3. For the X-axis we use a double rail of the type WSX-10-40.

to engrave a larger object. For example, a potential future extension could be a height-adjustable table for CO2 laser engraving and laser cutting or 3D printing.

In the event the machine is used as a pick & place machine for populating circuit boards with SMD components, then the frame can be used for attaching the reels that contain the components and even SMD feeders, if that is desired.

Because of the frame, the electronics for the control and the power supply can be mounted on the back of the machine (**Figure 1**). This makes the machine not only more compact, but the electronics is also better protected from flying swarf and dust. The adjustable feet also allow the machine to be placed securely on an uneven surface. It is, of course, also possible to build the machine without a frame, using only a piece of plywood as the base.

Despite the complexity, the machine us still reasonably easy to build. It can be built without specialist tools and machines. However, some knowledge of working with metal and (access to) a drill press or drill stand, a drop saw with a blade for aluminium, a thread tap set and a step drill bit are essential. You can optionally also make a work surface from MDF with T-slot grooves (spoilboard), for fastening the work pieces combined with Proxxon step clamps (24257). This can then be attached to the plywood and is easily replaced when it becomes damaged. For this you will need a router with a T-slot bit. The mechanical drawings, component list and a construction manual are available as downloads from the article support page on the ElektorLabs Magazine website [1].

X and Y axes

Guide rails and bearing blocks

For the guide rails for the X and Y axes we chose the profiles and bearing blocks from the DryLin-W system from the German company Igus [2]. These hard-anodised aluminium profiles and polymer bearings offer a very good price/performance ratio and are robust and not very susceptible to dust and dirt. In addition, the plastic sleeve in the bearing blocks can be easily replaced, should that ever become necessary. An additional advantage is that the alignment is a little less critical, compared to ball-bearing guide rails, which makes the assembly of the machine much easier.

For the Y-axis we used two single rails, type WS-10, with a length of 500 mm, see **Figure 2**, while the X-axis traverse is built around a double rail, type type WSX-10-40, with a length of 750 mm (**Figure 3**). Since the X-axis traverse has to be self-supporting across a reasonable distance, we specifically selected a double-height rail. Should you wish to build a bigger machine, then the design can be, within certain limits, scaled up somewhat.

For the bearing blocks of the X-axis you have, depending on the desired applications for the machine, the choice of type WJUME-01-10 with adjustable play, preload bearing blocks type WJ200UM-01-10-P40 / -P90 / -P140 or precision bearing blocks type WJUM-01-10K with JUMO-01-10K bearing sleeve (for the Z-axis, bearing sleeve type JUMO-01-10K needs to be used, because the present implementation of the Z-axis was specifically designed for routing).

The bearing blocks with adjustable play and the preload bearing blocks ensure, provided they are adjusted properly, a very smooth glide for the X-axis, where the drive motor does not need to provide a lot of power. This is ideal for applications where there are only small forces along the X-traverse, such as laser engraving/cutting, 3D printing, pick & place, ... If you would like to use the machine as a CNC milling machine, then we recommend the precision bearing block type WJUM-01-10K. These bearing blocks give the X-axis more rigidity (less play), which is absolutely necessary when mounting a Z-axis with a routing spindle, and even more so if you would like to work with somewhat harder materials.

A disadvantage of these bearing blocks is the stiffer glide of the X-axis. But the

motors do have sufficient power because of the drive belts (pulley system), even at higher speeds. You can, optionally, carefully lubricate the bearing blocks with a small amount of gun oil (type Ballistol GunCer, available from the better gun stores).
The bearing blocks for the Y-axis are a hybrid type (WJRM-01-10) that both roll and glide along the rail, which is more stable at higher speeds (> 10 m/minute).

• Drive system and motors
Because the machine is a multifunctional design, and particularly for pick & place and laser engraving we wanted a fast machine, which would have been more

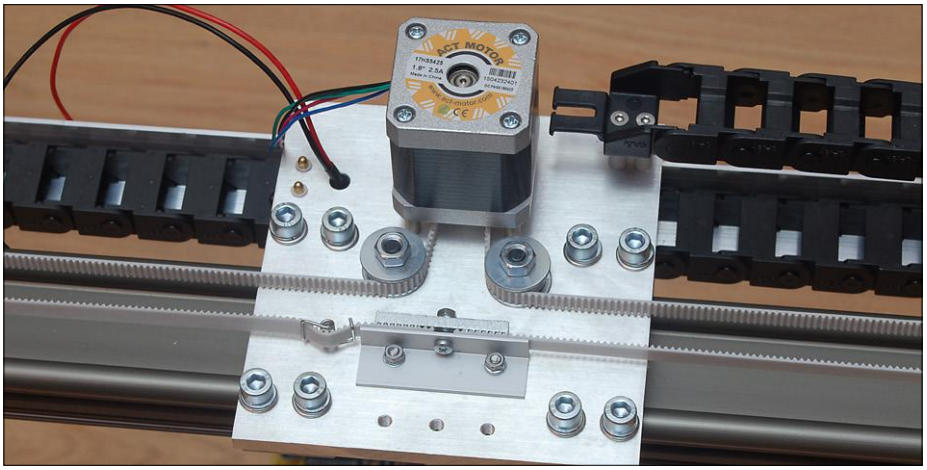


Figure 4. The force of the motors is increased through a pulley system.

The force is doubled through a pulley system
Steel-reinforced GT2 belts are much stronger than neoprene

difficult to achieve with spindle-driven machines, we chose steel-reinforced, 6-mm GT2 toothed belts for the drive of the X and Y axes. These are normally white in colour and are very strong compared to the regular black neoprene toothed belts. Typical for the belts with the GT2 profile is also the very small amount of play. With this it is important to tension the belts quite tight.
GT2 belts are easily found on eBay from sellers who offer materials for the DIY construction of 3D printers. For the guiding of the belts, the readily-available 606 2RS ball bearings are used, mounted between M6 washers with an outside diameter of 20 mm.
For the motors we selected standard *nema-17* stepping motors with a 5 mm shaft diameter. By using a pulley system with the toothed belts (see **Figure 4**), the force of the motor is doubled, which makes them more than powerful enough for light milling work and all other applications that the machine was designed for. With a holding torque of 0.58 Nm for the stepping motors (type 17HS5425) and a diameter of the pulleys of 12.8 mm we theoretically arrive at a maximum force of:

[2 × (0.58 Nm × 71 % / 0.0064 m) = 128.7 N

on the X and Y axes at a low speed. Here we assumed a theoretical dynamic torque of 71 % of the holding torque, when driven at a low speed through micro-stepping. In practice we are not driving the motors with their maximum rated current and there are also losses through the wiring, the mechanical transmission and friction in the linear slides, so that some 60 to 70 Nm is a lot more realistic. To give you an idea, when we hold on to the X-traverse at low speed, the machine is quite capable of moving itself along the floor.
Bigger motors have as their biggest disadvantage the larger mass and therefore more inertia at faster acceleration and deceleration rates. Furthermore, they often require more expensive stepping motor drivers that can supply more current.
The Y-axis is driven by a single stepping motor. Such axes are quite frequently driven with two stepping motors. This goes well as long as the two motors remain synchronised to each other. Should one of the motors miss a step then the traverse can start to run at an angle with all the problems that entails.
With the GT2 pulleys with 20 teeth that have been used here, the displacement of the X-axis is about 20 mm per revolution of the stepping motor. For the Y-axis the thickness of the toothed belt also plays a role and with our prototype we arrived at a displacement of 20.6 mm per revolution. These are guide values and it is important to calibrate the finished machine using verniers.
The carrier plates for the ball bearings and slide bearings and the motors are made from 6 mm thick T6 aluminium. This material is relatively easy to work with and gives the machine as a whole sufficient rigidity. If the machine will only be used for light applications, then you could also use 6 mm HPL plate instead of aluminium. This is much easier to work with.

• Z axis
The carrier plate for the X-axis motor is provided with three holes with an M5 thread for mounting various extensions. Our first extension is a Z-axis with a routing spindle for routing, engraving and drilling. Because with this extension, precision is more important than speed, a spindle-driven linear glide is used here. This can be bought from Igus as a complete assembly using order code SLW-1040 (see **Figure 5**). This gives you the option to choose a few parameters, such as the stroke length. For this 60 mm is usually sufficient. It is important no to choose a stroke length that is too large, because of stability. Because the axial load is facing downwards as a consequence of the weight of the routing spindle, a preload nut on the spindle drive to take up the play is therefore not necessary.
For the bearing blocks of the SLW-1040

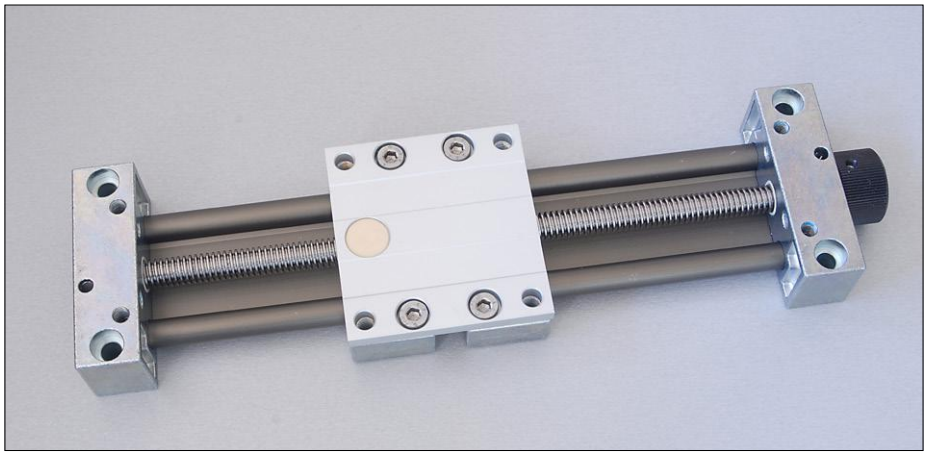


Figure 5. In the Z-axis a spindle-driven glide of the type SLW-1040 ensures a high accuracy.

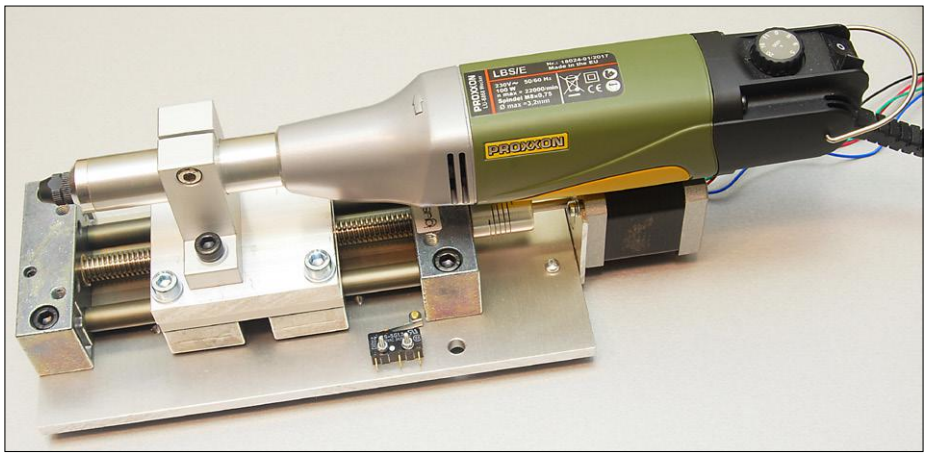


Figure 6. This Proxxon LBS/E drill/grinder is not the only suitable candidate.

you need to select the new WJUM-01-10K bearing material (special order from Igus), otherwise there is too much radial play in the milling spindle.

The drive, just as with the other axes, is provided by a nema-17 stepping motor via a flexible coupling. Theoretically, the force with a displacement of 2 mm per revolution is much higher than for the X and Y axes, but we need to take into account the maximum allowable axial load of the SLW-1040 (700 N static) and the load of the Z-axis as a whole on the motor carrier plate and the glide bearings of the X-axis. It is therefore a good idea to limit this to about 20 N in practice, which is sufficient for most applications. An SK20 holder for 20-mm shafts is used for mounting the drill and milling spindles.

For our prototype we used a Proxxon LBS/E drill/grinder (**Figure 6**). This is suitable for the lighter milling work. But brands such as Kress also supply suitable models. Moreover, don't let anything prevent you from mounting a real milling spindle, with or without a suitably adapted holder.

The mounting of the Z-axis on the carrier plate of the X-axis is done using four M5 bolts and a piece of square aluminium bar. With this, the Z-axis can be mounted at five fixed positions with different heights. If you are using the Proxxon LBS/E, then you can also adjust that in height, which gives you a great deal of flexibility.

Wiring and limit switches

The cables that connect the motors and limit switches to the electronics are in constant motion with small bend radii and subject to contamination. They therefore have to meet high demands that ordinary cables cannot meet. The cables used here are of the type ChainFlex (CF9.02.06, see **Figure 7**) with 6 conductors, each with a cross-section of 0.25 mm^2, which is just sufficient for nema-17 motors and currents up to 2.5 A. The copper wires in this type of cable are formed as a spiral around a plastic core, so that when the cable is bent as a whole, less force is transferred to the copper conductors. The plastic core and plastic fibres between the conductors and outer sheath provide the necessary reinforcement and resilience against pulling forces. The outer sheath withstands contamination from common chemical products, oils and greases. The manufacturer guarantees fault-free operation for at least 5 million back and forth movements at temperatures from -35 to +100 °C.

It is not recommended to use thicker cables with a bigger copper cross-section. These are much stiffer when bending. For this same reason it was decided to use three cables with six conductors each and running side-by-side, instead of one or two thicker cables with more conductors.

To hold the cables in place, two flexible cable trays are used, also from Igus. The cable trays are actually too long for the path travelled by the cable, but this saves us an additional cable tray or conduit. In addition to the cables, the cable trays could also carry a PVC tube for, for example, suction, compressed air or cooling fluids. In any case, do not feed the mains power cable for the Proxxon drill/grinder through the flexible cable tray because this cable does not meet the requirements for such use!

Micro switches with a roller are used for the limit switches (**Figure 8**). For the sake of safety, the normally-closed contact is used, because a contact break is more secure than a contact close. Limit switches are placed both at the beginning and the end of both the X-axis and the Y-axis. The switches on the left and at the bottom function both as limit switches and as *homing* switches. The other switches have the function of emergency stop, should the machine, through a fault condition, attempt to move beyond the physical end of the X or Y axis. These emergency stop switches are optional and don't necessarily have to be fitted for the proper operation of the machine. Also from a machine safety standard these switches are not mandatory, because the forces generated by the machine are relatively limited.

For the Z-axis there is provision for only one limit/*homing* switch and that is for the upper position (Zmax). A limit switch for the lowest position doesn't make sense here. Before this becomes acti-

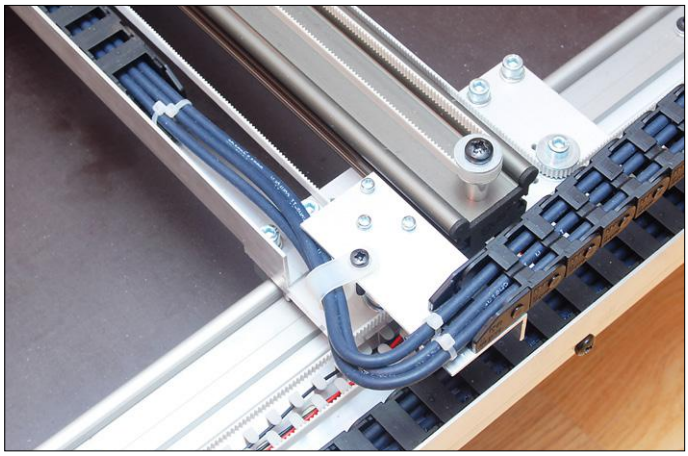


Figure 7. Suitable flexible cables withstand bending thousands of times.

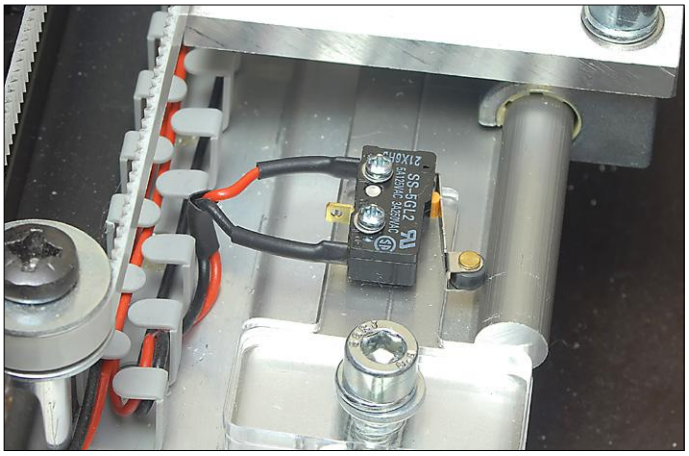


Figure 8. The limit switches with a roller are best fitted this way.



Figure 9. An emergency stop can prevent an incorrect command from becoming fatal to the milling tool.

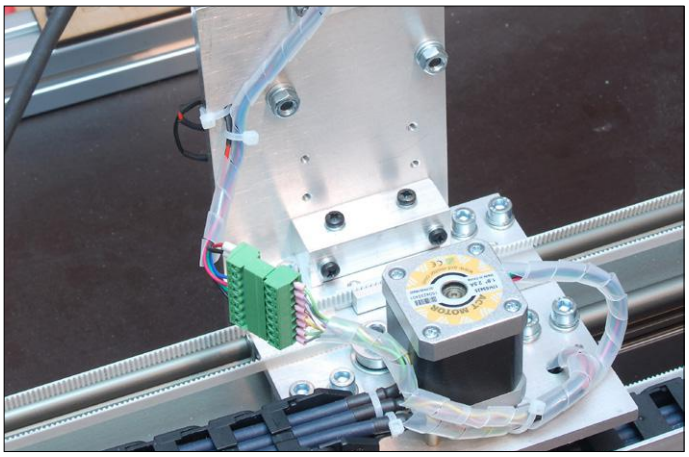


Figure 10. Using a multi-pole connector, the Z-axis is quickly swapped for another type.

vated the router is already well through the material.

Note that all homing switches are mounted length-wise with respect to the axes. Because this way the switches do not form a hard limit and the homing of the axis can be done faster. The instant that the homing switch is activated there is still some room to move and slow down the axis, without any risk of a hard collision with the switch.

Finally, in addition to a connection for a Z-probe there is also provision for and emergency stop (**Figure 9**). The appears perhaps superfluous, but the additional costs are immediately recouped if thanks to the emergency stop you can prevent a routing tool from breaking.

The Z-axis is connected to the rest of the machine through a multi-pole connector (see **Figure 10**). This way the Z-axis can be easily removed and exchanged with another extension, for example a laser head.

Electronic control and software: in part 2

In the next article about our CNC machine will cover the control, stepping motor drivers and the software in depth. We will then also discuss G-code, CAM software and the workflow to go from CAD drawing to finished part. ⏮

180246-04

Web links

[1] www.elektormagazine.nl/180246: www.elektormagazine.nl/180246

[2] www.igus.com: www.igus.com

[3] www.alu-messing-shop.de: www.alu-messing-shop.de

[4] de.act-motor.com: http://de.act-motor.com

[5] www.proxxon.com: http://www.proxxon.com

@ WWW.ELEKTOR.COM

→ Book: 3D Printing and CNC Fabrication with SketchUp
www.elektor.com/3d-printing-and-cnc-fabrication-with-sketchup

Operational Amplifiers in Practice

Part 2: FET-input opamps and wide-bandwidth applications

By **Burkhard Kainka** (Germany)

In many applications it is essential to use an opamp with extremely high impedance inputs. In these cases a standard bipolar device will not do the job, and instead we must use an opamp with FET (field-effect transistor) inputs.

The input leakage current of a typical LM358-type opamp is around 30 nA. If an input resistor of 1 MΩ is used, then this will have a voltage drop of 30 mV across it, which in many circuits is not acceptable. Opamps with FET inputs, on the other hand, have practically zero input leakage current: for example, a JFET-input opamp might have an input current some 1000 times lower than that of its bipolar brother. Typical examples of JFET-input opamps are the TL071 (single), TL072 (dual) and TL074 (quad). These devices use JFETs in the input stage and bipolar transistors for the rest of their circuitry, as **Figure 1** illustrates.

Even better input isolation can be obtained by using MOSFETs in the input stages. One example of this technology is the CA3140 (**Figure 2**). This device can run from a single or symmetrical supply of at least 4 V, and the input voltages can swing down to 0.5 V below the negative supply rail. The MOSFETs are very sensitive to excessively high voltages, and so Zener diodes are included in the circuit for protection. Input currents can be as low as 2 pA. The output stage of the CA3140 is conventionally constructed using bipolar transistors, and so the manufacturer calls the device a 'BiMOS' amplifier. As well as the single version, there is also a dual version, the CA3240.

Yet further improvement is brought by the CA3160 which uses CMOS transistors also in its output stage. This allows output voltage swings to within 10 mV of the negative and positive supplies ('rail-to-rail operation').

Voltage ramps

A FET-input opamp is a good choice when it is desired to generate slowly-varying waveforms such as voltage ramps. **Figure 3** shows a typical circuit of a ramp generator, which can, for example, be used to help plot characteristic curves automatically. The opamp is wired as an integrator, and the slope of the ramp is determined by the low charging current that flows towards the inverting input. The high input impedance of the opamp means that its input leakage current does not affect the operation of the circuit. The switch across the integration capacitor allows it to be discharged to begin a new ramp.

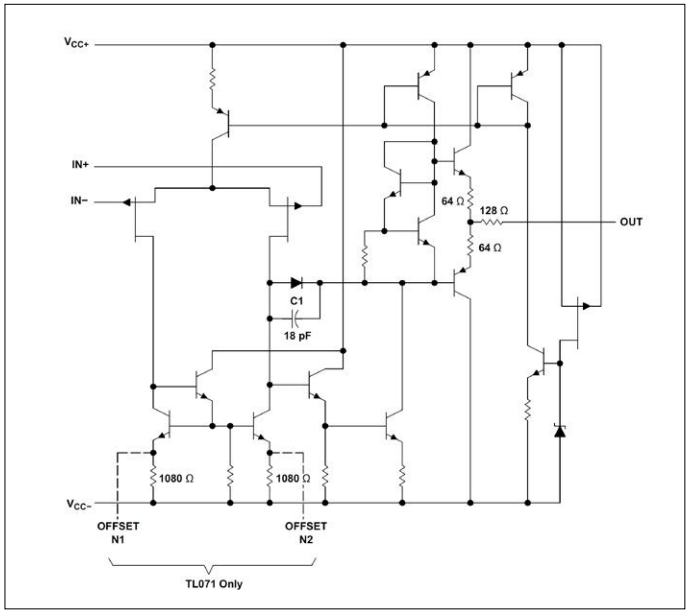


Figure 1. Circuit diagram of the TL071 (Source: Texas Instruments).

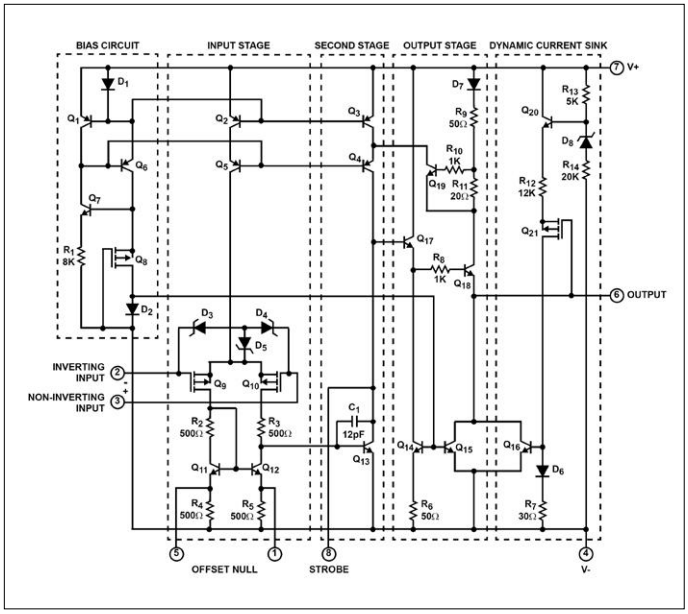


Figure 2. Circuit diagram of the CA3140 (Source: Renesas).

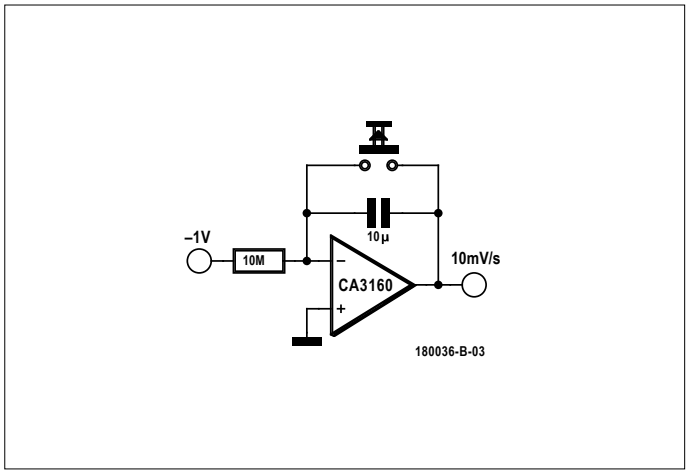


Figure 3. A ramp generator.

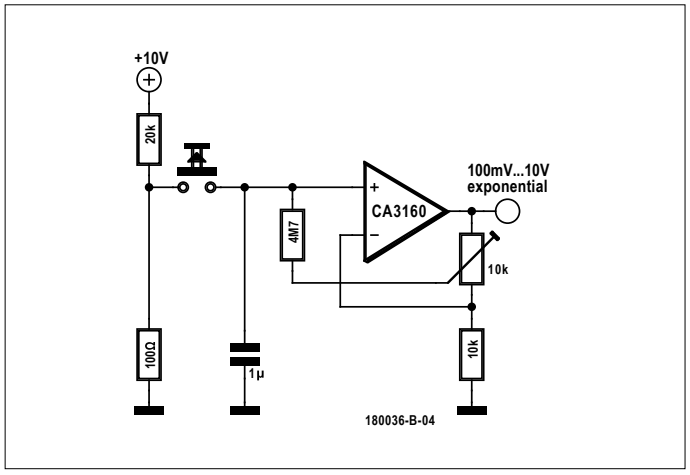


Figure 4. An exponential ramp generator.

In some situations an exponential ramp is required rather than a linear one. An example of how to achieve this is shown in **Figure 4**: this circuit could be used, for example, to drive a tone generator for audio testing purposes. The charging current for the capacitor is generated here from a voltage which is in turn proportional to the capacitor voltage. The result is exponential growth of the output voltage (see **Figure 5**). To initialise the circuit the capacitor must be charged to a non-zero voltage, say 50 mV. Then the gain of 2 provided by the amplifier will result in 100 mV at its output. The high input impedance of the BiMOS opamp allows the component values in the circuit to be varied over a wide range.

Many modern opamps are constructed entirely in CMOS technology and use no bipolar transistors at all. An example of this kind of device is the TLC272 CMOS opamp, which offers input leakage currents of less than a picoamp! **Figure 6** shows its internal circuit. Some CMOS opamps offer operation with both input and output voltages swinging all the way to the supply rails.

New opamps with ever better characteristics continue to be developed. Sometimes these improvements take the form of higher-impedance inputs or lower input offset voltages, or in true rail-to-rail operation at the inputs and outputs, or in the ability to drive low-impedance loads at the output. In many respects, therefore, we are getting closer and closer to the ideal opamp. However, it is often the case that obtaining particularly good characteristics in one area entails compromises on other points. So, for example, CMOS opamps with a very high input impedance often have a large input offset voltage, and parts optimised for low offset voltage only have a low bandwidth.

An example from the new generation of universal opamps is the TS914. This offers rail-to-rail operation on both inputs and outputs. The circuit diagram (**Figure 7**) shows that the device has a complementary differential amplifier stage at the input and a push-pull output employing MOSFETs. The part is a good choice in many applications and is used, for example, in the Elektor SDR (software defined radio). It has an input leakage current of just 1 pA and an input offset voltage of 5 mV. Its bandwidth is 800 kHz and it can drive loads with impedances down to less than 100 Ω.

Extremely high input-impedance opamps are often used in instrumentation amplifiers. As **Figure 8** shows, an instrumentation amplifier has a differential input and a single-ended output. It can therefore be used to make measurements on circuits without having to connect one side of the input to ground, although of course the voltage relative to ground on the inputs must lie within the permissible input voltage range of the opamp. The circuit comprises an adjustable differential amplifier with two impedance converters connected before it:

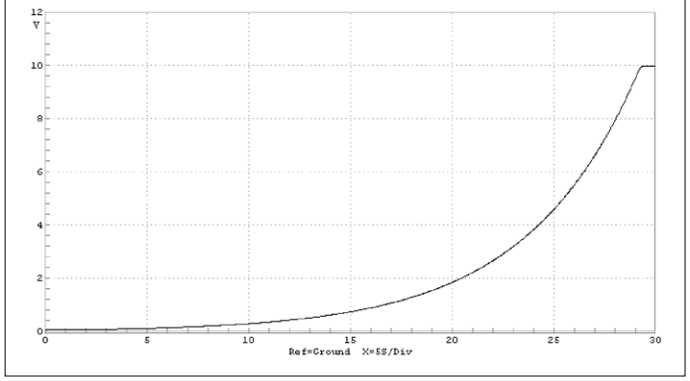


Figure 5. Output trace from the exponential ramp generator.

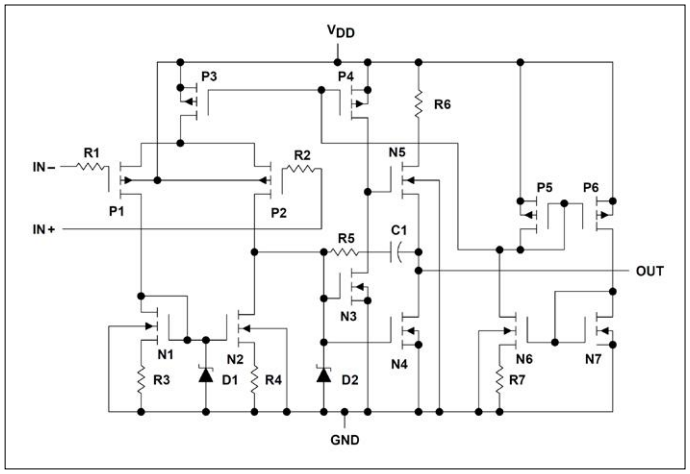


Figure 6. Circuit diagram of the TLC272 (Source: Texas Instruments).

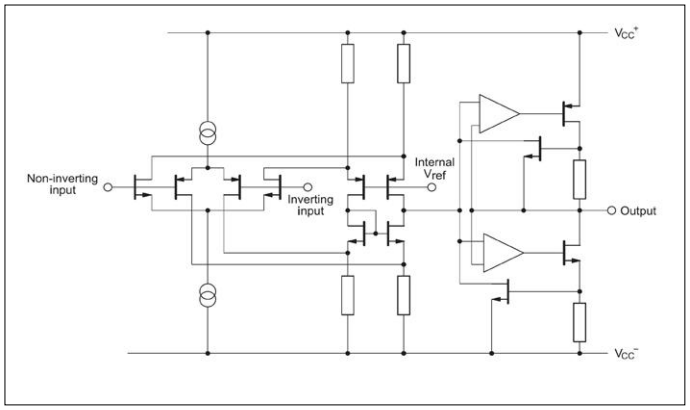


Figure 7. Circuit diagram of the TS914 (Source: STMicroelectronics).

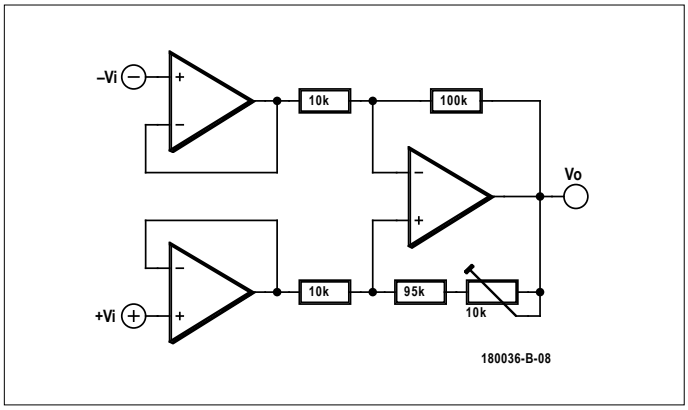


Figure 8. An instrumentation amplifier.

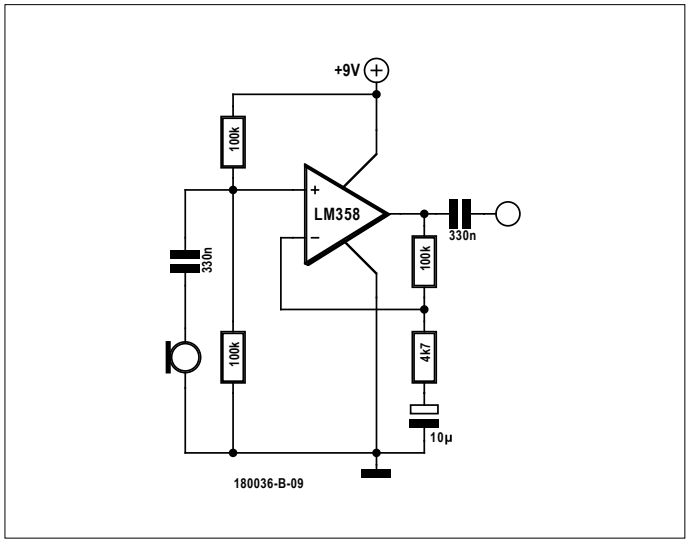


Figure 9. A microphone amplifier using an opamp.

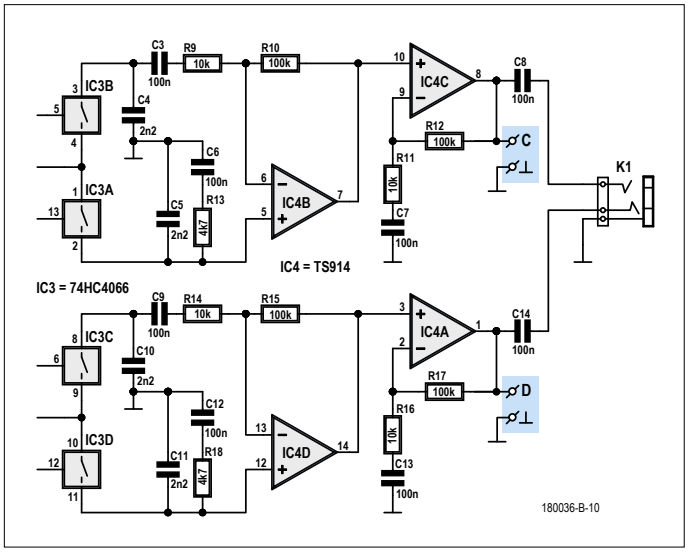


Figure 10. The TS914 as used in the Elektor SDR shield.

this is how it achieves its extremely high input impedance. The differential amplifier should be adjusted to optimise the stability and common-mode rejection performance of the circuit.

Wide-bandwidth applications

Although an opamp can basically be thought of as a DC amplifier, it is of course also suitable for use as an AC amplifier. One example is the microphone amplifier circuit of **Figure 9**. With a single supply voltage it is necessary to create an artificial supply between the rails, in this case at half the supply voltage. The circuit then behaves as if it had both a negative and a positive supply voltage relative to this intermediate rail. Amplifier circuits like this are capable of operation over a broad frequency range, although sometimes it can come as quite a

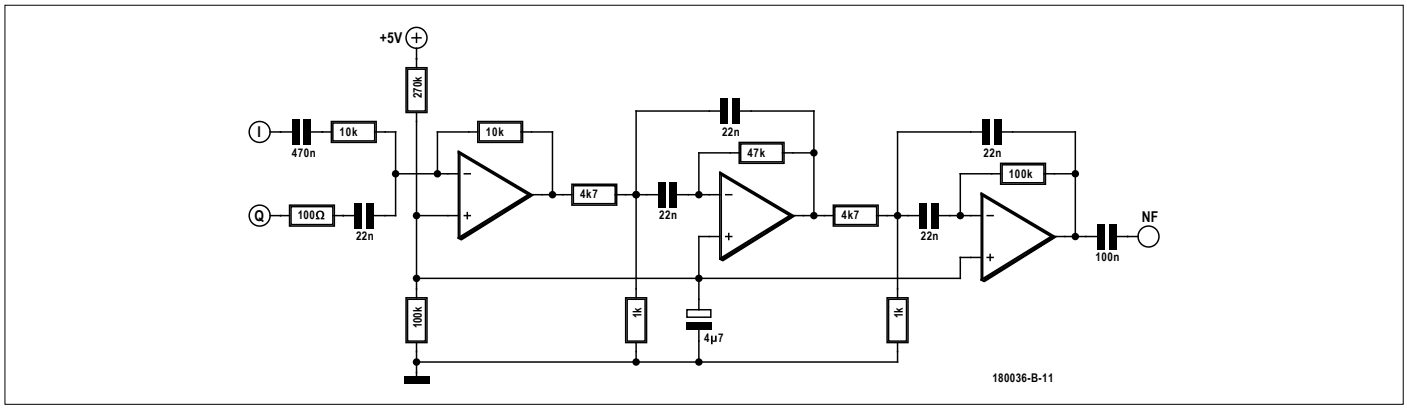


Figure 11. A bandpass filter with quadrature inputs.

surprise how quickly a device can bump into its frequency limit. Suppose, for example, that we want an amplifier with a gain of 100 that can operate at up to 20 kHz. The gain-bandwidth product is thus 100 × 20 kHz, or 2 MHz. However, many devices have a gain-bandwidth product of only 1 MHz or even lower. That means that at 20 kHz we already do not have enough gain available, and the negative feedback network will not operate as hoped. The result is distortion as internal stages in the device are driven into saturation. The oscilloscope will show a sinewave signal being distorted into a triangle wave, because inside the device the saturated stages are generating square waves, whose fast edges are integrated in the output stages to uniform ramps.

The Elektor SDR shield uses a type TS914 quad opamp which offers a gain-bandwidth product of 0.8 MHz (see **Figure 10**). In order to increase the overall bandwidth of the circuit it is divided into two stages each with a gain of 10. These stages will thus each have a bandwidth of 80 kHz, which is sufficient for this application, where the signals are taken to the stereo inputs of a sound card for further processing by the SDR software. Depending on the sound card, these inputs will have a maximum bandwidth of 24 kHz or 48 kHz; by processing the two channels separately we can achieve an effective total reception bandwidth of 48 kHz or 96 kHz.

For standalone operation of the receiver the signal processing software running on the PC must be replaced by a purely electronic solution. One approach is to use the bandpass filter with a centre frequency of 750 Hz shown in **Figure 11**. The circuit has separate in-phase (I) and quadrature (Q) inputs: first a relative phase shift of 90 ° at 750 Hz is applied to these signals, and then they are added together. The result is passed through a two-stage 750 Hz bandpass filter. The design uses type LM348 dual opamps.

The filter stages were simulated and tuned using LTspice. Be aware, however, that here again the limited bandwidth of the opamps makes a difference! It is easy to think that since all the signals involved are firmly in the audio frequency range there cannot possibly be a problem, and even the second filter stage, which has a gain of 100, demands a gain-bandwidth product of only 80 kHz for an input signal at 800 Hz. However, this overlooks the fact that many higher-frequency components will be present in the signal, and we do not want these to generate distortion. Things become even more critical if we want to construct filters with a high quality factor operating at frequencies of 10 kHz and above. ⏮

180036-B-02



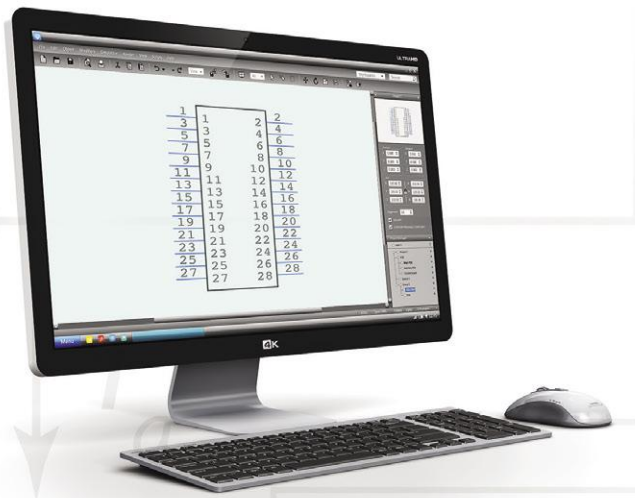
@ WWW.ELEKTOR.COM

Book: The Circuit Designer's Companion (4th Edition)
www.elektor.com/the-circuit-designers-companion-4th-edition

→ Book: Learning the Art of Electronics
www.elektor.com/learning-the-art-of-electronics-3rd-edition

Advertisement

MOUSER ELECTRONICS



SamacSys
ELECTRONIC COMPONENT LIBRARY SOLUTIONS

Download FREE

- Schematic symbols
- PCB footprints
- 3D models

Click & Drop into your ECAD tool

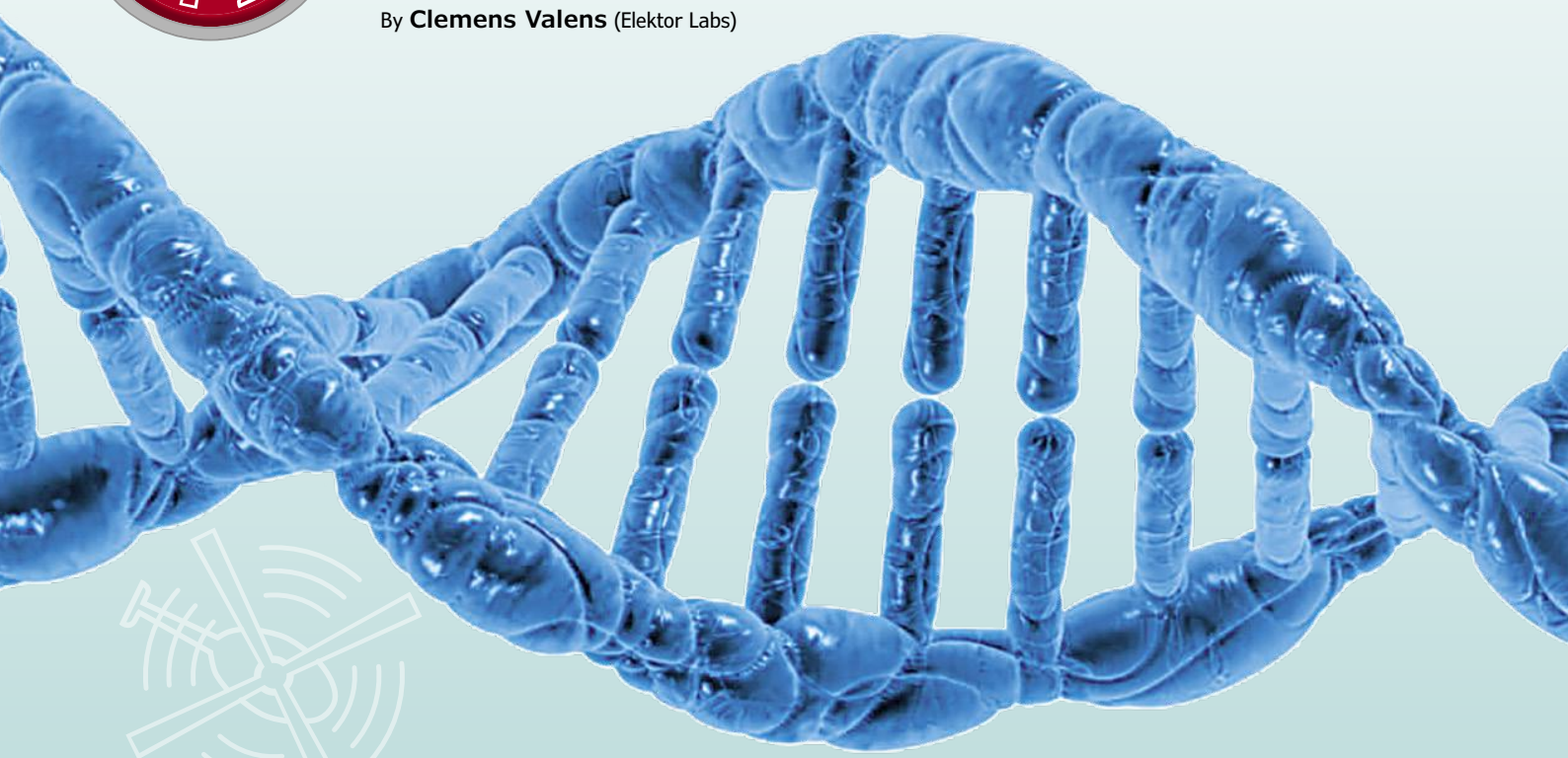
The Newest Products for Your Newest Designs®

mouser.com/ecad

HomeLab Helicopter

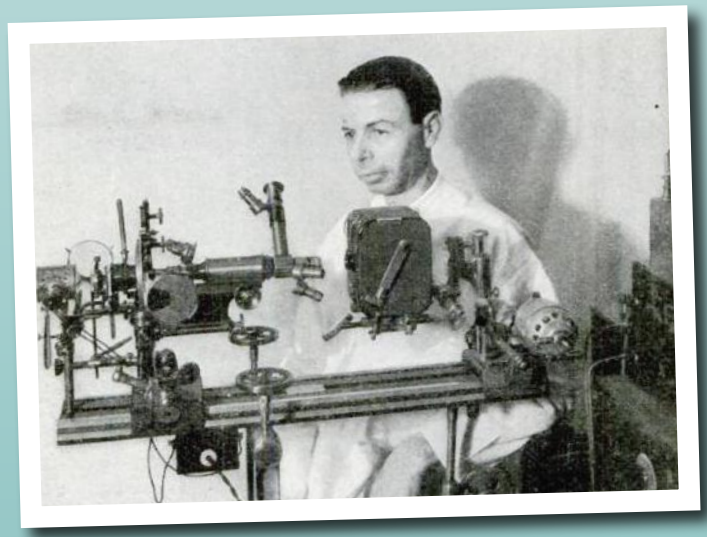
Wondrous Things Electronic Spotted From Above

By **Clemens Valens** (Elektor Labs)



Sympathetic Resonance — "it's a kind of magic!"

The Internet is full of communities, products and technologies just waiting to be discovered. Every once in a while, when surfing the net you stumble upon something that is new to you* and that for some reason attracts your attention. Following a few links you then quickly find yourself inside a community of people whose whole lives seem to revolve around this particular technology or product. To me, the Rife Machine was such a discovery. I had never heard of it, yet it seems to have quite a following. Or does it?



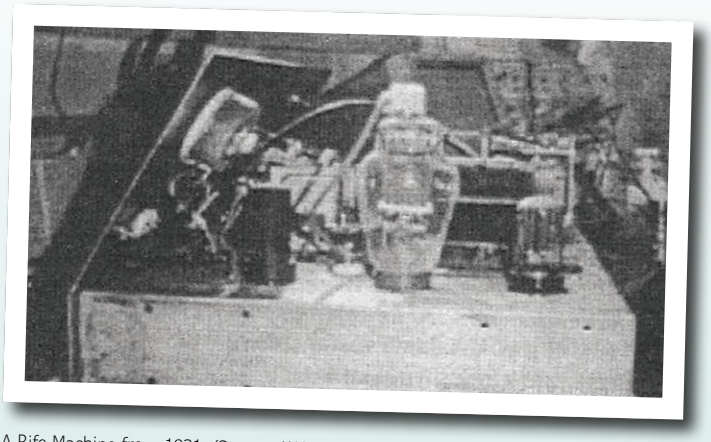
Royal Raymond Rife posing in 1931 with one of his microscopes he used for observing how germs are affected by sympathetic resonance. (Source: Wikipedia)

Before continuing, let's state that Rife Machines and the theory behind them are controversial to say the least. When investigating this technology, pseudoscience like Magnetic Field Therapy, Pyramid Power and others come to mind.

Invented sometime in the 1920s by Royal Raymond Rife (1888-1971), the Rife Machine is an electronic device intended to destroy malicious microorganisms like bacteria, viruses and cancer cells. Basically, the idea is that every living organism has a resonance frequency. When the organism is excited by this frequency, it will either be killed or disabled. Royal Rife dubbed this frequency the Mortal Oscillatory Rate or MOR, other people also call it the Royal Rife Frequency. How disastrous resonance can be is well known from the famous video footage of the Tacoma Narrows Bridge collapse back in 1940. Rife's work was kind of reinvented and repeated about ten years ago by music professor Anthony Holland and a very clear presentation of his work can be found on YouTube.

*** There used to be a website called StumbleUpon, intended for exactly these kinds of things.**

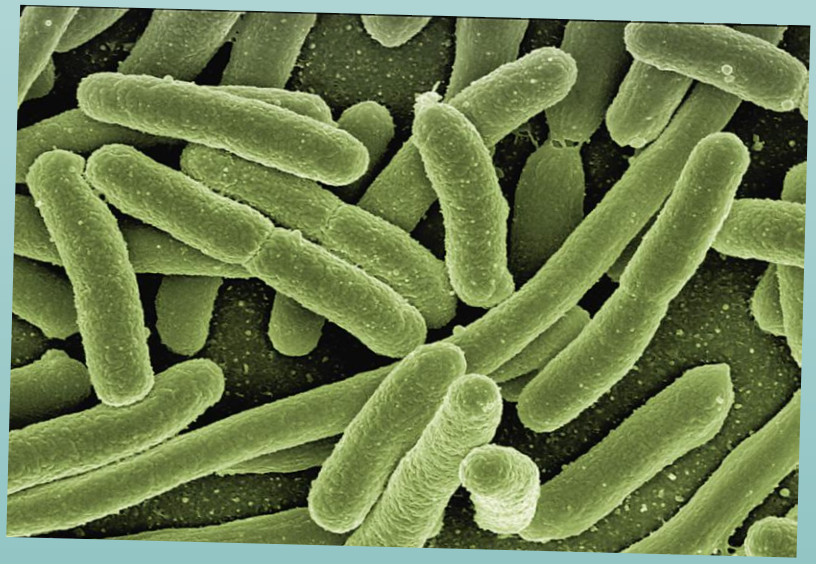
A Rife Machine uses a common function generator to create signals at special "mortal", empirically determined frequencies. These signals are then introduced by some means into the organism where the pathogens presumedly live that have to be destroyed through sympathetic resonance. The mechanism is the same as when a skilled singer shatters a wine glass. Three methods can be used to transmit the signal to the germs: direct contact with hand-held electrodes for instance; wireless with the help of plasma antennas; and remotely, by using nonlocal space, quantum physics and the antenna properties of DNA. In a plasma antenna highly ionised plasma, being a good conductor, replaces traditional transmission line elements or antenna surfaces. Due to the fact that a plasma has no physical length, a plasma antenna can be tuned easily. As early as in 1919 a patent was awarded to one J. Hettinger concerning this technology. More recently, in 1999 to be precise, James E. Bare obtained US patent #5,908,441 for his Resonant Frequency Therapy Device based on a plasma antenna.



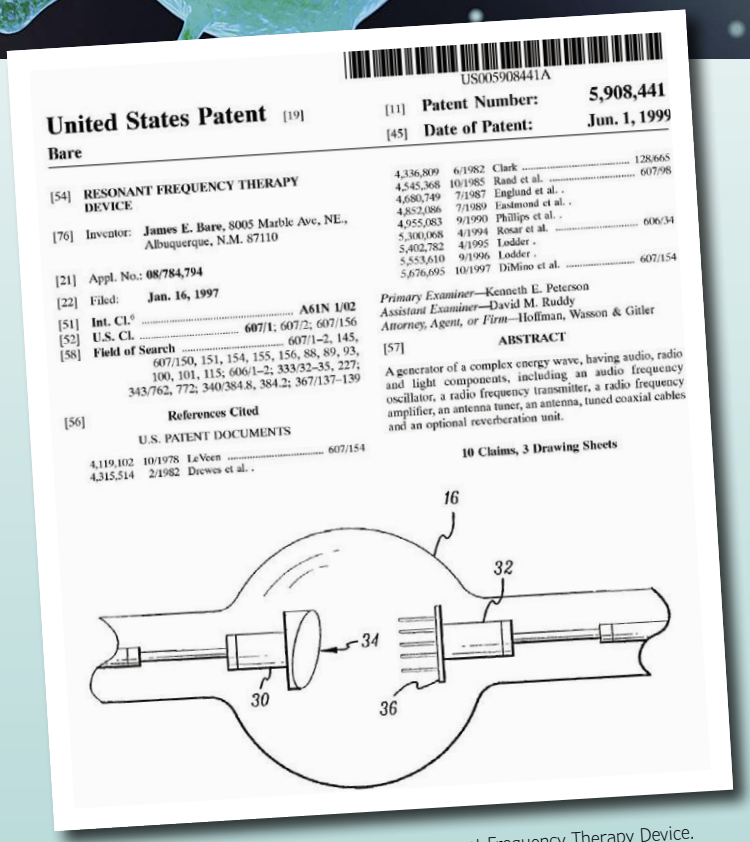
A Rife Machine from 1931. (Source: Wikipedia)



The main advantage of the remote-quantum-DNA method over the others is that remote machines work over distances of thousands of miles and more. To tune the generator to the patient, a nail clipping must be placed in a small container. This is then connected to the generator that sends audio frequency energy to the DNA contained in the nail clipping. On the website supporting the Spooky2 — a low-cost Rife Machine — it says: "*Because DNA has two strands helically coiled around one another, it acts as an antenna that's capable of transmitting and receiving information encoded on energy signals via nonlocal space.*" Quantum entanglement is the mechanism that supposedly relays the energy instantaneously via nonlocal space to the parent DNA, which happens to reside in the relevant patient only (it appears to be a highly accurate technique). The Spooky2 website goes on to explain that quantum entanglement "*means that if any part of a single system is removed from that system to a different location, any action performed on the part will also be instantaneously performed on the parent system,*







US005908441A

United States Patent [19]

Bare

[11] **Patent Number: 5,908,441**

[45] **Date of Patent: Jun. 1, 1999**

[54] **RESONANT FREQUENCY THERAPY DEVICE**

[76] Inventor: **James E. Bare**, 8005 Marble Ave, NE., Albuquerque, N.M. 87110

[21] Appl. No.: **08/784,794**

[22] Filed: **Jan. 16, 1997**

[51] Int. Cl.[6] **A61N 1/02**

[52] U.S. Cl. **607/1**; 607/2; 607/156

[58] Field of Search 607/1–2, 145, 607/150, 151, 154, 155, 156, 88, 89, 93, 100, 101, 115; 606/1–2; 333/32–35, 227; 343/762, 772; 340/384.8, 384.2; 367/137–139

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,119,102	10/1978	LeVeen	607/154
4,315,514	2/1982	Drewes et al. .	
4,336,809	6/1982	Clark	128/665
4,545,368	10/1985	Rand et al.	607/98
4,680,749	7/1987	Englund et al. .	
4,852,086	7/1989	Eastmond et al. .	
4,955,083	9/1990	Phillips et al. .	
5,300,068	4/1994	Rosar et al.	606/34
5,402,782	4/1995	Lodder .	
5,553,610	9/1996	Lodder .	
5,676,695	10/1997	DiMino et al.	607/154

Primary Examiner—Kenneth E. Peterson
Assistant Examiner—David M. Ruddy
Attorney, Agent, or Firm—Hoffman, Wasson & Gitler

[57] **ABSTRACT**

A generator of a complex energy wave, having audio, radio and light components, including an audio frequency oscillator, a radio frequency transmitter, a radio frequency amplifier, an antenna tuner, an antenna, tuned coaxial cables and an optional reverberation unit.

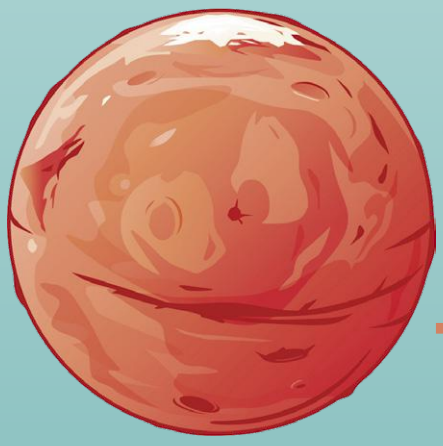
10 Claims, 3 Drawing Sheets

The first page of a 1999 patent concerning a Resonant Frequency Therapy Device.

and vice versa." Though this may sound like an oversimplification, similar techniques have been used for centuries in what is called sympathetic magic. Practitioners of this kind of magic believe that when a person comes in contact with an object, some of his/her energy is transferred into the object. By placing such an object — or a lock of hair or nail clippings — inside a (voodoo) doll and then sticking needles in the doll, the person energetically connected to the object will be directly affected. Clearly, quantum entanglement is old hat. Actually, there are two other ways of transporting the mortal oscillations into the body: cold lasers or Low Level Laser Therapy (LLLT) based on non-thermal light photons — biophotons as investigated by Fritz-Albert Popp — and Pulsed Electromagnetic Field Therapy or PEMFT. Both LLLT and PEMFT are quite common in alternative medicine and there is some evidence that they may have a slightly positive effect. The American Food and Drug Administration (FDA) even cleared some PEMFT devices, and they are commercially available as 'wellness' devices.
Because there is no scientific or medical foundation to back the usefulness of Rife Machines at any level, selling them is illegal in certain countries. To electronics engineers that's not really a problem because most will have a function generator in their lab, or access to one. Basically all you have to do is connect the generator output to a living human being in a safe way. That's why the remote machine is so appealing: it avoids physical contact between the generator and the patient. Simply connect the output of the generator to a nail clipping provided by the patient — impedance matching may be an issue here — and spin the dial. Power is irrelevant as the helical DNA antenna is very efficient; just don't burn the clipping with your AC power. Ever since I set up such a system in my lab, continuously sweeping over a wide frequency range to cover many germs, I have not contracted Spanish flu once.

If this column generates enough sympathetic response, I might be inclined to dedicate some future instalment to the Keely motor, a motor based on "vibratory sympathy".

Note: All responsibility for any damage or malfunctioning caused by the use of sympathetic resonance and/or magic on living beings as well as on inanimate objects, is strictly declined.



Listen to the sound of Mars: youtu.be/o3cxuIsEFRM

Tips & tricks



How to test if a battery is empty or full? Easy, just let it drop from a height of 20 cm on a hard flat surface like a table top. If the battery jumps up it is empty, if it just falls over it is full. Sounds unbelievable, right? Yet this trick does work for non-rechargeable alkaline AA- & AAA-type batteries. Really, we have tried it in our lab. The reason is the sticky stuff inside the battery that solidifies when it is discharged. It loses its shock-absorbing qualities, and the battery will bounce when it hits a hard surface.

Electric vehicles too quiet?



Although many seem convinced that electric vehicles are a cure-all, in reality that's not going to be the case. For instance, because they lack a loud combustion motor EVs fail to make enough noise to alert pedestrians and cyclists, which can lead to dangerous situations. The weight of the problem is proven by the fact that governments all over the world have started investigating legislation for setting a *minimum* sound level for electric vehicles. To help the visually impaired, pedestrians and cyclists hear and locate EVs, more noise production is required!. Luckily plenty of technology is available to solve the problem, from Indian-style continuous honking to dragging a string of just-married cans behind the vehicle... making noise never has been a problem. However, here at at Elektor we tend to prefer electronic solutions, which is why we promote application note AN-1552 from Analog Devices about the Electric Vehicle Warning Sound System (EVWSS). An EVWSS "*produces a series of sounds designed to alert pedestrians to the presence of an EV. The driver can initiate warning sounds (similar to the sound from a car horn, but less urgent); however, the sounds must automatically be enabled at low speeds. These sounds vary from artificial tones to realistic sounds that mimic engine noise and tires moving over gravel*". (Did they have hovering EVs without wheels in mind when they wrote this?) Reading this one cannot help remembering the 1975 movie classic '*Monty Python and the Holy Grail*' in which King Arthur's footman (Patsy) is banging together a pair of coconut halves to suggest the King is riding his royal steed.
AD has prepared two EVWSS solutions for applications with an in-cabin engine sound as well as an external engine sound based on the ADSP-BF706 BlackFin DSP. Entry-level systems can use the ADAU1450 SigmaDSP.

180567-B-01

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

Rotary Encoder Interface without µC

Simple and compact

By **Mike Mittelhäuser** (Germany)



An incremental rotary encoder is a very interesting electromechanical component with many applications wherever an accurate manual control is required, both in analog as well as digital circuits. The cost is typically no more than some $/€/£ 2 and they are available from most electronics components retailers. But unfortunately the pulses from the two outputs cannot be read and used immediately, because these are phase-shifted signals that appear at each step of the rotation. This circuit provides an easy solution.

A microcontroller with appropriate software is usually employed to interpret the pulses generated by a rotary encoder and to determine the direction of rotation. For many applications that's either not very convenient or it requires just too much effort to use a rotary encoder as a digital potentiometer. The Flight Simulator project author was working on required a versatile application of 20 to 50 rotary encoders for both analog and digital applications. He therefore had to find a much simpler solution for this. Preferably a cheap, small circuit board, no bigger than the encoder itself, with separate Schmitt-Trigger outputs for clockwise and anti-clockwise rotation and with pulses that do not merge together when the encoder is turned rapidly.

Principle of operation

To meet these requirements the author designed the circuit described here, which is suitable for encoders with and without a built-in pushbutton switch. The design is suitable for most types on offer, including those made by Alps.

The circuit, see **Figure 1**, consists of two D-type, positive-edge-triggered, CMOS flip-flops with preset and clear inputs, in the form of a 74HC74, and six inverters with Schmitt-Trigger inputs: a 74HC14. The encoder is wired active Low. All the outputs from the encoder/inputs to the circuit (including that from the pushbutton) are connected via two 10-kΩ resistors in series to the positive power supply rail. C1 through C3 provide decoupling of the inputs. The circuit ensures a contact-bounce free, active-Low pulse with a falling edge time of 7 ms and a rising edge time of 14 ms.

The Schmitt-trigger inverters convert the pulses from the encoder into active-High signals with steep edges, in order to trigger the D-flip-flop with a rising edge. The time between successive pulses is limited to a minimum of 10.5 ms to prevent pulses overlapping.

The output of inverter IC1.A is connected to the D-input and the reset input of the first flip-flop (IC2.A). The preset input is not used and is connected to V_{CC}. The output of the first inverter is also connected to the clock input of the second flipflop.

The output of the second inverter (IC1.F) is connected in the same way as the first one, with the difference

that IC2.A and IC2.B are swapped around. With this the circuit is in effect complete.

In the rest state nothing happens at the outputs of the flip-flops. Both are Low (which makes the outputs on K2 High). However, when the encoder rotates a step, it gives two phase-shifted pulses, which depend on the direction of rotation. We first look at what happens when the rotary encoder turns anti-clockwise.



Rotating the encoder

When the encoder is turned anti-clockwise, the circuit sees the first pulse (‘$\overline{\text{FFA}}$’) appear on pin 1 of K1 (‘FFA’ in **Figure 2**), after a quarter period this is followed by a pulse on pin 3 (‘$\overline{\text{FFB}}$’). This first (active-Low) pulse ($\overline{\text{FFA}}$) is inverted by IC1.A and then activates IC2.A (releases the reset). Because there is no clock signal (no pulse yet on pin 3 of K1), the output of IC2.A does not go High and the corresponding output (active-Low, pin 2 of K2) remains High. However, IC2.B does receive a clock pulse. But this results in no change at the output of IC2.B, because its input was Low.

When subsequently a pulse appears at pin 3 of K1 (active-Low) as well (time ‘B clock A’ in **Figure 2**), the output of IC1.F will go High, with the consequence that IC2.A will ‘clock’ the signal on its D-input to its output. Pin 2 of K2 will now go Low to indicate the encoder is turned anti-clockwise. Although the D-input of IC2.B is now High, this signal is not ‘clocked’ to the output because the signal on the clock input of IC2.B was High (and did not *change* from Low to High), so pin 4 of K2 remains High.

Once the pulse on pin 1 has gone (time ‘reset A’ in **Figure 2**), IC2.A is reset and pin 2 of K2 goes High again. Nothing changes at the output of IC2.B, since the clock input only reacts to a *rising* edge.

The final step in the cycle is the disappearance of the pulse on pin 3 of K1 (‘$\overline{\text{FFA}}$’). The clock input of IC2.A will go Low, which therefore results in no change at its output (and the reset was active as well). IC2.B is reset, but its output was already Low, so nothing changed there either.

When turning the encoder clockwise a nearly identical sequence of events occurs, with the only difference that the output of IC2.B changes and the output of IC2.A remains Low.

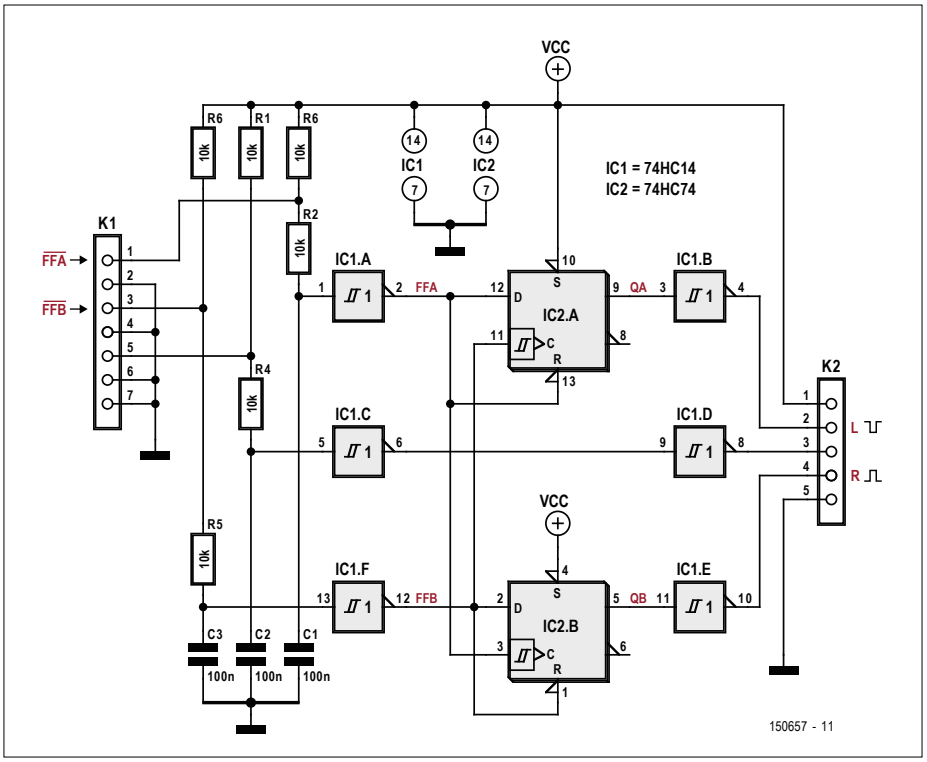


Figure 1. The circuit for decoding the output of a rotary encoder is built around two ICs from the 74HC family.

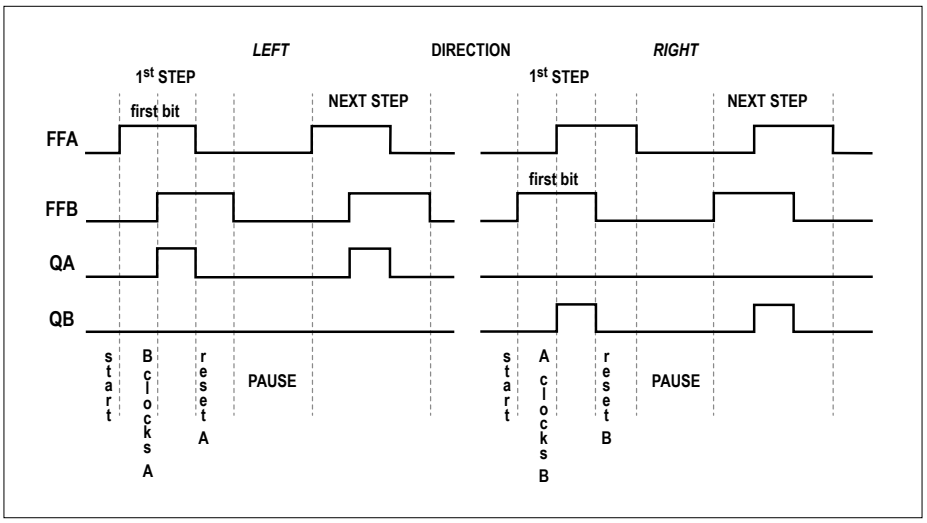


Figure 2. The timing diagram shows how the circuit converts the two phase-shifted pulses into two discrete pulse signals.

Finally

We note that the durations of the pulses at the output are half as long as those from the encoder at the input. The pushbutton output remains Low until the button is released. It is also buffered using two Schmitt-Trigger inverters.

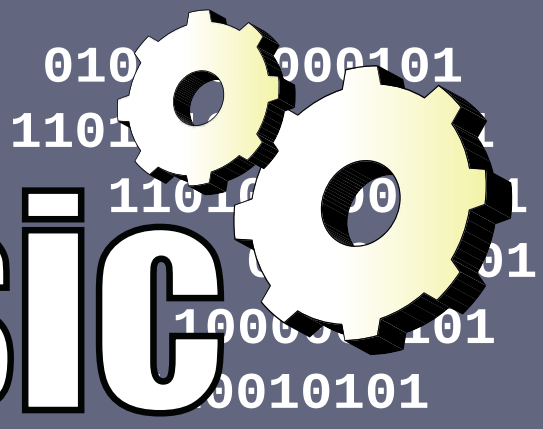
The author offers ready assembled modules (without the rotary encoder), which you can order directly from him [1].

(150657)

Author Contact

[1] mike.mes-engineering@web.de

PureBasic



One BASIC Compiler, many platforms

By **Waldemar Pieczerk** (Germany)

A little-known program for cross-platform development is PureBasic, a compiler for the decades-old programming language BASIC. The programs compiled by PureBasic are small, fast and can be started without any installation via a USB stick. This compiler has much to offer and is certainly worth a closer look.

Key Features

- Builds applications for both 32-bit and 64-bit systems
- Form designer to create user interfaces (GUIs)
- Creation of custom DLLs
- Use of indicators/pointers
- Access to operating system APIs
- Inline assembler
- Integrated Compiler and Debugger
- Direct generation of executable EXE files.

When it comes to developing software, cross-platform development tools are trending at the moment. The big idea behind this approach is that you no longer develop a native application that runs specifically on one operating system. Instead you write one version of the software and allow the development environment to configure it according to the desired target operating system. The idea has been around for a long time but is becoming more and more important. We now expect to be able to run the same successful applications in our PC operating systems as well as on the operating systems running in our mobile devices. This approach is called 'Rapid Application Development' (RAD). Both large and small suppliers of programming language tools offer many different solutions but the range of target platforms supported by each development environment will differ.

In practice two variants exist and each has their own advantages and disadvantages:

1. The code is written just once and then compiled separately for each target platform. The executable file can be kept very small because the program uses, as far as possible, functions in the operating system (e.g. to draw a button). Under certain circumstances the program can run immediately without the need for installation.

2. The code is compiled just once and is immediately executable on all target platforms. The elements of the graphical user interface (for example, buttons) look different in different operating systems with different sizes and borders, the user interface for each target system would therefore need to be checked and adjustments made as necessary. To avoid this complication, the development environment comes with its own OS-specific elements. These must be installed on the target system where the program will run. As a result, the space requirement of the compiled program on the hard disk is significantly larger overall.

One of the lesser-known programs for cross-platform development is PureBasic [1], a compiler for the BASIC programming language. Despite its low level of awareness amongst programmers, the language can trace its roots back to the dawning of the personal computer age. It is also allegedly popular in the Russian hacker scene. PureBasic belongs to the first category outlined above: the compiled program is compact and fast and can run without the need for installation from a USB stick. It doesn't, however support any of the operating systems running on mobile devices.
The majority of modern programming languages are object-orientated but PureBasic is an exception, it is a procedure-orientated language but it can still deal with

objects. Procedural programming uses a simpler syntax and this simplicity makes it an attractive option for hobbyist programmers because for newbies the training period is generally shorter.
Both professional engineers and electronics enthusiasts are always interested in how a language handles external peripherals such as printers, serial communication and Ethernet as well as special functions such as sending e-mails. There are numerous code examples on the forums that support this language showing how measurement data can be displayed as charts or diagrams. Some of the examples are so large they cannot be run on the demo version of PureBasic because of its code size restriction. The many examples included are a useful resource to facilitate better understanding of the language.

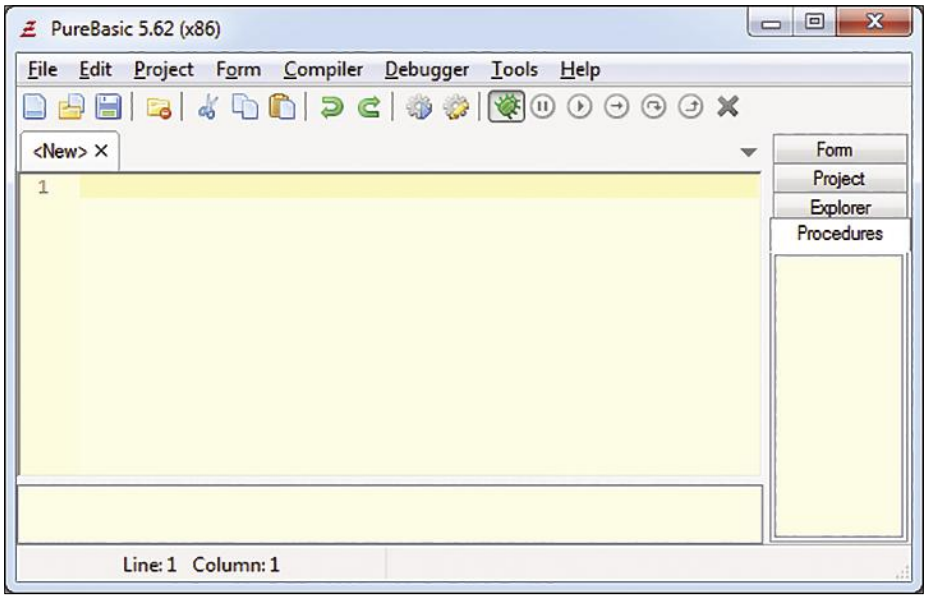


Figure 1. The PureBasic user interface.

A peek at the Windows version

Running in a Windows environment the minimum system requirement for PureBasic is Windows XP. To test the claim I actually installed it on a machine running XP and it worked! PureBasic not only creates EXE files that run without the need to install additional runtime libraries, even the development environment itself doesn't need any files other than those in the program directory. Once PureBasic is installed, you can copy the PureBasic folder from the program directory to a USB stick and have the basic compiler always with you! The only disadvantage is that the compilation process runs a bit slower from the USB stick.
In principle Linux is also supported, but there is one small damper on the proceedings: PureBasic cannot compile programs for the popular Raspberry Pi single-board computer.
This is because it is only compatible with the Intel x86-based processor instruction set. PureBasic was originally written to run in the AmigaOS. This is still evident today; in some places the syntax differs significantly from the other variants of Microsoft BASIC (QBasic, VB, VBA). It seems also that the influence of some elements of the C programming language can be detected in some places.

Here are just a few differences from other BASIC dialects:

- comments are prefixed in the program code with a semicolon (;);
- elements of the user interface windows (text entry boxes, buttons, and so on) are called gadgets;
- a separate function is used to indicate exponentiation: `pow(a,n)` represents a^n.
- The keyword DEFINE declares variables. `Define.s Variable1, Variable2, Variable3` indicates that three string variables are defined;
- direct declaration of a variable is also possible: `Variable4.d`. The variable name ends with a full stop followed by a letter indicating the variable type. (`d.` indicates double);
- there is no distinction between functions and procedures; there are only procedures;
- passing variables to and from procedures is performed like so:

```
Procedure.d
MyFunction(Number1.d, Number2.d)
  result.d = .............
  ProcedureReturn result
EndProcedure
```

First `MyFunction` must be declared then the return type is set to double. Finally, two double numbers are passed to the function.

- the If–Then condition: The keyword *Then* is not used. A conditional branch looks like this:

```
If a=1
....
EndIf
```

- Message-Box: The information window called a message box in the Microsoft world is called 'MessageRequester' in PureBasic.
- Mathematical operators can also be used in a short hand notation: `A+1` is the same as `A=A+1`, `B*2` is the abbreviated notation for `B=B*2` and `C<<1` means `C=C<<1`, a bitwise shift to the left.
- For-Next loops only work with integer values.
- to break out of a loop, the keyword `Break` is used:

```
For i=1 to 10
  If i=5
    Break
  EndIf
Next
```

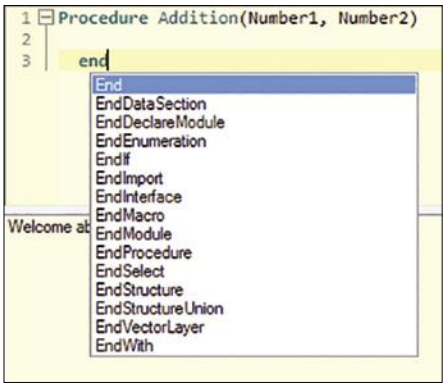


Figure 2. Auto-complete and syntax highlighting.

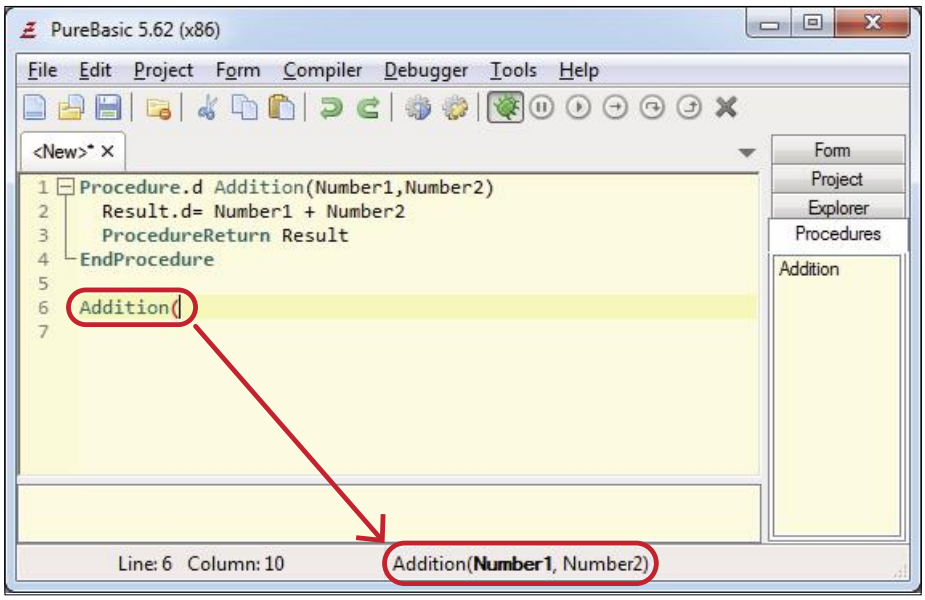


Figure 3. Auto-complete: Parameter display.

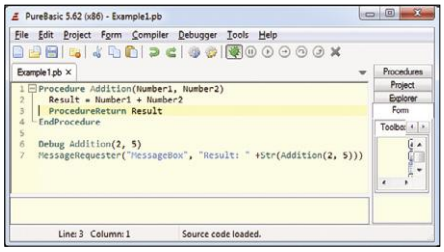


Figure 4. The finished program.

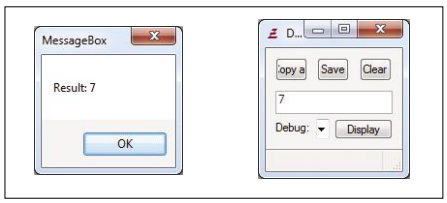


Figure 5. Program results displayed in the MessageBox and the debug window.

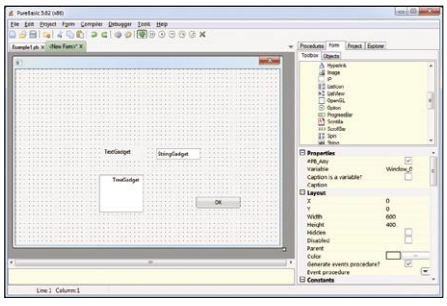


Figure 6. Graphical GUI development.

- when `Break` is used together with a number this allows you to jump out of a nested loop, the number indicates the level of nesting (2 in this example).

```
For i=1 to 10
   For j=8 to 20
      If i=5
         Break 2
      EndIf
   Next
Next
```

As you can see, there is a lot to take in. It may be a bit tedious but in the long run it's of great benefit to make use of online help resources and read the language reference manual.

My first program

At start the program displays the page shown in **Figure 1**. The large free area is reserved for the source code, below which system messages are displayed. In the area on the right you can select an overview of all the procedures in the current file or project, the available control elements of the design from the operating window and a small file explorer. In the toolbar, the depressed button with the green beetle symbol indicates the debugger is selected.
This can only be disabled in the full version of PureBasic, doing this makes the compiled programs even more compact, because they no longer contain additional information used by the debugger.
The editor also takes care of AutoComplete and syntax highlighting. Procedures can be 'collapsed' as in **Figure 2** so that only the declaration is visible. The first program just adds two numbers and outputs the result.

```
Procedure Addition(Number1, Number2)
  Result.d= Number1 + Number2
  ProcedureReturn
EndProcedure
```

Granted, this example is not very original or even useful, but it's got to be an improvement on 'hello world' and it serves as a good taster for PureBasic syntax. When a command or procedure is entered to which parameters have to be passed, the expected parameters are displayed after the first open bracket symbol is typed. After our 'Addition procedure' has been defined, we also get this support (**Figure 3**).

The program window in **Figure 4** shows the complete program, an 'Addition' procedure that adds two numbers and returns the sum. Since no variable types have been specified, the type integer is used by default for all variables. The two lines below show two possible ways to display the result. Line 7 opens the debug window (only works when the debugger is switched on) and line 8 opens a MessageBox.
The program contains an error because both of the values are integers; their sum could result in a value larger than the range of values allowed as integers. Quad would seem to be a suitable data type to use here but thereby waits a trap to ensnare the casual programmer: When choosing the data type for the result, it should be noted that the programs can be compiled on 32-bit and 64-bit machines. On a 64-bit machines integer variables are 8-bytes long and therefore the same length as the quad type. A Float or Double type should therefore be used for the result of the addition.
We did not correct this error, but just started a test run. This can be initiated by selecting 'Compiler ➞ Compile/Run' in the menu or by clicking the button with the blue arrow symbol.
PureBasic doesn't complain about the error, but converts the BASIC code into assembler. The built-in assembler compiler generates an EXE file and this all takes place in the back-

ground. Shortly afterwards, the MessageBox and Debug window show the result of the addition (**Figure 5**). Now we can save our program with 'File ➞ F Save'. Note that PureBasic does not automatically add the file extension! Valid file extensions are:

- *.pb – for the source code
- *.pbi – for uncompiled function libraries (PureBasic include)
- *.pbf – for graphical user interfaces (PureBasic-Forms)
- *.pbp – for project files (PureBasic-Project)

Projects contain collections of several program files. If you cannot find your source code, then you probably saved it without adding a file extension.

Graphical User Interfaces

To create a graphical user interface (GUI), simply select in the menu 'Form -> New Form'. An empty window appears, to the right of it a list with the gadgets and their properties. The gadgets can now be freely placed in the window (**Figure 6**). Some properties of the gadgets will only displayed correctly at runtime, such as font changes. The runtime representation can be viewed by clicking on the symbol with the blue arrow.

This visual form designer creates BASIC source code in the background, which can be viewed by clicking on the 'Form ➞ Switch Code/Design View' button. The basic structure of Gadgets is linked with event procedures. This form-file should be saved as *.pbf. It is better to save the source code in another file (*.pb) so that the Form Designer cannot overwrite it. There is insufficient space available to show an example unfortunately.

A COMTest example

A program example called 'COMTest.zip' is available from the Elektor Web page [2]. It shows how user interfaces are programmed under PureBasic and how to access the serial port.

A lot of test equipment fitted with a USB port use a virtual serial interface (COM port) for data communication. If you still have devices with RS232-interfaces you have the problem for every new piece of kit to determine if the cable uses the correct pin assignment. It's therefore useful to first test if data communication can be established before you start programming. For this purpose the 'COMTest' program was written not as a pure demo program but also a useful tool to test the comms.

Compiling

- Extract the files from the ZIP archive into any directory.
- Start PureBasic and load the 'COMTest.pb' file.
- Select 'Compiler ➞ Create Executable' in the menu.
- After the destination directory has been entered, the program is then compiled. It will then be ready to run immediately.

The generated EXE file is only 60.5 KB. Copy it to a USB stick and run the program on the stick on different computers. You will discover that this little .EXE file is a complete working Windows application without any of those huge runtime libraries!

Function

The program searches for available COM ports at startup and then displays them in the combo box as 'Com port' (**Figure 7**). Now it is necessary to enter the communications parameters and using 'Data to send' enter a command to be sent to the DVM which will make it send a response. The 'Send' button can now be clicked to send it and then use the 'Receive' button to display the received message from the DVM in the 'Received data' window.

The ':READ?' command used here for

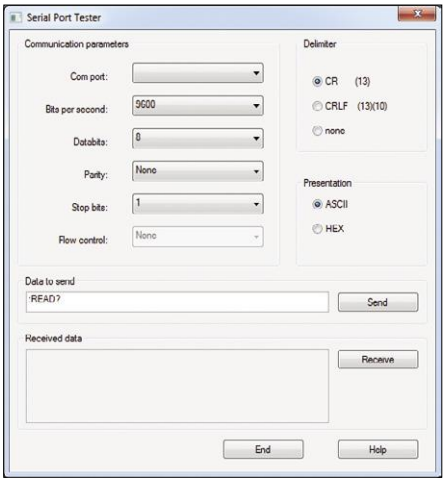


Figure 7. The COMTest GUI.

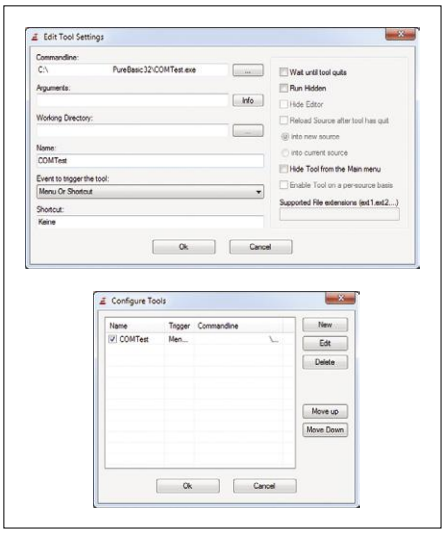


Figure 8. PureBasic can be expanded with external tools.

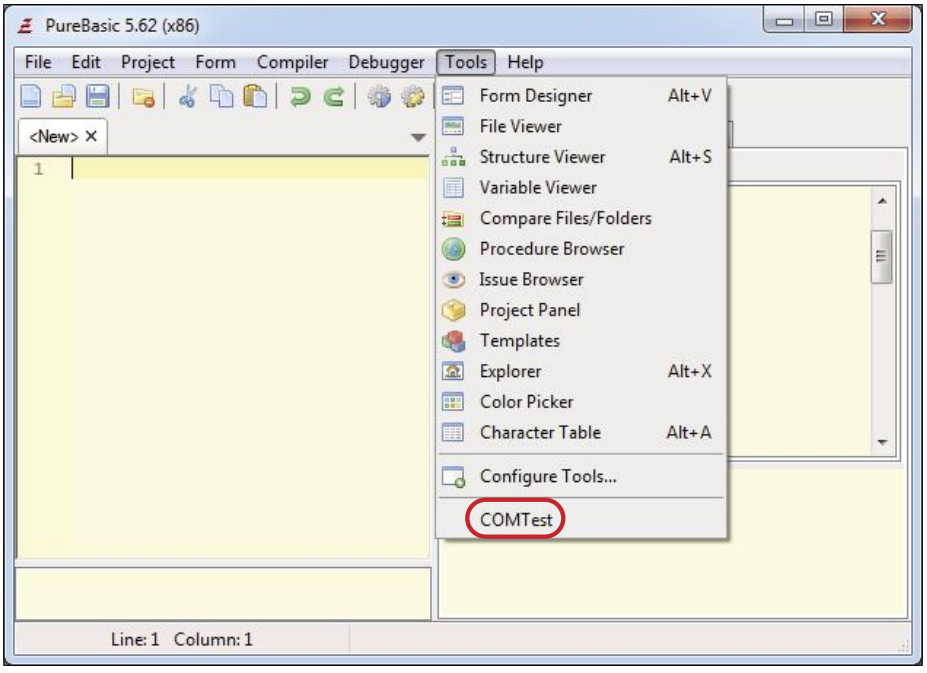
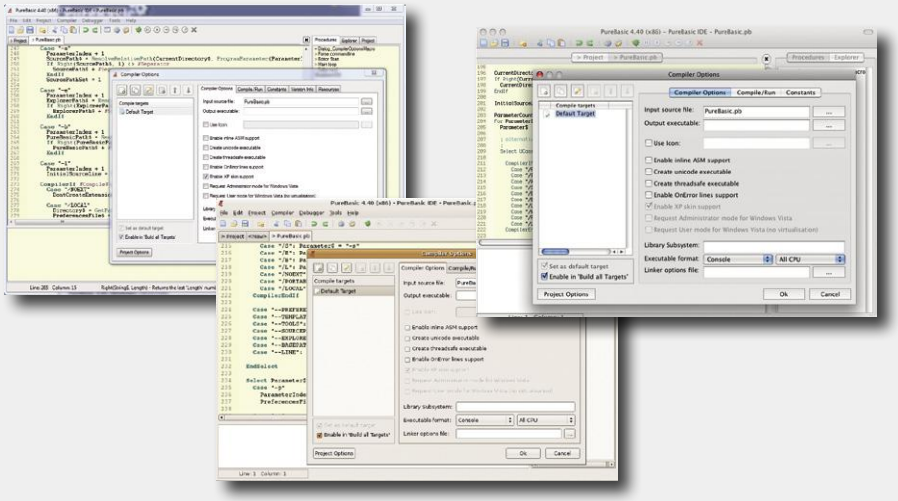


Figure 9. COMTest now appears as an entry in the Tools menu.

Available versions and costs

A demo version can be downloaded from the PureBasic website. This version is fully functional, but the code size is limited to 800 lines. In addition, no DLLs can be created and used. The full version costs €79 and includes versions suitable for Windows, Linux and MacOS X operating systems. With the full license you also get free lifetime updates. The version for AmigaOS is now open source and will no longer be supported.



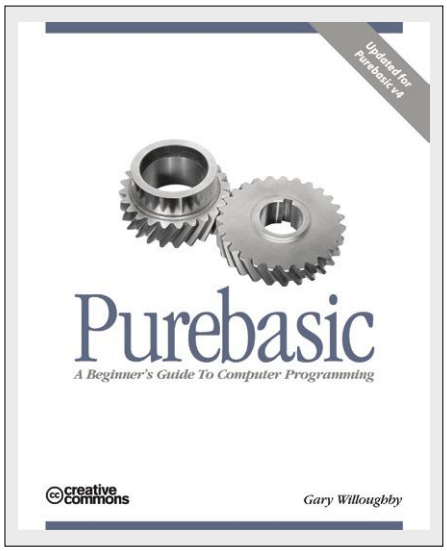


Figure 10. Weighty and insightful: An introduction by Gary Willoughby.

example tells the 34401A-benchtop DVM to send its latest measurement.

Add-ons

PureBasic can also be expanded by adding its own programs (such as the EXE file created above) which will be integrated into the PureBasic user interface (**Figure 8**). This program can then be selected from the Tools menu. It can also be selected in various ways such as automatically when the editor is started.

Adding external tools:

- Select 'Tools ➞ Configure Tools' in the menu.
- Then press 'New' so that a new window appears.
- Select the tool under 'Commandline'.
- The tool must be given a name ('COMTest').
- After closing both windows using 'OK' the Tools menu has now been expanded with this new tool (**Figure 9**).

Conclusion

Programming GUIs may be a bit cumbersome but PureBasic is a very comprehensive BASIC compiler with simple syntax capable of producing very fast and compact code. The compiled programs do not need any additional runtime libraries - a BASIC without all the additional baggage! The simple accessibility to external peripherals makes it of interest to electronics developers and when you consider that updates are free, the price for the full version works out to be really cheap. If you want to delve deeper into this programming language you can download the 1679 page PureBasic Reference Guide [3] or the 258-page book 'PureBasic — An Introduction to Computer Programming' by Gary Willoughby [4] (**Figure 10**). Enthusiastic support available in the forums can help iron out any small and more complex problems you may encounter (providing you are not identified as a hacker ironing out bugs in your latest malware virus). ⏮

(150839)

Web Links

[1] PureBasic Homepage: www.purebasic.com

[2] COMTest on the Elektor project page: www.elektormagazine.com/150839

[3] Language Reference handbook: www.purebasic.com/documentation/PureBasic.pdf

[4] Introduction: www.purearea.net/pb/download/PureBasicBuch.pdf

[6] PureBasic support site: www.purearea.net

Err-lectronics

Corrections, Feedback and Updates to published articles

DIY Temperature Controlled Solder Station

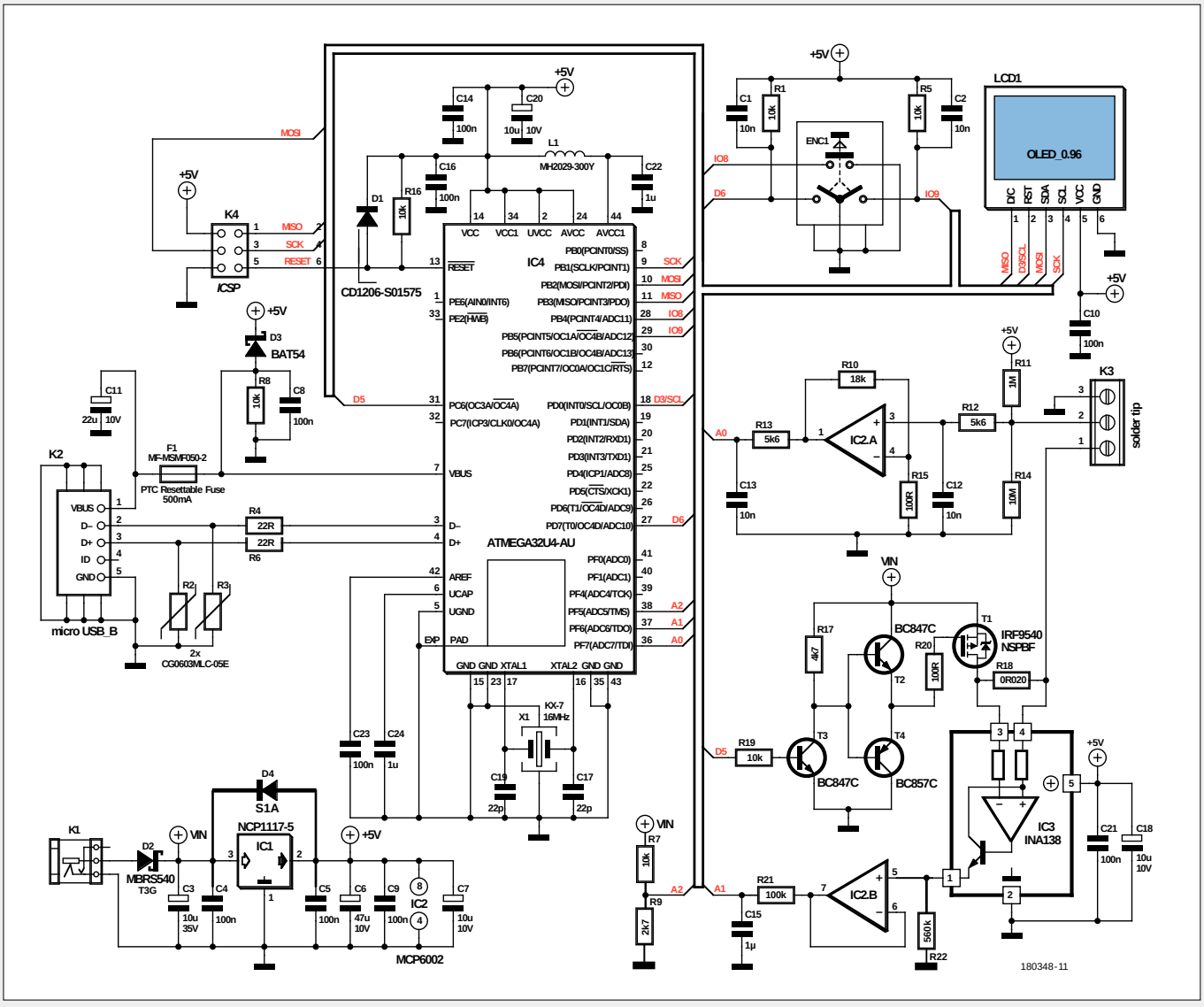
ElektorLabs magazine 1/2019, p. 14 (180348)

CORRECTION. The circuit diagram contains the following errors:

- T4 must be a PNP type with the emitter connected to the emitter of T2.
- R22 (560 kΩ) at the output of IC3, to 0 V, is missing from the schematic.
- D4, a freewheeling diode across IC1, is missing from the schematic.
- R7 and R9 form a voltage divider for measuring V_{IN} via analogue input A2 on the controller; the two resistors are missing from the circuit diagram. The software does not check the voltage on port A2 at this time, but you can add this function as a user. For example, you could power the soldering station from a (car) battery and avoid a deep discharge.

The printed circuit board is correct.

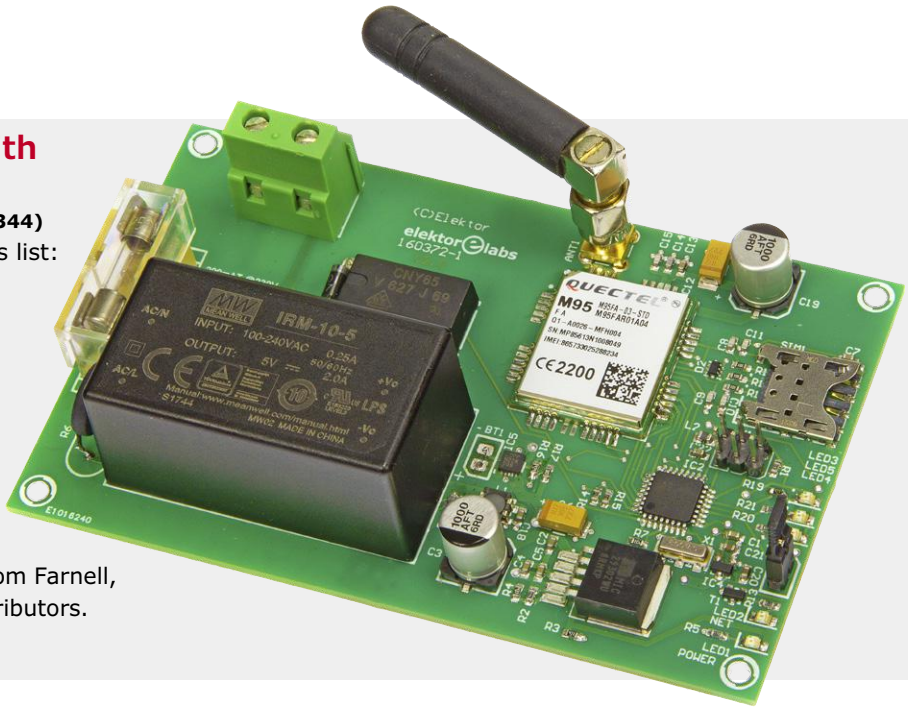
A new circuit diagram can be downloaded free of charge for all readers under the web link:
www.elektormagazine.com/180348-01



Power Outage Detector with Text Messaging

ElektorLabs magazine 5/2018, p. 12 (180344)

UPDATE. Errata and notes on the parts list:



- L2 is a type BLM21AJ601SN1D (not BLM12...).
- MOD2: The Quectel GSM module can be used in both M95FA-03STDN and M95FA-03STN versions.
- X1 (CSTCC8M00G53-R0, ceramic resonator, 8 MHz, Murata) is available from Farnell, TME, Mouser, Distrelec and other distributors.

Lighthouse 2.0

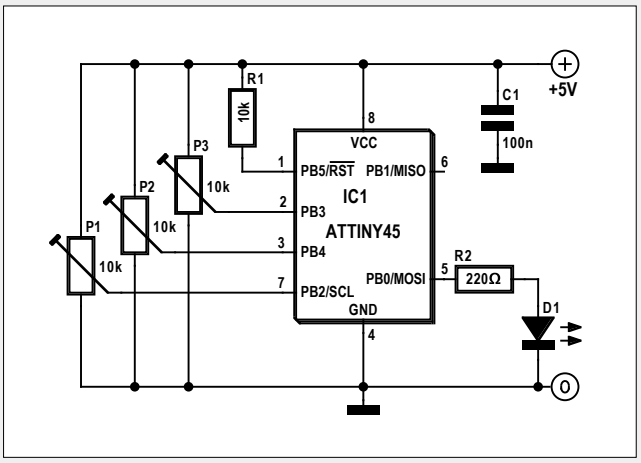
ElektorLabs magazine 1/2019, p. 24 (180334)

FEEDBACK. In an extensive endurance test, it turned out that software crashes can occur after an indefinite period of time. The more exact investigation showed that the acquisition of the potentiometer values within the DO-LOOP loop is the problem. A relocation of the DO-LOOP loop in a new software version [see www.elektormagazine.com/180334-02] now causes the potentiometer values to be queried only once at the start of the program. If necessary, the circuit must be completed with a reset.

The adjustment now follows the procedure below:

First set all three potentiometers to the lowest value. Now the desired flash length follows with P1. Next it is recommended to set the number of flashes/links with P3. Since the return is still at minimum, the program automatically sets it to a value that lasts 2 seconds longer than all flashes/flashes and their interruptions require. Once the number of flashes/flashes has been determined, P2 is used to set the duration of the return to the desired value. This will set the parameters of the lighthouse. To accept the respective setting, press the reset button to restart the program. Alternatively, the supply voltage can be switched off and on again.

Friedrich Lischeck (Author)



Alarm with 3-way Display

ElektorLabs magazine 6/2018, p. 82 (170112)

FEEDBACK. I read in the November and December 12018 issue how sines and cosines are laboriously calculated in two projects (Alarm and DDS).

I often use the Arcus tangent function in my programs, the whole thing runs on a simple 8-bit MCU without much MHz and memory. For the calculation I use the CORDIC algorithm, which is mathematically quite complicated, but as C code (with 16 bit integers) it is quite understandable.

The same algorithm can also be used for Sin/Cos. You need a LUT with N entries, in which the arctan(1/2^*i*), *i* = is precalculated. During *N* iterations a vector is rotated clockwise or counterclockwise by arctan(1/2^*i*) until it reaches approximately the desired angle.

The accuracy and speed is directly dependent on *N*. And only additions and shifts are necessary; all numbers (angle, sin, cos, etc.) are interpreted as integers.

Roughly speaking the Sin/Cos-CORDIC looks like this:

```
for (i=0; i
{
  if (angle < 0)
  {
    int temp = sin;
    sin -= cos >> i;
    cos += temp >> i;
    angle += lut[i];
  }
  else
  {
    int temp = sin;
    sin += cos >> i;
    cos -= temp >> i;
    angle -= lut[i];
  }
}
```

What is missing here, besides the LUT, are the start values for sin, cos and angle. These depend on the concrete situation.

The CORDIC can do much, much more; therefore the information on the web (Wikipedia etc.) is quite mathematical and (for electronics engineers) rather incomprehensible. The best I found for the Sin/Cos version is this one:
http://nonagon.org/ExLibris/sites/default/files/pdf/Cordic-Faster-Sin-Cos.pdf

Even better understandable is the similar arctan2 version described here:
http://bsvi.ru/uploads/CORDIC--_10EBA/cordic.pdf

Of course, the LUT in the alarm clock article also has advantages (speed). But it needs a lot of memory and is not very flexible. On the other hand, the sin()/cos() functions in the math.h lib are very slow and resource-hungry (floating points).

By the way, thanks for the great articles, I can always learn something new!

Andreas Dick



Experimenter's Transistor Tester

Elektor magazine 2/2015 (March & April), p. 56 (130544)

FEEDBACK. Although the Platino-based transistor tester from Elektor 2/2015 is now an older project, in my opinion there is nothing against suggestions for improvement. Anyway, the board is still offered in the Elektor Store and so I bought it about two years ago. There were no major problems with the setup, only there seemed to be something wrong with the software. If a transistor was connected, it was measured without problems. However, if the terminals were without DUT, there was no feedback. Something seemed to happen after pressing the button in the first line of the display, but only keeping the button pressed produced a pale "No transistor".
The improvement was quickly implemented with my BASCOM full version: After line 320: `Lcd "No transistor"` the line `Waitms 1000` was inserted and the text "No transistor" remained visible. However, after that the values were displayed again, which came from the previous measurement with connected transistor. So the variables had to be deleted after the newly inserted line:

```
321: Waitms 1000
322: I = 0
323: T_type = 0
324: Cal_pin = 0
```

Super, everything worked!
Since I had discovered in the source code that the measured values are also output via the serial interface (unfortunately the article only gives a brief overview), was triggered. So I connected a USB-to-serial module and started a terminal window. It took me some time to realize that the speed of the serial port in the terminal was set incorrectly; there was simply a baud = 9600 missing in Line 14 of the source code. Once inserted and recompiled everything seemed to work. Until the cable between serial interface and PC was cut and a new test of the transistor tester showed that it didn't react anymore. At this point I remembered a suggestion from Clemens Valens, who started troubleshooting from a "floating serial port" 2 years ago. So I put a 100-kΩ resistor to VCC on PD0 (TXD) and PD1 (RXD) of the ATmega32 (photo). And hey presto: Everything worked smoothly.



Hans Schneider

Editor's note: The software version by Hans Schneider (V3.1) can be downloaded at:
www.elektormagazine.com/labs/130544-1-platino-transistor-tester

180573-B-02

ELEKTOR STORE HIGHLIGHTS

Measuring, powering and printing…

…with gear from the Elektor Store

Multimeter with Bluetooth module

An electronicist needs at least two pieces of equipment to be able to work on an electronic circuitry: a multimeter and a soldering iron. Most of us will have these by now, but sometimes it is necessary to acquire a new one or obtain an additional multimeter that will come in handy when it is desirable to make multiple measurements simultaneously.



There is now an abundance of multimeters available, from very cheap instruments starting at ten euros or so to thousands of euros. Considering robustness and safety, it is probably best to stay away from the cheapest models.

When it comes to features, the MS8238H from Mastech that we describe here is a mid-range instrument with all the basic functions that you can expect from such a device: voltage (DC and AC), resistance, diode test and continuity beeper. In addition there are a temperature range (a thermocouple is supplied), two battery measuring ranges (for 9 V and 1.5 V) and an NCV position to sense high (mains) voltages without making contact. Some positions of the rotary switch have multiple functions that can be selected with the Func pushbutton. With the Hold button the reading on the display can be frozen. The Ran button is used to select the sensitivity of a measuring range manually. Normally the meter automatically selects the most optimum range.

Finally there is the blue Link button on the meter, which is used to activate the built-in Bluetooth 4 module. This will then transmit the measured value and the position of the rotary switch to the accompanying app on a smartphone or tablet. This is quite unusual for a multimeter that is not quite 50 euros!

The 3¾-digit display has no backlight, but at low light levels the display is still quite easy to read. The basic accuracy is typical for this price category, with 0.5% for the DC voltage ranges. The AC ranges use true-RMS measurements.

@ WWW.ELEKTOR.COM

→ Mastech multimeter type MS8238H
www.elektor.com/
mastech-ms8238h-digital-multimeter-with-wireless

Power to the Raspberry Pi

Add-on boards for powering a Raspberry Pi have been available for a while already. Nevertheless, the StromPi 3 has a few unique features to offer. It is supplied with two stand-offs, two screws and two nuts, plus a very brief manual in German and English. With the aid of this, the mounting of the HAT does not create any insurmountable problems.

The StromPi 3 supports the StromPi 3 Battery HAT, a 1000 mAh LiFePo4 battery with a nominal voltage of 3.2 V (3.2 Wh). This

can simply be plugged into the StromPi 3 (when not powered up) and can then power the Raspberry Pi.
The StromPi 3 is provided with several voltage inputs for the Raspberry Pi. One of these has a very wide range of 6 to 61 V; furthermore, it also has a USB port with a buffer capacitor of 1 F and a connection for a LiFePo4 battery. Additionally, the HAT has an RTC, so that the Pi can continue to keep time during a power or mains failure. The StromPi 3 is therefore an excellent resource when the Raspberry Pi has to be powered from a typical 12-, 24- or 48-VDC distribution system.
For the event that one of the power sources to the StromPi 3 fails, the order in which the remaining sources are used as a backup is configurable. There are feed-through connectors



for stacking multiple HATs, that also connect the PoE connections of the Pi 3B+. What is remarkable, are the USB-A and two microUSB connectors on the side. These can be used to connect devices to the StromPi 3 that require more than 500 mA (such as a hard disk). While it is true that the StromPi 3 was specifically designed for the Raspberry Pi, but because of its construction it can nevertheless also be used with other SBCs, such as the BeagleBone Black.
The StromPi 3 is both a UPS (when combined with the battery HAT) as well as a universal power supply for the Pi, with the emphasis on power supply. In any case, the hardware has been very well thought through.

Anycubic Photon 3D-printer

The Anycubic Photon is a DLP 3D-printer (Digital Light Processing) that prints objects using a liquid resin. UV light projects an image from underneath a transparent bath, containing a liquid resin, that then builds the object layer by layer, while slowly raising the platform. Compared to the conventional FDM 3D-printer, using a hot nozzle to extrude plastic wire into a particular shape, this is actually a simple process without many moving parts (only the bath with resin moves up and down).
The printing is straightforward with the aid of the supplied, complementary copy of the Photon slicer software. Note: the the printer has to be exactly level. The printing itself takes several hours, independent of how many objects are being printed simultaneously. Once the printing has finished, the objects have to be rinsed in alcohol and need to finish curing for a while exposed to daylight.
A disadvantage is possibly that the resin (which is available in different colours) is not all that cheap. Nevertheless, this printer, when considering the price/quality ratio, is one of the best choices in the printer market at this time. .

180689-B-01

Tested by elektorlabs



@ WWW.ELEKTOR.COM

→ StromPi 3 – Power to the RPi
www.elektor.com/strompi-3-power-solution-for-raspberry-pi

@ WWW.ELEKTOR.COM

→ Anycubic Photon 3D-printer
www.elektor.com/anycubic-photon-uv-resin-dlp-3d-printer

Hardware Design using (V)HDL (3)

Driving a WS2812 LED matrix

By Jörg Zollmann (Germany)

In the previous installment of this series we built a digital watch with a binary output. This time we will equip the watch with a display in the form of a matrix of 64 pieces of WS2812 RGB LED. Fortunately the signal required to drive the WS2812 LEDs adheres to a very simple asynchronous protocol. Our hardware driver implemented in VHDL will include, among other things, a state machine.

A matrix comprising 64 WS2812 RGB LEDs is available from various online retailers from around £5 (US$7). Note that a WS2812 LED is more than just a simple RGB indicator: besides the three individual LEDs the tiny package also contains drivers for them (in the form of current sources) and a simple controller. We can communicate with the controller using a serial one-wire protocol to tell it how brightly to light each of the three internal LEDs. On the Internet you will find any number of microcontroller projects and software libraries that concern themselves with driving WS2812 LEDs: [1][2]. Here we will go one step closer to the metal and code up a hardware driver in VHDL to talk to the controller built into the LED modules.
As in the previous installments of this series, the software can be downloaded from the web page accompanying this article [7].

Table 1. Data transfer timings.

Name	Description	Duration (TH+TL = 1.25 µs ± 150 ns)
T0H	0 code , high voltage time	0.35 µs
T1H	1 code , high voltage time	0.9 µs
T0L	0 code , low voltage time	0.9 µs
T1L	1 code , low voltage time	0.35 µs
RES	Reset-Code, low voltage time	> 50 µs

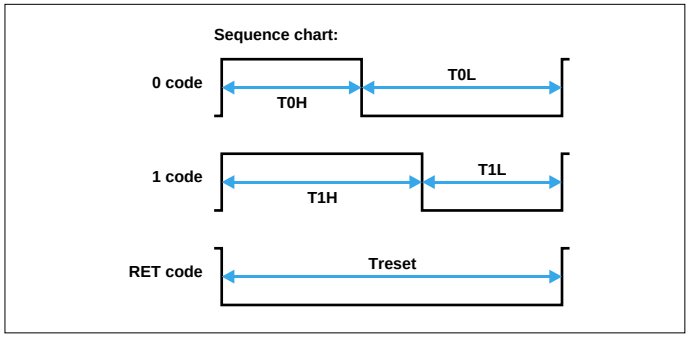


Figure 1. WS2812 data format [3].

The structure of the source code and the hardware platform used were described in the first installment [4].

The WS2812 protocol

In order to design a hardware driver in VHDL to communicate with an external device we need to get to grips with the internal details of that component, and that will require detailed knowledge of the protocol it uses. In the case of the WS2812 LED we are fortunate that a simple asynchronous method is used. The protocol is designed to allow LEDs to be daisy-chained: each LED receives 24 bits of data over its DIN port, and forwards all subsequent data bits to its DOUT port, which can be connected to the next LED in the chain [3].
An 8-bit value from 0 to 255 is used for the intensity of each of the three RGB components: these values are used to control the current output of the internal LED driver circuits. After reception of the first 24 bits, which are latched internally, any further bits that are received are passed on, buffered, to the DOUT output. Because of the signal conditioning of the output that takes place data transfer is very robust and the protocol can in principle be used to drive a chain of as many LEDs as you like. The only limitations will be from the total current consumption and in the maximum frame update rate that can be achieved.
The first incoming value received after a reset code (which is detected by the input signal line being held low for at least 50 µs) will be stored. The data bits themselves are encoded as 'logic 1' and 'logic 0' pulses of different widths: see **Table 1** and **Figure 1**. The total time allocated to one bit is 1.25 µs. If during this period the signal line is high for 0.35 µs and then low for 0.9 µs, then a 'logic 0' is detected; if the signal line is high for 0.9 µs and then low for 0.35 µs, then a 'logic 1' is detected. The color components are sent green first followed by red and blue, and within each component the eight bits are sent in order from MSB to LSB.
If we use a clock frequency of 50 MHz then the time resolution that we can achieve with our CPLD is 20 ns. The WS2812 protocol requires that the pulse durations are within ±150 ns of their nominal values, and our pure VHDL implementation will also need to comply with this specification. Experience

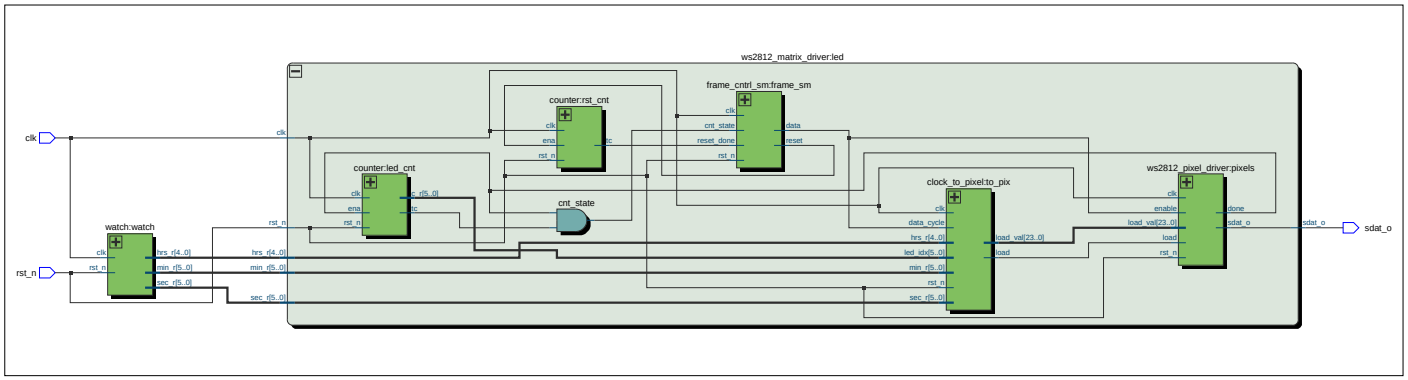


Figure 2. The WS2812 time display in RTL View.

with the WS2812 LEDs shows, however, that they are very tolerant towards timing errors in the serial protocol, as long as the duration of the 'logic 0' pulses is not too great. It is also observed that gaps in the pulse train rather shorter than 50 µs can be interpreted as reset codes by the controller in the LED.

Pixel driver

The WS2812 driver that we will be describing here will be used to display the time in binary format. It is constructed in a hierarchical fashion and consists of a number of modules (see **Figure 2**). The top-level module instantiates the watch design from the previous installment [8] as well as an instance of the module `ws2812_matrix_driver`. The inputs to this latter module include the data values from the watch registers (hrs, min, sec) that are to be shown on the 8x8 display. Inside the matrix driver module there are further components, each responsible for carrying out a different task.

The `ws2812_pixel-driver` module is designed in such a way that it can work with LED strips as well as with larger matrix panels (and so there is plenty of room here for further experimentation). The function of the pixel driver can be described as follows. At the start of the period allotted to one pixel a 24-bit shift register (`do_reg`) is loaded with the brightness values for the LEDs: more precisely, the `load` signal must be high for one clock cycle to copy the value from the `load_val` input signal to the shift register. Then the contents of the shift register must be clocked out one at a time: see **Listing 1**.

The protocol requires us to convert logic 0 and logic 1 states to pulses on the data line. This is achieved using two processes. In the first process, `code_proc`, the two 'codes' for logic 0 and logic 1 are generated continuously (see **Listing 2**). Which code actually makes it to the output pin is determined by the most significant bit (bit 23) of the shift register; the other bits are subsequently shifted in turn into this position. The output multiplexer responsible for connecting the output signal `sdat_o` to the correct code signal depending on the value in the shift register is shown in **Listing 3**.

Listing 1. Shift register.

```
shift_proc: process (rst_n, clk)
  begin
    if (rst_n = '0') then
      do_reg <= (others => '0');
    elsif rising_edge(clk) then
      if(enable= '1') then
        if (load = '1') then
          do_reg <= load_val;
        else
          if (bit_done) then
            do_reg <= do_reg(22 downto 0) & '0';
          end if;
        end if;
      end if;
    end if;
  end process;
```

Listing 2. Bit encoding process.

```
code_proc: process (rst_n, clk)
begin
  if (rst_n = '0') then
    zero_one_count <= 0;
    one_code <= '1';
    zero_code <= '1';
  elsif rising_edge(clk) then
    if (enable = '1') then
      if (bit_done)then
        zero_one_count <= 0;
        one_code <= '1';
        zero_code <= '1';
      else
        zero_one_count <= zero_one_count + 1;
        if (zero_one_count >= c_CLKS_T0_H_T1_L)
   then
          zero_code <= '0';
        end if;
        if (zero_one_count >= c_CLKS_T1_H_T0_L)
   then
          one_code <= '0';
        end if;
      end if;
    end if;
  end if;
end process;
```

Listing 3. Output multiplexer for sdat_o.

```
mux : process (all) is
  begin
    case (out_sel ) is --
      when "00" => sdat_o <= '0';
      when "01" => sdat_o <= zero_code;
      when "10" => sdat_o <= '0';
      when "11" => sdat_o <= one_code;
      when others => sdat_o <= '0';
    end case;
  end process;
```

The rest of the logic in the pixel driver consists of counters similar to those we learned about in the previous installment. The counters are configured to various bit widths and count various types of event. There is a counter that indicates when a bit has been shifted out (which occurs every 1.25 µs), a counter that indicates when an entire byte has been shifted out, and a counter that indicates when a pixel is complete.

From clock to pixel

There are two further counters in the `ws2812_matrix_driver` module. One of these counters indicates when the duration of the reset code has been reached, and one counts the number of pixels that have been output so far. Since the matrix of pixels is actually connected as a single string, this in effect assigns a pixel number to each of the 64 LEDs. The conversion of the counter value to the required 24-bit pixel value is done in the module called `clock_to_pixel`. This module uses a synchronous process to ensure that the shift register in the pixel driver is always loaded with new data at the correct moment. After the pixel number is incremented it is compared against various values. If the pixel number is smaller or greater than the range of pixel numbers that are to be used for the time display, the background color `led_bg` is loaded into the shift register. The time display requires four pixels for the hours and six pixels each for the minutes and seconds. Reading the time in binary is tricky enough for humans already, and so each part of the display is given a whole row in the LED matrix. The rows that are used are specified by constants, for example as follows.

Listing 4. Frame control state machine written as a single process.

```
-- Build an enumerated type for the state machine
type state_type is (data_cyc,reset_cyc);
-- Register to hold the current state
signal frame_state: state_type;
...
frame_ctrl: process (rst_n, clk)
  begin
    if (rst_n = '0') then
      frame_state <= reset_cyc;
    elsif rising_edge(clk) then
      case frame_state is
        when reset_cyc => -- when current state
   is reset_cyc
          frame_state <= reset_cyc;
          if(reset_done) then
            frame_state <= data_cyc;
          end if;
        when data_cyc => -- when current state is
   data_cyc
          frame_state <= data_cyc;
          if(cnt_state) then
            frame_state <= reset_cyc;
          end if;
        when others => frame_state <= reset_cyc;
      end case;
    end if;
  end process;
```

```
constant c_hrs_idx: natural := 15;
```

We start with the hours display in the third row of the matrix (the first two rows are not used). If one of the bits representing the time is a 1 then a value corresponding to yellow (`led_on`) is loaded into the shift register; otherwise the LED is turned off (`led_off`). The trailing LEDs (numbers 5 to 8 or 7 and 8) in each row are always off.

In order for the comparisons against the current pixel number to work properly, we need to take care with the various data types that are involved in the comparison. The output of the pixel counter (`led_cnt`) is a signal of type `unsigned`, whereas the constants against which it is compared have type `natural`. For such a comparison to be permitted, we need to use the function `to_integer` from the `numeric_std` library, for example as follows.

```
if (to_integer(led_idx)) = c_hrs_idx +0) then...
```

A handy overview of the built-in type conversions in VHDL can be found, for example, at [5] and [6].

State machines

The rest of the functionality of the system is implemented in the module `frame_cntrl_sm`, which describes a finite state machine. A finite state machine, or FSM, is a sequentially-controlled logic circuit that is designed to orchestrate repetitive events and sequences of events. Hardly any digital hardware design of any size will not use a finite state machine at some point. The machine consists of a sequential part formed from a register, which stores the current state of the machine, and one or two combinatorial parts which use the current state to determine the next state and the new value of the outputs. Depending on the application, it can be worthwhile to synchronize the inputs on entry to the machine and possibly also to store the outputs in an intermediate register.

Finite state machines are divided into two types, known as Moore machines and Mealy machines. The differences between these two classes are apparent in both their truth tables and in their state transition diagrams. While the outputs of a Moore machine are a function only of the current state, in a Mealy machine the outputs are a function of both the current state and the current input data. It is possible to convert between

the two types of machine with the help of a little Boolean algebra. In practice (or at least in the opinion of the author) distinguishing between the two types of machine is not of great importance; what matters more is solving the concrete problem at hand. Experience shows that an implementation using a Moore machine generally requires more states ('Moore is more') but is often easier to understand.

There are several ways to approach implementing a state machine in VHDL. The first one we shall look at is the standard textbook method which involves dividing the description of the state machine into two separate processes. One process is responsible just for storing the newly-calculated state vector, and this encapsulates the sequential part of the machine. The second process contains the combinatorial logic, and this computes the next state to which the machine will transition. The alternative approach is to use one process to encapsulate both the sequential and combinatorial parts that are required to store the current state and compute the next one, and a further process which contains any further logic that may be needed to generate the outputs. This approach has the advantage that it is easier to write and more compact, as well as being easier to understand.

Listing 4 shows the source code of the finite state machine that monitors the progress of emitting a frame in the driver. This state machine has just two states, indicating whether a pixel is in the process of being output or whether the reset code is being emitted. For the individual states it is possible to create a 'user-defined type' in VHDL using the `type` keyword. For example, the line

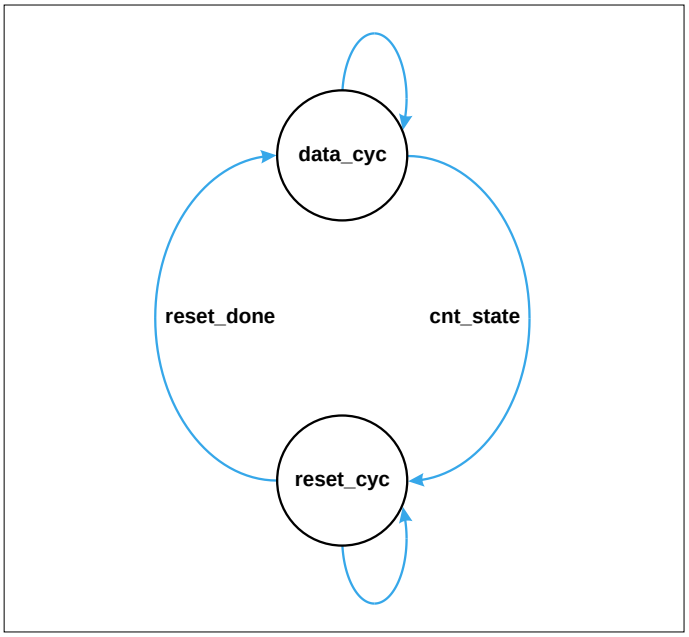


Figure 3. State machine to monitor the progress of a frame.

```
type state_type is (data_cyc,reset_cyc);
```

creates a new type called `state_type` based on the standard enumeration VHDL data type.

It is customary to document the states of the machine using a

Advertisement

Join the Elektor Community
Take out a GOLD Membership now!

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

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

Also available:
The all-paperless GREEN Membership!
www.elektor.com/member

state diagram, and **Figure 3** shows the state diagram for the frame-level state machine. ModelSim and Quartus have the facility to display a state diagram for a state machine as long as the state machine is recognized as such. If, when compiling the VHDL file for ModelSim the command-line switch `+cover` is supplied, most state machines will be automatically recognized. For Quartus it is additionally necessary to adhere to a prescribed coding style; for example, the state machine for the WS2812 driver does not get recognized (perhaps because it only has two states).

Brightness

Each individual LED can, according to data on the Internet, draw 20 mA at full brightness. We have 64 x 3 LEDs, which makes a total of getting on for 3.9 A, enough to bring a simple plug-in breadboard power supply to its knees. Therefore, when starting development of the driver, the author used a separate power supply that was capable of delivering enough current for the panel. However, although the LEDs can draw a lot of current, when they are driven at full brightness they are too bright for a time display. We can therefore reduce the brightness values for the LEDs. The colors and brightnesses for the LEDs can be changed by adjusting the constants in the `clock_to_pixel` module, which take the form of arrays of `std_logic` vectors. We define a separate constant for each different color that we wish to display.

```
type grb_lut is array (0 to 2) of std_logic_vector(7
   downto 0);
constant led_off: grb_lut:= (x"00", x"00", x"00"); --
   all off
constant led_bg: grb_lut:= (x"00", x"10", x"00"); --
   acceptable red
constant led_on : grb_lut:= (x"1F", x"1F", x"1F"); --
   acceptable yellow
```

The default value of 31 (= 0x1F) that we have chosen makes for an perfectly acceptable display, and now also the 700 mA output of the breadboard power supply is adequate (see **Figure 4**). The serial data line of the LED matrix is connected directly to the output of the CPLD (pin 54 of the CPLD, or pin 5 of K1). According to the data sheet for the LED matrix this strictly speaking should not work as the minimum input voltage for a high signal is 0.7 VDD = 3.5 V, which is 200 mV above the maximum 3.3 V we can expect at the LVTTL output. Depending on the LED matrix in use, therefore, it may be necessary to insert a level shifter.

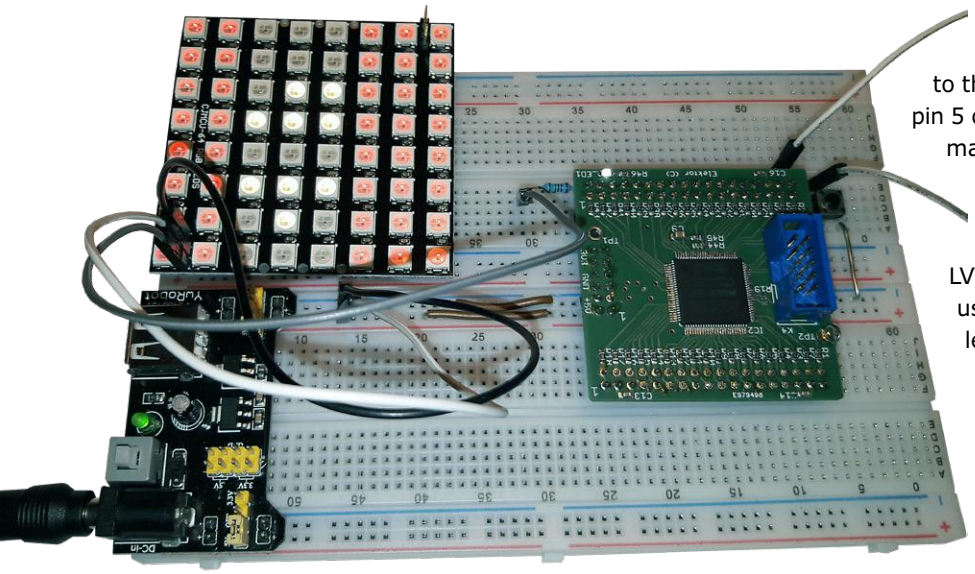


Figure 4. Time display on an 8x8 panel.

Outlook

In this installment we have for the first time looked at the construction of a larger-scale digital circuit.

A glance at the 'fitter resource usage summary' in the Quartus compilation report will show that our design fills 36 % of the CPLD. There is therefore still a certain amount of room for expansion. In the next installment we will look briefly at timing constraints, which are necessary to ensure a working circuit. Plus, we will also program a simple game in hardware. ⏮

(180285-C-02)

@ WWW.ELEKTOR.COM

→ CPLD Breakout Board
www.elektor.com/cpld-breakout-board-160425-91

Web links

[1] Driving the WS2812: https://cpldcpu.wordpress.com/2014/01/14/light_ws2812-library-v2-0-part-i-understanding-the-ws2812/

[2] 'ESP8266 Web Server for NeoPixel LED Strips', Elektor September & October 2018: www.elektormagazine.com/180355-02

[3] WS2812 data sheet: https://cdn-shop.adafruit.com/datasheets/WS2812.pdf

[4] 'Hardware Design using (V)HDL (1)', Elektor 5/2018: www.elektormagazine.com/180285-02

[5] VHDL type conversion: www.nandland.com/vhdl/tips/tip-convert-numeric-std-logic-vector-to-integer.html

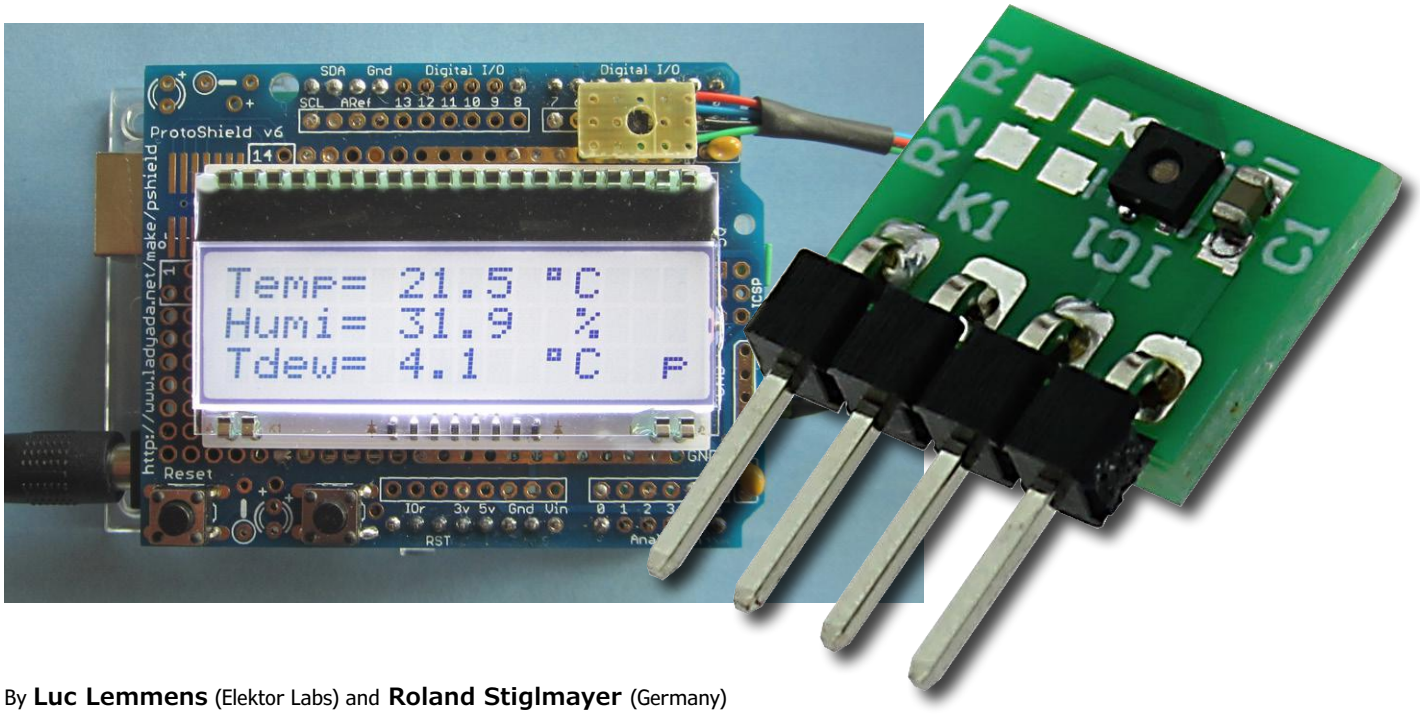
[6] VHDL vector arithmetic: www.doulos.com/knowhow/vhdl_designers_guide/numeric_std/

[7] Article support page: www.elektormagazine.com/180285-C-02

[8] 'Hardware Design using (V)HDL (2)', Elektor 6/2018: www.elektormagazine.com/180285-B-02

An ENS210 Dew Point Sensor e-BoB

With a sample implementation on Arduino UNO



By **Luc Lemmens** (Elektor Labs) and **Roland Stiglmayer** (Germany)

Values measured for a physical parameter called 'the dew point' indicate not only the risk of mould growth but also corrosion of steel objects including tools. Ventilating and heating afford some control over the dew point at a certain spot. However, to do this in a sensible way, the dew point has to be determined beforehand. That is exactly what this smart device is doing.

Project Highlights

- Easy to use breakout board (BoB)
- High accuracy, miniature, temperature and humidity sensor
- sample application for Arduino

First and foremost, however, this device is a very precise measuring instrument. The high measuring accuracy of temperature and humidity makes it very useful. The sensor used here, a type ENS210 from a m s, is probably the most precise, precalibrated sensor currently on the market. The temperature is measured with an accuracy of ±0.2 °C in a range of 0 °C to 70 °C without any calibration. This makes the ENS210 suitable for use as a reference for the calibration of temperature and humidity sensors.

What's that "dew point"?

The dew point is calculated from the measurement results for temperature and relative humidity. The 'Magnus' dew point approximation is used for this, and readers interested in the maths and theoretical backgrounds should not fail to read up the Wikipedia entry [1].
Additionally, the difference between the dew point and the current temperature is determined, which gives a very good indication of whether mould growth or corrosion is likely to occur.

Caution: tiny object ahead

Your desire to get cracking with the latest ICs may be frustrated by their extremely small size, often dictated by the industry as a result of "customer demand". The ENS210 sensor chip is no exception with its package of only 2×2 mm (**Figure 1**). Fortunately, as you can see from the photo in **Figure 2**, with some effort the author managed to solder the tiny chip onto a piece of breadboard (stripboard, Veroboard, ...).

PROJECT DECODER

ENS210
Arduino
BoB

entry level
➡ intermediate level
expert level

0.5 hour approx.

Arduino UNO,
IDE,
solder iron

€15 / £12 / $18 approx.

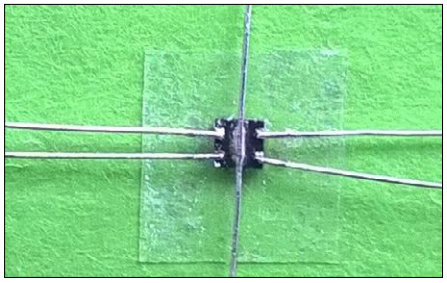


Figure 1: Tiny, tinier, tiniest... the ENS210 dew point sensor from a m s is very small indeed. (bottom view; author photograph)

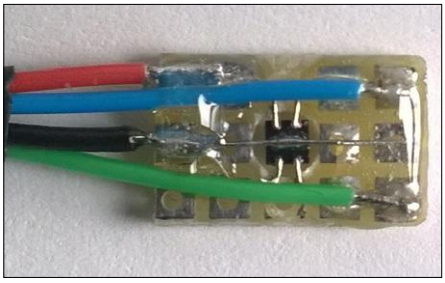


Figure 2: An improvised yet entirely successful method of fitting the ENS210 onto breadboard. (author photograph)

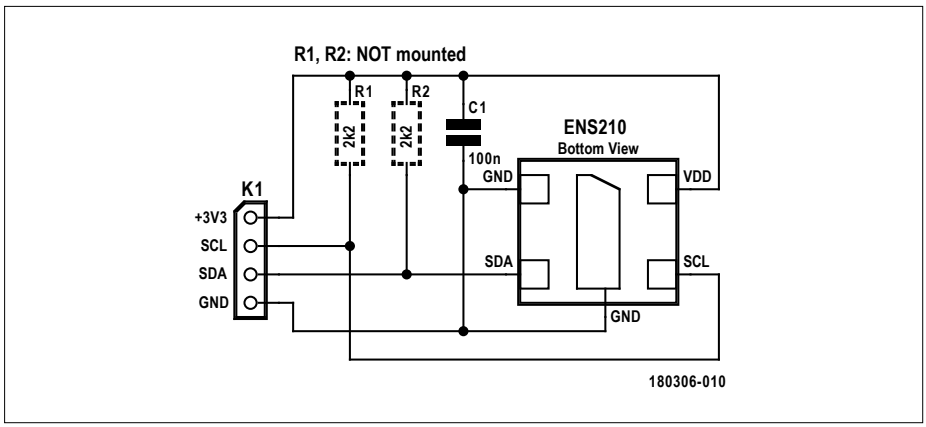


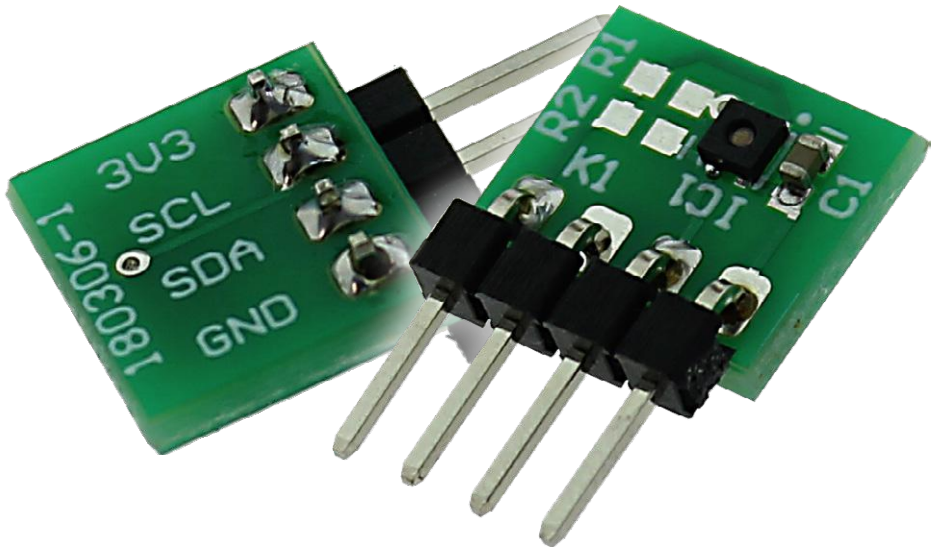
Figure 3: Circuit diagram of the ENS210 e-BoB. I²C bus line pullups R1 (SCL) and R2 (SDA) are optional.

Frustration about the impossibly small size of today's ICs is one thing, despair and rage about being able to handle them at all, is worse. That's why, in good tradition, Elektor Labs set out to design a breakout board (e-BoB) for this exquisite sensor. Our e-BoB avoids all the risks of damaging, burning, or even losing the ENS210 by handling and soldering the device, or vacuuming the workplace, and gives easy access to the sensor's valuable data.

The Elektor ENS210 e-BoB

The impossibly :-) simple schematic of the ENS210 e-BoB is pictured in **Figure 3**. This follows the manufacturer's recommended setup with no more than a 3.3-V decoupled supply rail (C1) and (optional) pull-up resistors R1 and R2 on the SDA and SCL lines as part of the I²C interface. R1 and R2 are supplied at 3.3 V and should be mounted:

- when no other pull-ups are present on the system's SDA and SCL lines;
- when voltage limiting is required due to a component on the I²C bus operating at 5 V, like an ATMEGA with its internal pull-ups resulting in a safe High level of +3.6 V at the sensor's SDA and SCL pins.



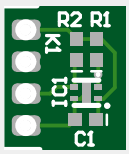
COMPONENT LIST

Resistors
R1;R2 = see text

Capacitor
C1 = 100nF, X7R, 0603

Semiconductor
IC1 = ENS210-LQFM, (a m s) ,
e.g. Mouser # 985-ENS210-LQFM

Miscellaneous
K1 = 4-pin SIL pinheader, 0.1" pitch
PCB, 180306-1 v. 1.0, from Elektor Store



Under no circumstances should the sensor be operated without pull-ups on the SDA and SCL lines.
Finally, all ENS210 data traffic and powering at 3.3 V takes place through 4-way pinheader K1.

The Arduino connection

The author hooked his ENS210 sensor to an Arduino UNO to create a practical dew point meter and kindly reported on it on the Elektor Labs site. Thanks to the use of an UNO, the hardware required for his project proved a simple affair, with just one challenge, possibly: the few electrical connections to the LCD display. The easiest solution here may be to revert to using a Proto shield, which has a socket to connect the sensor via a cable.
Figure 4 shows the schematic of the author's direct-reading dew point meter with quite some bells and whistles, designed to run his home-developed software. The resulting program, or 'sketch' in Arduino-speak, is freely downloadable [2]. Note that this circuit and the Arduino software are not associated with an Elektor Labs designed PCB, hence this part of the article is strictly in the **Homelab Project** category.
The ENS210 providing a full-blown I²C interface, hardly any additional hardware is required. Whilst all measurement results appear on a display, they are also sent to the serial monitor or the serial plotter of the Arduino IDE. The serial output format is selectable for either the monitor or the plotter by means of a pushbutton.
An attractive 3-line LCD display with white backlight is used, from a company called Electronic Assembly, but you may want to use one of your own choice. This 3-line LCD display is supported by the Arduino sketch 'ENS210_3L_34'. With only a few additional functions, many chunks of the standard library 'LiquidCrystal' can be used for further development of the project. The library itself does not need to be altered. Of course, it is also possible to use an Elektor LCD shield together with the Arduino sketch 'ENS210_2L_25'. Other suggestions to extend this project and its functionality include some pushbuttons to control a data logger function. This function would store the measured data non-volatile in the EEPROM.
Finally, note that MOD2 in the schematic is the author's homebrew sensor PCB pictured in Figure 2, i.e. not the Elektor e-BoB which has a different pinout. Rewiring should be a simple affair though with just +3V3, SCL, SDA and GND to match between BoB and the UNO 'host' circuit. ⏮

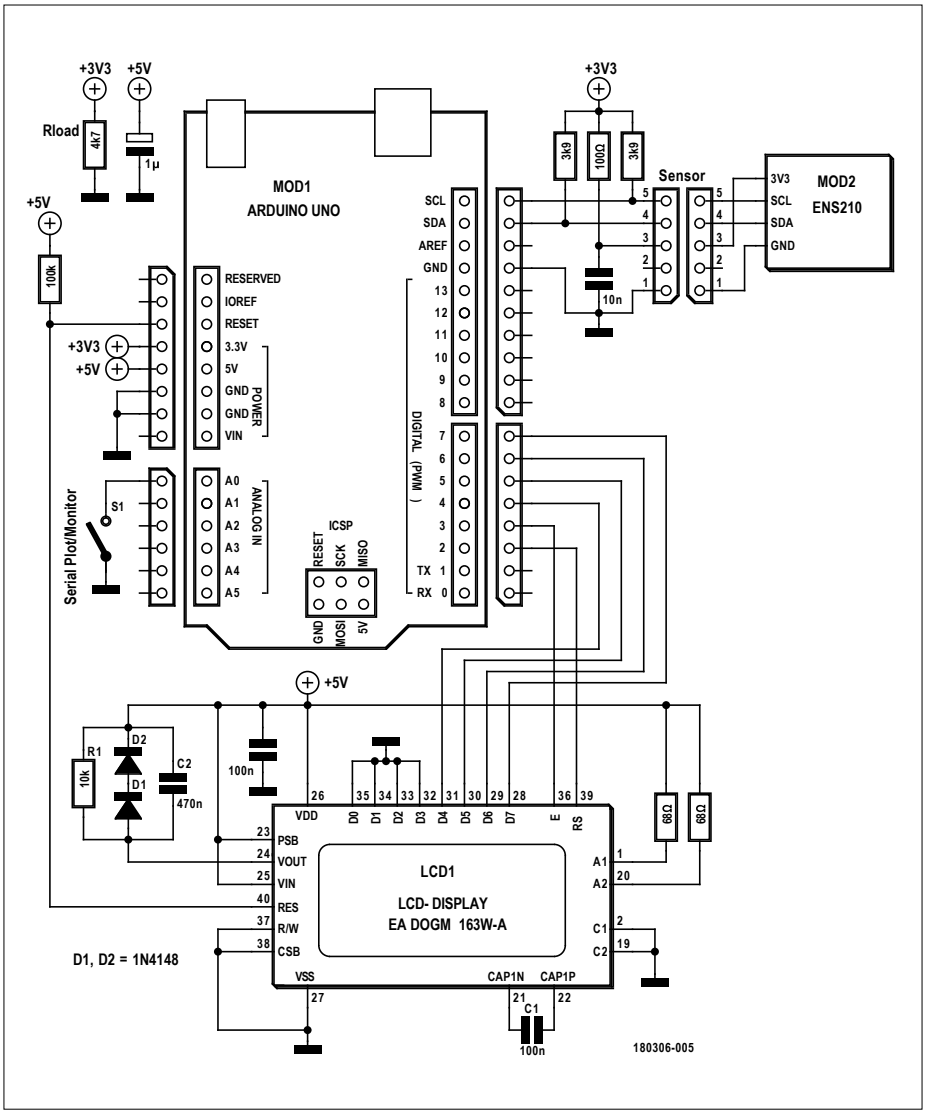


Figure 4: Author's design of an Arduino 'host' system for the ENS210 dew point sensor. The two combined, with an LCD added, make an interesting instrument for analysing areas potentially affected by mould growth and corrosion. If the boost function of the display is not required, omit C1, C2, D1, D2, R1, and connect pin 24 to pin 25.

180306-01

@ WWW.ELEKTOR.COM
→ Dew Point Meter e-BoB assembled board
www.elektor.com/180306-1

Web links

[1] Dew point, theoretical background: https://en.wikipedia.org/wiki/Dew_point
[2] Arduino sketches: www.elektormagazine.com/180306-01

(Almost) everything you always wanted to know about...

Enclosures (2)

With an attractive, made-to-measure housing, your electronics project will have a professional look. This used to be a difficult job, but nowadays it has become a lot more accessible.

By **Ilse Joosten** (Belgium)

Q *Which materials can we use to design enclosures?*

A There is a very wide range of useful materials for enclosures. In the context of this Q&A, we limit ourselves to the more common, easily available and easy to process materials.

- **PMMA, polymethyl methacrylate (acrylic, plexiglass, perspex™)**

PMMA is available in a wide range of colours and thicknesses and this is transparent, translucent and opaque. It is important to note that PMMA is available in two variants, cast (GS) and extruded (XT). What is most suitable depends on the application and the chosen processing method.

Unlike extruded PMMA, which is rolled between rolls to thickness, cast PMMA has considerable thickness tolerances of up to +/- 15%. If mechanical dimensions and fit are important for a design, this should be taken into account and extruded PMMA is preferred.

Cast PMMA is easier to work mechanically (drilling, sawing, milling, ...) and gives better results with laser engraving (frost look). In our experience, PMMA extruded is better suited for laser cutting due to its lower melting point. The cutting edges are more beautiful with a flame-polished effect while cast PMMA often produces rougher cutting edges. [1]

- **Polycarbonate (Lexan™)**

Just like PMMA, polycarbonate is available in different thicknesses but with a more limited range of colours. It is very strong and impact resistant and 250 times stronger than glass. Despite its sturdiness, it is easy to work mechanically and, compared to PMMA, there is less risk of cracks. It is less suitable for laser cutting because it creates ugly cutting edges. Polycarbonate is very suitable as a protective window for displays and is in that application also more scratch-resistant than PMMA.

- **Polyvinyl chloride (PVC, Vikupor™)**

PVC sheets are available as rigid PVC sheets and foam sheets in various thicknesses and colours. The rigid PVC sheets are very dimensionally stable and have a high chemical resistance. Laser engraving and cutting are an absolute no-go with this material because they release hydrogen chloride, a highly irritating and toxic gas, which forms hydrochloric acid in contact with water and affects the laser cutting machine. Mechanical processing is possible without any problems. Because PVC is flame retardant and self-extinguishing, it is a very suitable material for projects with mains voltage and/or power electronics.

- **Plywood**

Plywood is a collective term for laminated board material made of various types of wood. It is sturdier than MDF, but thinner boards tend to warp with fluctuations in humidity and temperature. The material is easy to process mechanically and if the thickness remains limited, laser cutting and engraving is also possible. Thicker (>10 mm) plywood laser cutting is only possible on laser systems with a higher output, otherwise the cutting edges will become carbonized. For laser cutting and engraving, plywood from "lighter" woods such as birch or poplar is particularly recommended (**Figure 1**). Plywood from tropical hardwood gives rather disappointing results.

- **MDF (Medium-Density Fibreboard)**

MDF is very easy to machine mechanically and is mechanically stable (does not warp). Laser cutting is possible with thinner material, but because the cutting edges look dark brown to black not everyone finds that equally beautiful (**Figure 2**).

- **Aluminium**

Aluminium can only be machined mechanically, laser machining is lim-

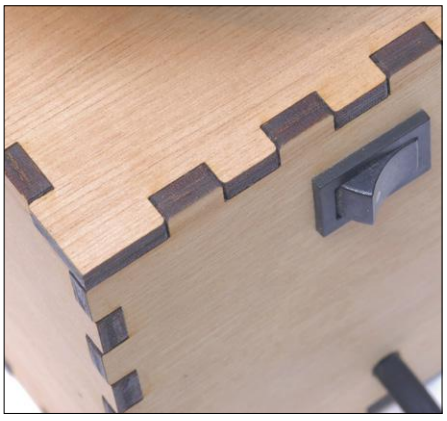


Figure 1. Thin plywood is very suitable for laser cutting.



Figure 2. Cutting wood and MDF with a laser produces brown edges. Not everyone likes that.

ited to marking anodized aluminium or blank aluminium in combination with a chemical product. For easy sawing and (CNC) milling it is recommended to choose an alloy with good machinability. Good available suitable aluminium alloys are for example 6061 T4 and T6 or 6082 T6.

- **Engravable materials (Rowmark, Lasermark, Gravoply, ...)**

These materials are usually rather thin and consist of two (or more) layers of a different coloured material. By lasering or milling away the material from the top layer, the different coloured material from the bottom layer will appear. This way it is easy to make beautiful front panels (**Figure 3**).

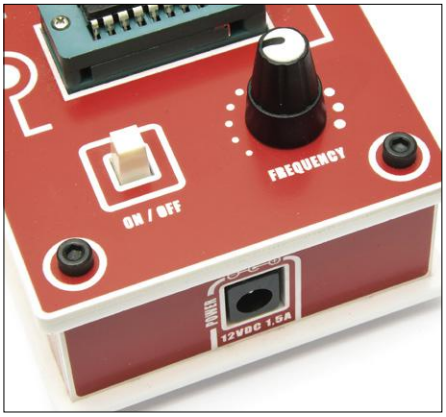


Figure 3. Two-colour laminated material is very suitable for making front panels.

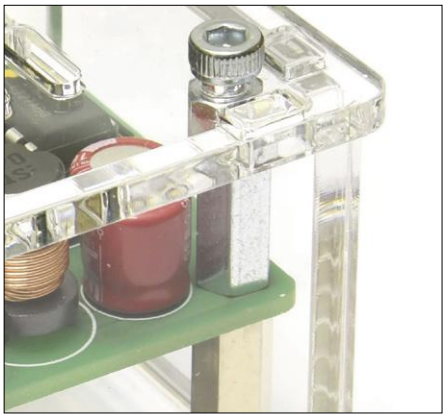


Figure 4. Easy: tabs, spacers, bolts and nuts keep everything together.

Q *How can I edit materials?*

A In the past, unless there was a lot of money at one's disposal, the options were reasonably limited to the more traditional means such as drilling, sawing, filing, ... Now the more advanced techniques have become much more accessible. Of course it is also possible to combine several techniques in one design. An overview:

- **3D printing**

3D printers have been very popular for many years now and are often the first choice when a housing needs to be made to measure. 3D printing is also very accessible, because quite a few printers are attractively priced and there is plenty of free open source software available. A disadvantage is that the technique is too slow for the production of larger numbers of enclosures and that the costs for filament or UV resin can increase. On the other hand, very complex shapes can be made.

- **CNC**

Just like 3D printers, more and more affordable CNC milling machines, whether or not as a kit, are being offered. The better machines can process a multitude of materials, but the learning curve here is a lot steeper compared to 3D printing. Cheap, small, manually operated milling machines are also available as an intermediate solution. These are ideal for occasional rectangular holes in a front panel or existing housing to mill.

- **Laser Cutting**

For the sake of clarity, we are talking here about CO_2 laser systems and not about other systems such as fibre lasers and combined systems with a very high price tag. Unlike 3D printers and CNC milling machines, prices for laser cutting machines remain quite high. In addition to professional production machines with metal RF laser tubes, budget laser systems with glass laser tubes are also available for home use. If you think the prices are still too high and what is adventurous, you can also consider buying a Chinese laser cutting machine. Finally, you can also outsource laser cutting, sending your design to a company that will then produce your parts. This is also possible for CNC milling and 3D printing. You can also always visit a FabLab or Hackerspace.

Another tip: raster engraving with a laser cutting machine is very slow, if speed is important, try to engrave as much vectorially (with lines) as possible.

- **Thermoforming**

Thermoplastics can be bent into various shapes by heating the material (locally) so that it becomes soft. This can be done with hot air, a filament, but also with a laser cutter. The material is heated with a fast moving laser beam from focus after which the gravitation can do its work (laser origami). The latter may not be very practical, but it is a nice idea.

In vacuum forming, a sheet of preheated plastic is pulled over a three-dimensional mould using underpressure. [2]

Q *Which software can I use?*

A Numerous software packages are available for 3D printing, whether free or not and/or open source. Examples are Blender, SketchUp, SolidWorks, Fusion 360, Inventor, TinkerCAD, etc. Some of these packages can also be used for 3D CNC milling, but the CNC machine must also be suitable for this. Many cheaper CNC milling machines only support 2.5D (2D layer 3D structure building) instead of real 3D.

Often all that 3D violence is not immediately necessary and you can also use 2D CAD programs and (vectorial) drawing programs such as DraftSight, AutoCAD LT, InkScape, CorelDraw or Adobe Illustrator. 3D operations on a CNC machine take a lot of time, just like 3D printing. Assembling a three-dimensional housing with simple "2D parts" is simply faster and easier to realize in numbers. By applying materials in layers, you can also create complex enclosures with a professional look.

Q *How can I connect several parts for a housing?*

A The most common methods are fasteners, clamps, click systems and adhesives.

- **Fixing materials**

This includes screws, nuts, bolts, spacers, spacers and connectors. Although this mounting method entails additional costs for the fasteners, it has the advantage that the housing, once assembled, can be easily dismantled later if necessary. The simplest method is to hold the enclosure together with a combination of tabs, spacers and bolts (**Figure 4**). For angled connections, a "T-slot" (**Figure 5**, [3][4]) can be provided, after which another piece of sheet material can be screwed against it at an angle of 90 degrees. Here it is important not to tighten the bolt too tightly.

- **Clamps**

Clamp connections [5] can be made with

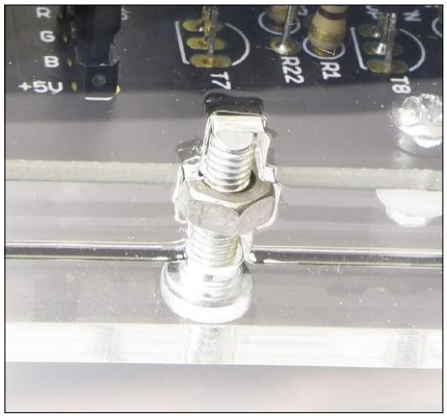


Figure 5. A T-lock makes bolt-nut connections easy.

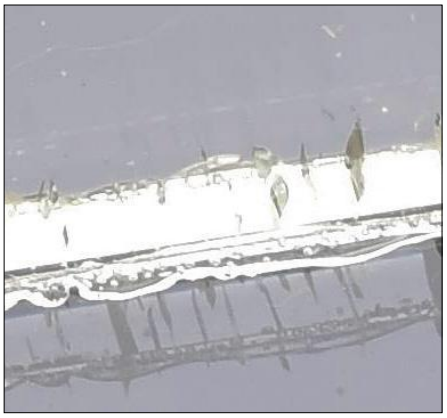


Figure 6. To prevent cracks, PMMA should be "relaxed" after laser cutting before bonding.

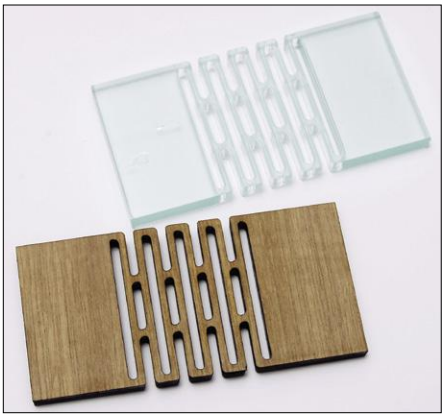


Figure 7. Living hinges can be made with wood or PMMA.

tabs and cutouts or with toothed connections. The idea is to make the tab or part of it slightly larger than the hole it should fit into so that the connection stretches. With toothed joints there should be a minimal difference in size between the teeth and the recesses into which the teeth fall. Clamp connections can be very tight and loosening them again can be difficult without causing damage. There is also the risk that they will come loose spontaneously.

- **Click systems**

Here, toothed tabs are used which, because the material can bend a little, can move. The tab is then bent and forced through a hole, after which the toothed part clicks into place after passing through the hole. 6] This is a vulnerable system, especially when fragile materials such as PMMA are used and it is therefore not suitable for frequent assembly and disassembly.

- **Bonding**

Adhesive joints can be very strong, even as strong as the bonded material itself. Often special products are needed and sometimes also the mastery of specific techniques. These kinds of connections are usually final and sometimes you only get one chance to confirm a part properly. Laser cutting of PMMA creates stresses in the material which can cause cracks during bonding (**figure 6**). To prevent this, PMMA must first be tempered in an oven before bonding.

Q *What are living hinges?*

A A "Living hinge" is a flexible hinge that connects two rigid parts and consists of the same material as the rigid parts. An example is the hinge of the lid of a Tic Tac box. By means of laser cutting or milling, sheet materials can be provided with a pattern of notches, making part or the entire sheet flexible and flexible (**figure 7**). This is possible because rigid materials still have some flexibility. The applications for this are numerous and include not only hinges and flexible connections but also the creation of round shapes. If desired, artistic patterns can also be used for the notches. Keep in mind that laser cutting or milling a living hinge can take time and that living hinges can sometimes be fragile. [7] [8] ⏮

180685-D-02

Web links

[1] Acrylate cheat note: https://makezine.com/2015/10/29/skill-builder-acrylic/

[2] Video on thermo shaping: www.youtube.com/watch?v=P8a2KrEOzxU

[3] Acrylic joints: www.ponoko.com/blog/how-to-make/how-to-make-snug-joints-in-acrylic/

[4] Screw joints: www.instructables.com/id/How-to-Make-Anything-Using-Acrylic-and-Machine-Sc/

[5] Clamp joints: www.ponoko.com/blog/how-to-make/how-to-create-better-nodes/

[6] Click systems: www.defproc.co.uk/blog/2013/so-whats-a-practical-laser-cut-clip-size/

[7] Living hinge on Wikipedia: https://en.wikipedia.org/wiki/Living_hinge

[8] More on living hinges: www.rs-online.com/designspark/laser-cut-living-hinges-for-neater-designs

Balanced Antenna Cable

Peculiar Parts, the series

By **Neil Gruending** (Canada)

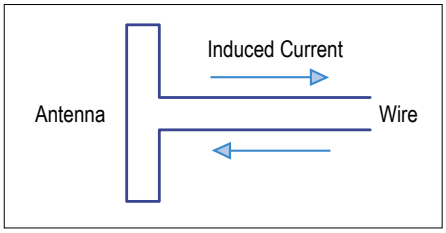


Figure 1. Principle of operation of the folded dipole antenna.

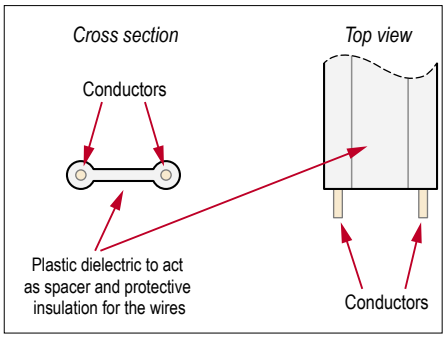


Figure 2. Balanced antenna cable [1].

How do you receive RF signals like FM radio or traditional broadcast television? Since this is Elektor it's much more interesting to use a dipole antenna instead of a dedicated wire connection like cable or the Internet. A dipole antenna works by generating small amounts of current in the presence of an electric RF field which is then sent to the receiver using two-wire conductor called twin lead cable which is a balanced antenna cable. Let's take a look at what makes this cable so special.

The challenge is that the signal levels are very small so interference is a significant issue. Fortunately for us a dipole antenna has a balanced output which means that it works as a current loop at the end of two wires like in **Figure 1**. The signal is the difference between the two wires, so as long as any interference is common-mode then it will be rejected by the receiver. Another characteristic of balanced lines is that they require a constant impedance to work properly. Any mismatches will disturb the current balance and create interference at the receiver.

We are used to using coaxial cable for applications like this but it won't work well in the case because the coax shield needs to be grounded for it to shield the signal conductor from interference. This ground connection disrupts a dipole antenna because it creates an unbalanced condition by adding another current path via the shield. Another challenge for coaxial cable is that it's difficult to match the antenna's 300-ohm characteristic impedance.

Balanced antenna cables like in **Figure 2** solve these problems by creating a transmission line with the two conductors. The spacing between the conductors controls the cable impedance so the cable is made with plastic between them which gives the cable its wide and flat look you may remember from the old days when everyone had their own TV antenna on the roof. This construction dramatically reduces the balanced cable's capacitance compared to coaxial cable, resulting in lower losses at identical frequencies and cable stretches.

On the down side, it's difficult to get a good quality signal from balanced antenna cable because any twists or any nearby metals will alter the cable impedance which causes interference like we talked about earlier. A modern solution is to use a 4:1 step-down balun to convert the balanced antenna connections ("bal") to unbalanced ones ("un") so that regular 75-ohm coaxial cable can be used for the downfeed. The balun works as a transformer to impedance-match and isolate the antenna's balanced outputs from the unbalanced coaxial cable. Then the receiver can ground the coax shield to protect the signal wire from interference. Building a VHF FM radio antenna is a great way to experiment with balanced cable now that terrestrial TV broadcasts are less common. **Figure 3** shows a commercial product [2], which employs a section of the 300-ohm cable as the dipole antenna. And then **Figure 4** shows an unexpected application of "ladderline" cable in a 1960s oscilloscope from Jan's *Retronics* collection.

(150704)

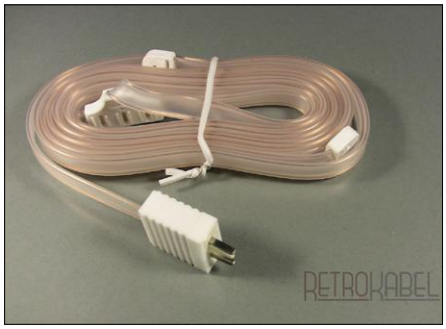


Figure 3. An in-room folded dipole antenna for VHF FM reception. It's made from 300-Ω "ladderfeed" balanced cable.

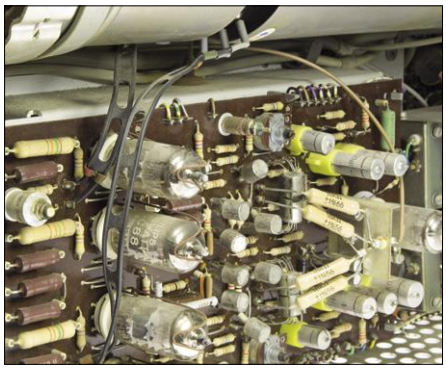


Figure 4. Balanced antenna cable used in a Philips PM3110 oscilloscope to drive the CRT plates from the Y amp outputs. These short stretches carry 10 MHz 360 V_{Y1-Y2} max. at a stable capacitance.

Web Links

[1] www.radio-electronics.com/info/antennas/balanced_feeder/balanced_feeder.php

[2] www.retrokabel.de

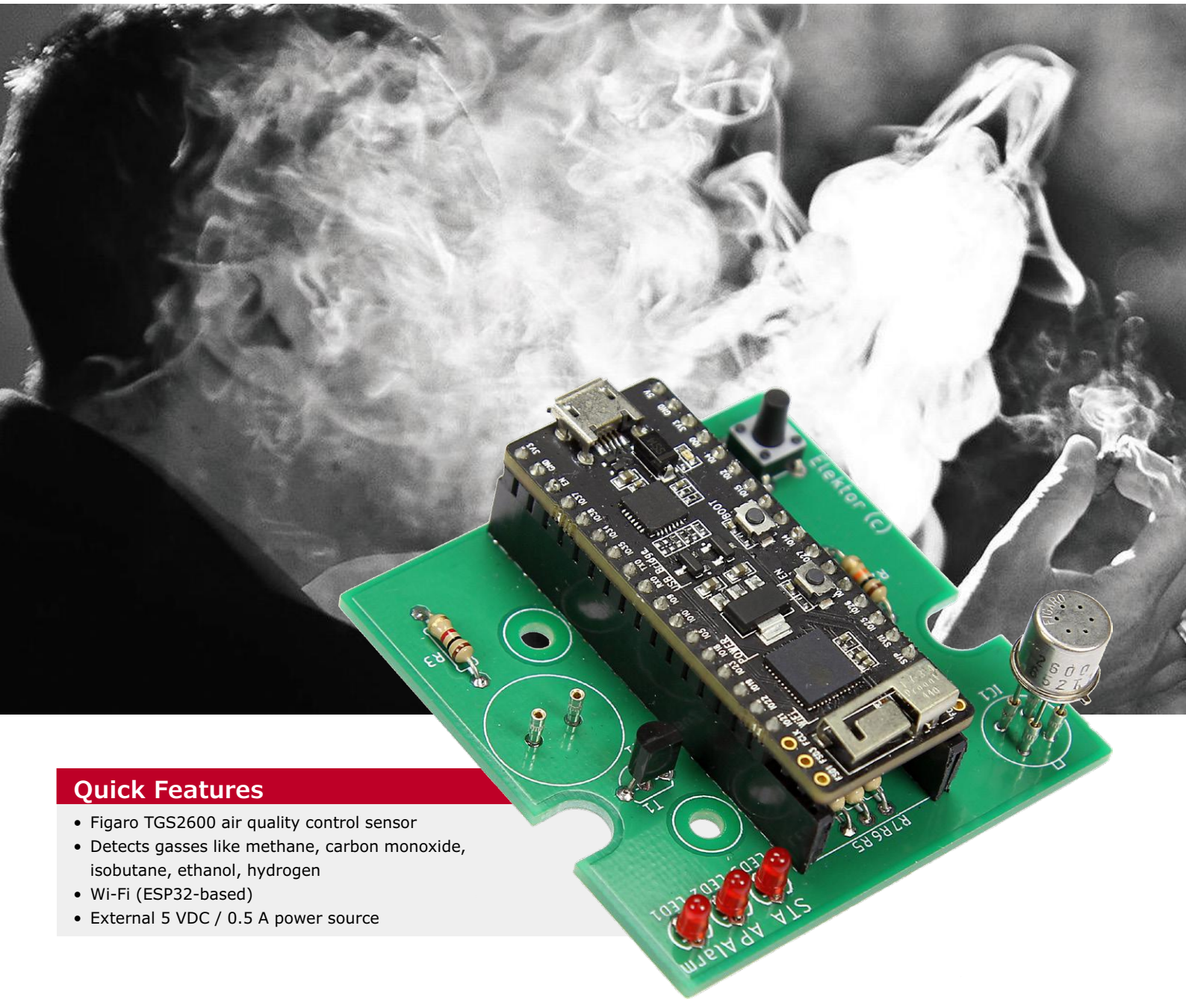
eLABS PROJECT

Air Pollution Monitor

Any smokers hiding in the toilet?

By **Bera Somnath** (India) and **Roy Aarts** (Elektor Labs)

The quality of the air we breathe should be of deep concern to all of us. We all know that airborne dust particles keep asthma sufferers awake at night, but the levels of certain gasses are equally important. This project measures the level of air contaminant gasses, and sounds an alarm when their concentration exceeds a certain level, for instance when someone lights a cigarette…



Quick Features

- Figaro TGS2600 air quality control sensor
- Detects gasses like methane, carbon monoxide, isobutane, ethanol, hydrogen
- Wi-Fi (ESP32-based)
- External 5 VDC / 0.5 A power source

The sensor used in this project, the type TGS2600 by Japanese gas sensor specialist Figaro, responds to low concentrations of various gases including methane, carbon monoxide, ethanol, hydrogen, and isobutane. The latter is often used as a propellant in aerosol cans. Note that the sensor does not actually distinguish between these gases — it only detects their effect on clean air. Also, the TGS2600 is not a gas sensor *per se* but



Figure 1: The Figaro type TGS2600 sensor detects gasses like methane, carbon monoxide, ethanol, hydrogen, and isobutane.

rather a gas-in-air sensor, meaning it will not work properly if the oxygen level is too low. Because the TGS2600 reacts to changes in air quality, it excels at detecting cigarette smoke, for example.
Even though the TGS2600 detects all sorts of gases, there are some it's not terribly keen on sniffing. Silicone (not: silicon) vapours and highly corrosive environments should be avoided, as are water and especially sea water (but then, hardly anything will prove resistant to seawater. And cat pee, but that is another story).

About the sensor, ...

The TGS2600, housed in a small metal can (**Figure 1**), is based on a resistive element made from tin dioxide (SnO_2). To enable it to react to changes in gas concentrations it requires heating, and fortunately a suitable heater is built into the device. Due to the heating, the sensor needs a few seconds to get ready after power on. When the sensor is powered for the first time or after an idle period of several weeks or more, the startup time is a good 15 to 20 minutes.
The resistive element (R_{GAS}; or R_S) inside the sensor is simply incorporated as the 'high' part of a resistive voltage divider. As its resistance is controlled by the composition of the ambient air, the resulting output signal is the voltage across the voltage divider's 'low' part, R_L, to be connected externally. The sensor's resistance in clean air is called R_O and this is the reference value. Consequently R_S is inversely proportional to air pollution. Ambient temperature and relative humidity levels affect the device's sensitivity to a degree, but as long as the sensor is used within the range of 10 to 50 °C, sensor compensation is not necessary.

the air pollution monitor circuit, ...

Using the TGS2600 in a circuit is easy: a microcontroller with an integrated analogue-to-digital converter (ADC) is all you need (**Figure 2**). Since we wanted to make the data available on a network, we used an ESP32-based microcontroller module for this. Actually, we made two versions of this project: one on a piece

PROJECT DECODER

- ESP32
- air quality
- smoke detector

- ➡ entry level
- intermediate level
- expert level

2 hours approx.

Arduino IDE, soldering iron

£25 / €30 / $35 approx.

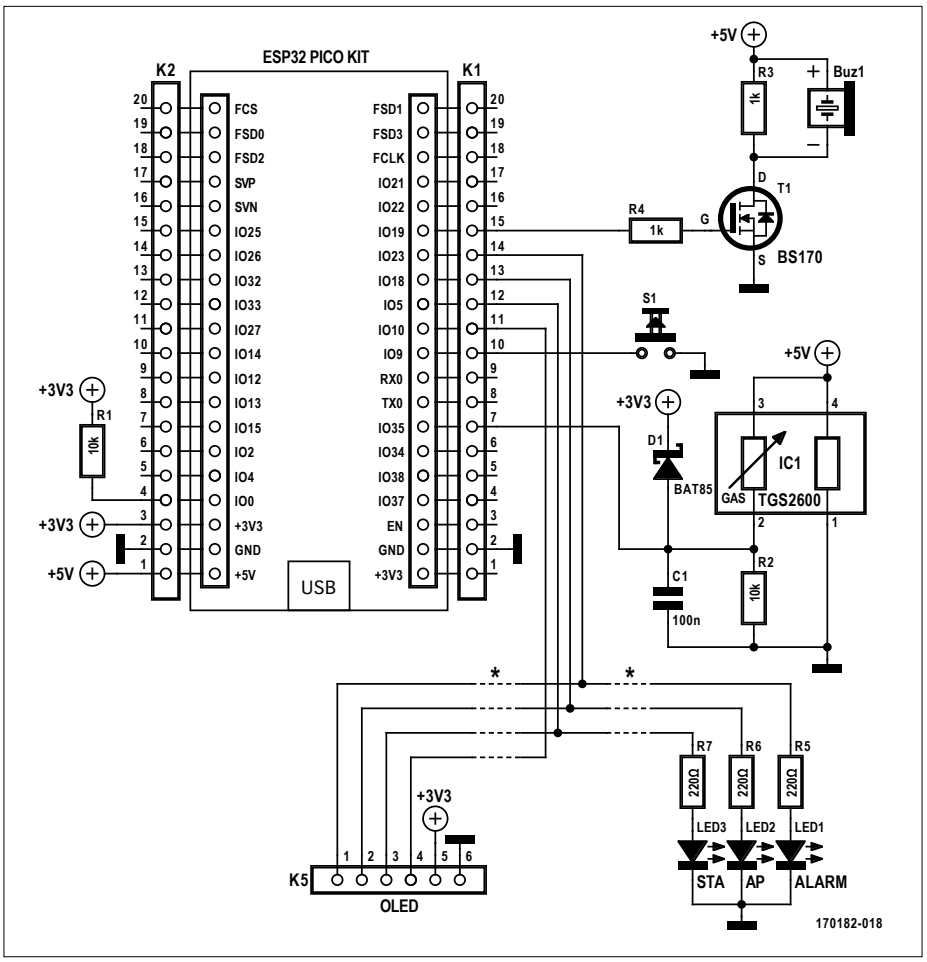


Figure 2: Because all the hard work is done by the TGS2600 really, the schematic of the air quality detector can remain simple.

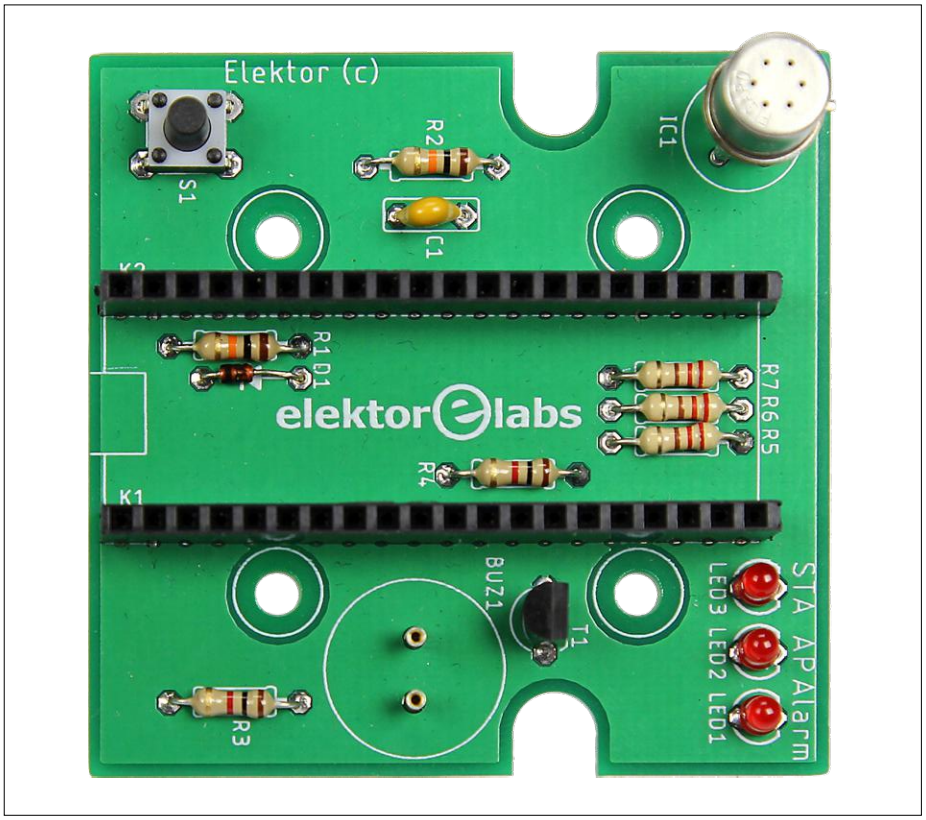


Figure 3: Some parts are hidden underneath the ESP32 module. The TGS2600 sensor is the tin can in the top right-hand corner of the board.

of prototyping board and a another on a printed circuit board (PCB). The first version consists of an ESP32 Pico Kit module, a 0.96" OLED display, a pushbutton, a buzzer and, of course, the sensor. The PCB version is almost identical except for the OLED display as it uses a webpage to visualise the data instead. Three LEDs were added to show the status of the wireless network connection (**Figure 3**). The PCB version developed in Elektor Labs was made to fit in a cheap Hammond type 1593k enclosure.

... and the software

Basically, all the software must do is measure the voltage across R2, which iequals R_L mentioned above. With the value of R2 known (10 kΩ as per the schematic) the value of R_s can then be calculated. Next, once R_o is known, the ratio R_s/R_o a.k.a. the air pollution can be computed. The weak spot here is R_o, because its value is unknown. To get it, a calibration has to be done in a reference environment that's supposed to be 'clean air'.

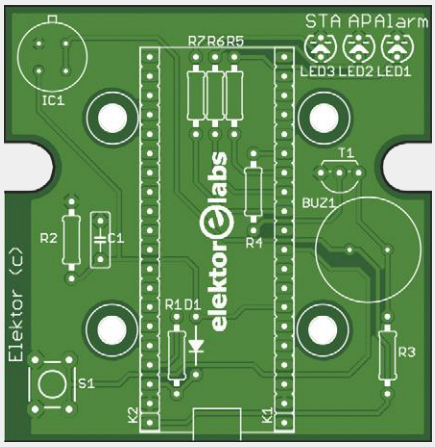
Weblinks

[1] U8g2lib graphics library: https://github.com/olikraus/u8g2

[2] ESP32 Arduino core: https://github.com/espressif/arduino-esp32

[3] SPIFFS upload tool: https://github.com/me-no-dev/arduino-esp32fs-plugin

COMPONENT LIST



Resistors

R1,R2 = 10kΩ
R3,R4 = 1kΩ
R5,R6,R7 = 220Ω

Capacitors

C1 = 100nF

Semiconductors

D1 = BAT85
IC1 = TGS2600
LED1,LED2,LED3 = red, 3mm diam.
T1 = BS170

Miscellaneous

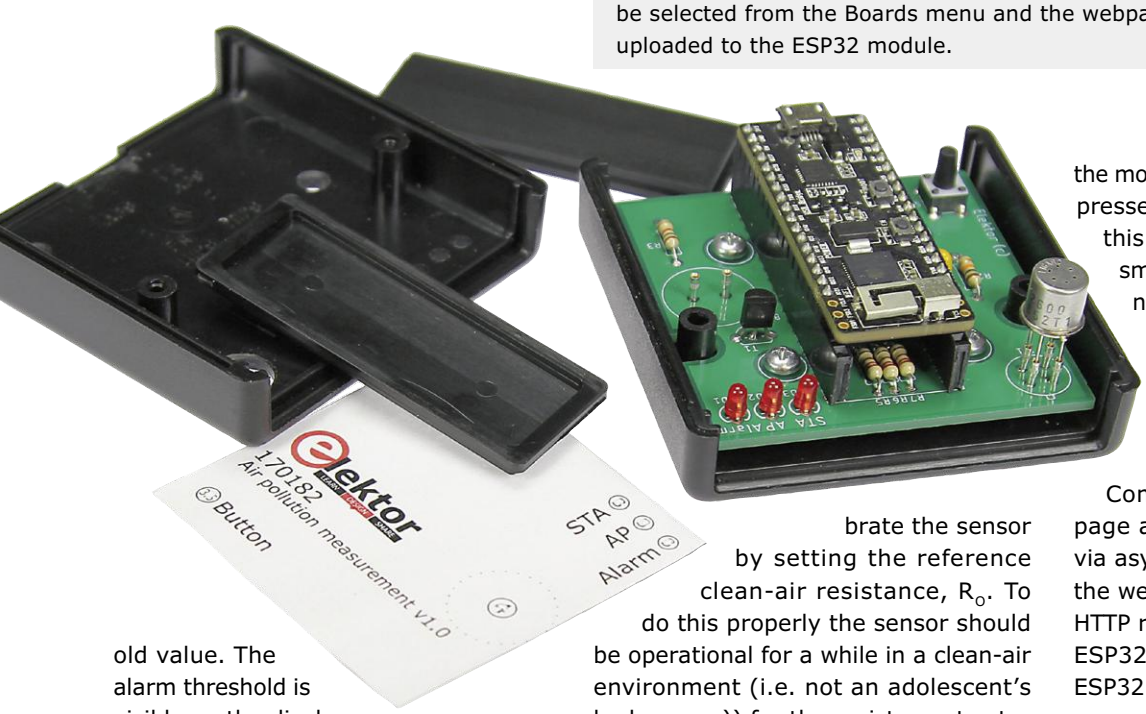
BUZ1 = PS1420P02CT
K1,K2 = 17-way pinheader socket, 0.1" pitch (for ESP32 Pico Kit)
S1 = tactile-feedback switch, PCB mount, 6x6 mm
PCB # 170182-1 (Elektor Store)
ESP32 Pico Kit (Elektor Store)

ThingSpeak

Air quality measurements are eligible for uploading to the open IoT platform called ThingSpeak. To do so, create a channel with one data field. This will result in a Write API key that has to be entered on the ESP32's configuration webpage. When set properly the measurements are uploaded every time the measurement interval has passed. Also, the value sent is the average value calculated over the interval.

To send the data to ThingSpeak, an HTTP request containing the sensor value is first sent to the website. ThingSpeak then responds with the number of entries of the channel, or with zero (0) if the request failed. One of the reasons for a request to fail is when the API key is entered incorrectly, or when requests are sent too fast in succession (ThingSpeak limits the upload interval to 15 seconds for non-paying, freeloading users). The minimum upload interval of the air pollution meter is one minute.

The calculated air pollution value is either shown in a line chart on the tiny OLED display, or uploaded to the popular ThingSpeak Internet of things (IoT) platform, where the measurements can be visualised as a line chart over a longer period of time. The OLED display only shows the last 20 measurements.
The measurements are also compared with a threshold value. When the air pollution drops below this threshold, the buzzer on the board will sound. According to the sensor's datasheet, an R_S/R_0 ratio of 0.85 is a reasonable alarm threshold value. The alarm threshold is visible on the display as a dotted horizontal line. If the alarm is on, pressing the pushbutton will stop it. Pressing the pushbutton will at the same time bring up some information about the current Wi-Fi connection together with the alarm threshold value. It will also tell you if ThingSpeak is enabled or not.



Configuration and calibration

We could have left it at that, but then where would the fun be? So, to positively complicate matters, slightly, a webserver was added. Doing so allows configuring the air pollution monitor by means of a convenient webpage, hosted by the ESP32 itself. On this page, the alarm threshold can be set as well as the ThingSpeak settings — i.e. the API key, required to upload the data and the time between two data uploads, see the **ThingSpeak text panel**. The Wi-Fi network to be used by your device can be configured here too. Most importantly, however, this page is where you calibrate the sensor by setting the reference clean-air resistance, R_0. To do this properly the sensor should be operational for a while in a clean-air environment (i.e. not an adolescent's bedroom :-)) for the resistance to stabilise. The sensor's datasheet mentions a period of seven (!) days. Press the 'Calibrate sensor' button when this period has expired.

Access point mode

First-time setup of the device requires the ESP32 to be in access point (AP) mode. Activate this mode by powering the module while keeping the pushbutton pressed; LED2 'AP' should light. When this is the case connect a computer, smartphone or tablet to the 'ESP32' network and navigate to IP address 192.168.4.1 to bring up the air pollution meter's configuration webpage. Now you can set up the Wi-Fi network and the other parameters mentioned above.
Communication between the webpage and the ESP32 module happens via asynchronous XMLHttpRequests on the webpage. When the page loads, an HTTP request is formed and sent to the ESP32. On receiving this request, the ESP32 sends a response containing the current configuration back to the webpage. Pressing a Submit button on the webpage will result in a similar transaction, but now containing the values entered by the user.

170182-01

One size fits all

Both versions of this project — breadboard with, or PCB without OLED display — use the same Arduino sketch. To differentiate between the versions a preprocessor definition is used. This is the first line in the code which reads `#define USEDISPLAY`. Remove or comment it out for the PCB version.
An additional library called U8g2lib (at: https://github.com/olikraus/u8g2) has to be installed for display operation. It's used to draw the text and the line chart on the display.
To upload the software to the ESP32, the ESP32 Arduino core needs to be installed (at: https://github.com/espressif/arduino-esp32). After installing the core, the SPIFFS upload tool to upload the webpage to the file system of the ESP32 (at: https://github.com/me-no-dev/arduino-esp32fs-plugin) needs to be installed also. When everything is set up correctly, the ESP32 Pico Kit can be selected from the Boards menu and the webpage and the sketch can be uploaded to the ESP32 module.

@ WWW.ELEKTOR.COM

→ Air Pollution Monitor, bare PCB
www.elektor.com/air-pollution-monitor-170182-1

→ ESP32 Pico Kit
www.elektor.com/esp32-pico4

→ TGS2600 Air Contaminants Sensor
www.elektor.com/tgs2600-air-sensor

→ 0.96" 128x64 OLED Display, 6-pin
www.elektor.com/oled-display-for-arduino



I²C Data Bus Reader

Everything on the PC screen, via USB

By **Willem den Hollander** (Switzerland)

The Inter-Integrated Circuit bus — or I²C bus — is frequently used for communication between integrated circuits. It is often useful to know what data is being sent about this bus. The circuit described here reads the I²C data and sends it via the USB to a PC that then hexadecimal form on the screen displays. Optionally, the data can be filtered to select data with specific addresses.

The I²C bus consists of two signal lines: the SCL clock and the SDA data line. The data is transmitted in series. A message consists of two or more bytes, each followed by an acknowledge-bit. A start sequence indicates the beginning of the message, a stop sequence closes it. To transfer all information, we need to send twice as many bytes over USB as the data itself comprises. The clock frequency in the "standard mode" of the I²C bus is 100 kHz, resulting in a data rate of 20 kB per second. In fast mode this is 80 kB per second. In the application described here, one 64-byte packet per millisecond is sent via USB. This is more than enough for reading data on the I²C bus in standard mode, but not enough for fast mode. However, the I²C bus data reader can still be used to read data in fast mode if there are only occasional data present, which is often the case.

The hardware

Figure 1 shows the diagram of the I2C data bus reader. Here we see two microcontrollers. We use the PIC16F1455 (U2) to read out the I2C data and send it to the PC via USB. The I²C module of this controller is configured as a slave receiver to collect all data at any address. However, this only happens if

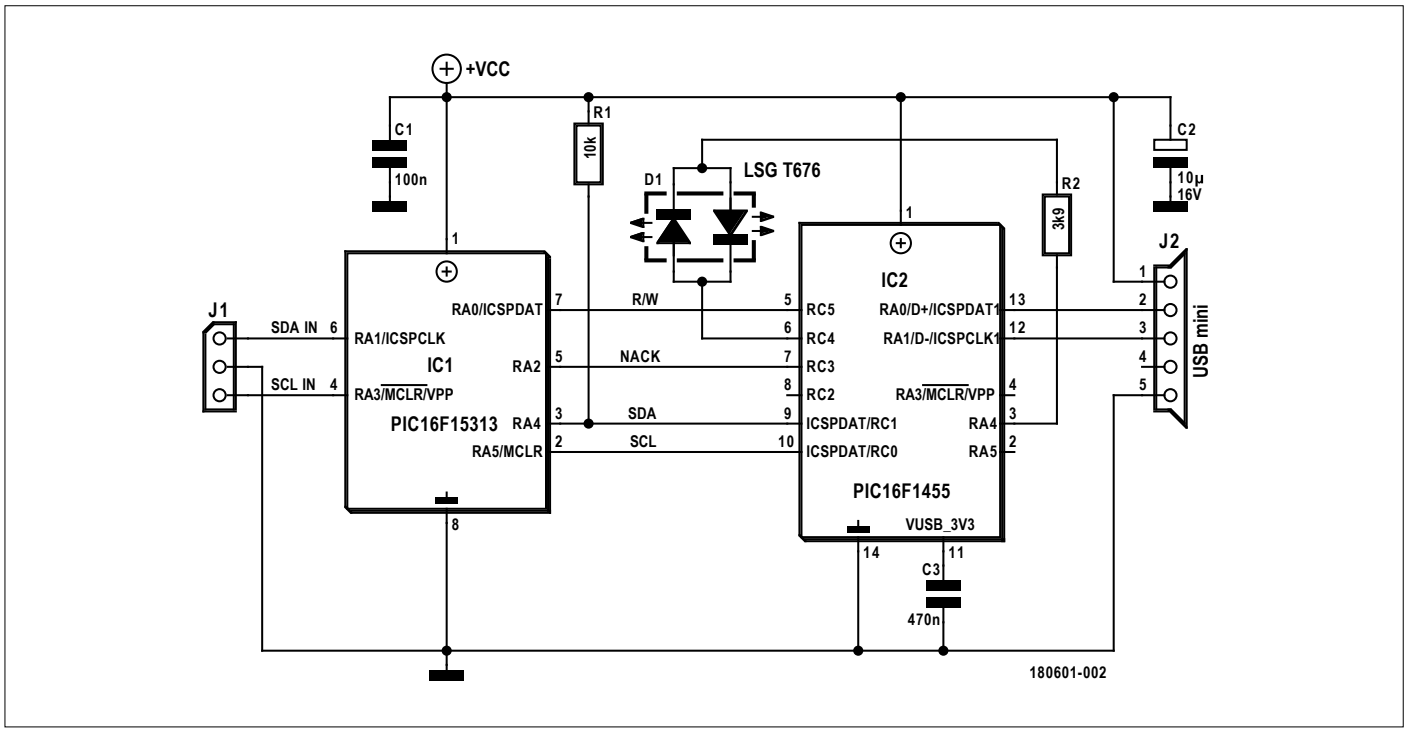


Figure 1. Full circuit diagram of the I²C data bus reader.

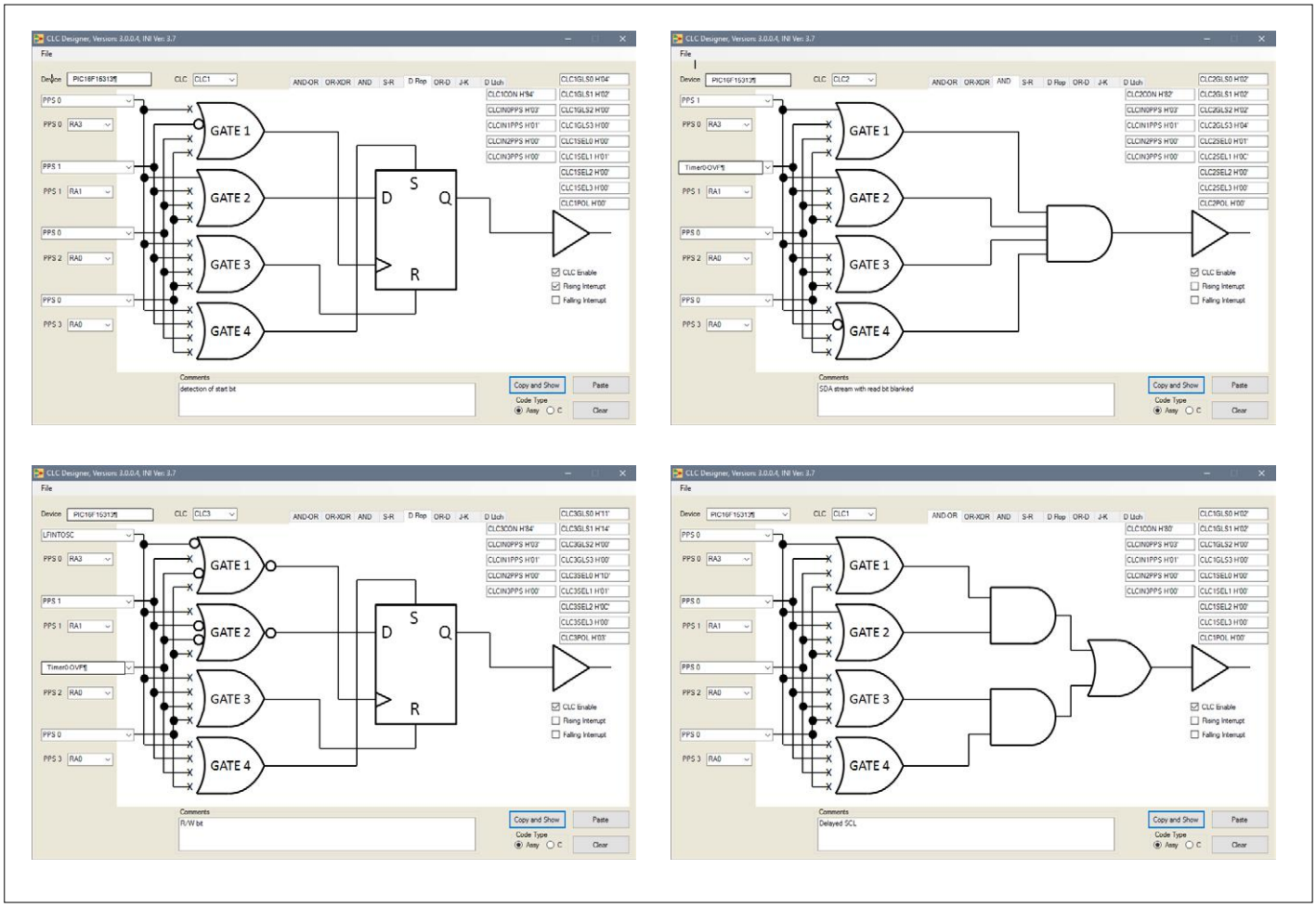


Figure 2. The CLCs in U1 are configured this way for the I²C bus data reader.

the write bit of the incoming data is present. When the read bit is set, all data is ignored. In addition, the confirmation information cannot be read, as the I2C module provides it itself.
For these reasons we have used a preprocessor, a PIC16F15313 (U1), which converts each 'read' bit into a 'write' bit. The read bits and the acknowledgement are removed from the data stream and passed separately to the PIC16F1455. The latter incorporates this information into the received I2C data and sends it to the PC via USB.
U1 is a PIC equipped with four configurable logic cells (CLC). **Figure 2** shows how they are configured for this project.
The PIC16F1455 microcontroller includes the full USB interface. The internal oscillator is accurate enough and an external quartz crystal is not necessary. The microcontroller stores all received bytes at the correct memory locations and after receiving sufficient data, they are sent to the PC.
The firmware for both processors is written in assembler. Critical timing is thus accurately under control. The configuration of the USB module is also written in assembly language. This part of the firmware was originally supplied by Microchip and after some adjustments and debugging it has been in use for about twelve years now.
The circuit requires only little power. It draws its power from the PC's USB connection.
I built the circuit with SMD parts on a small double-sided PCB with a size of 14 by 33 mm (see **Figure 3**).

Firmware and software

Timer0 of U1 generates a one-bit wide window that empties the read-bit, see also **Figure 4**. In **Listing 1** we show how CLC2 sets the read-bit to '0'. The modified SDA signal, the output of CLC2 - see top right in Figure 2 - is connected to output RA4.

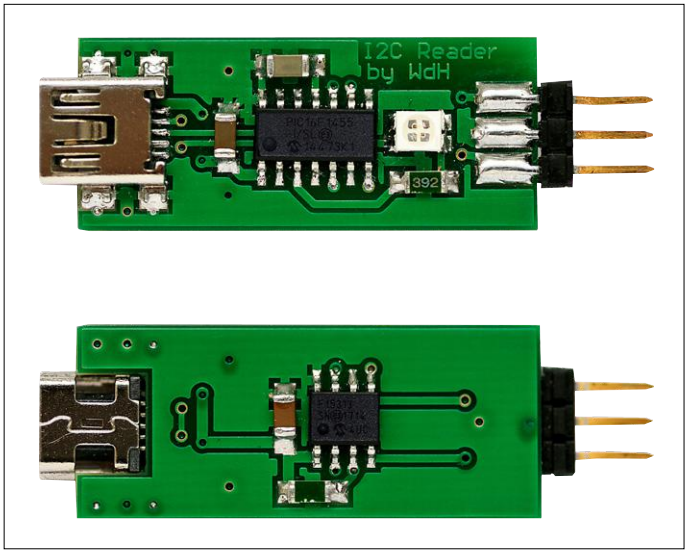


Figure 3. The top and bottom side of the author's SMD-based PCB.

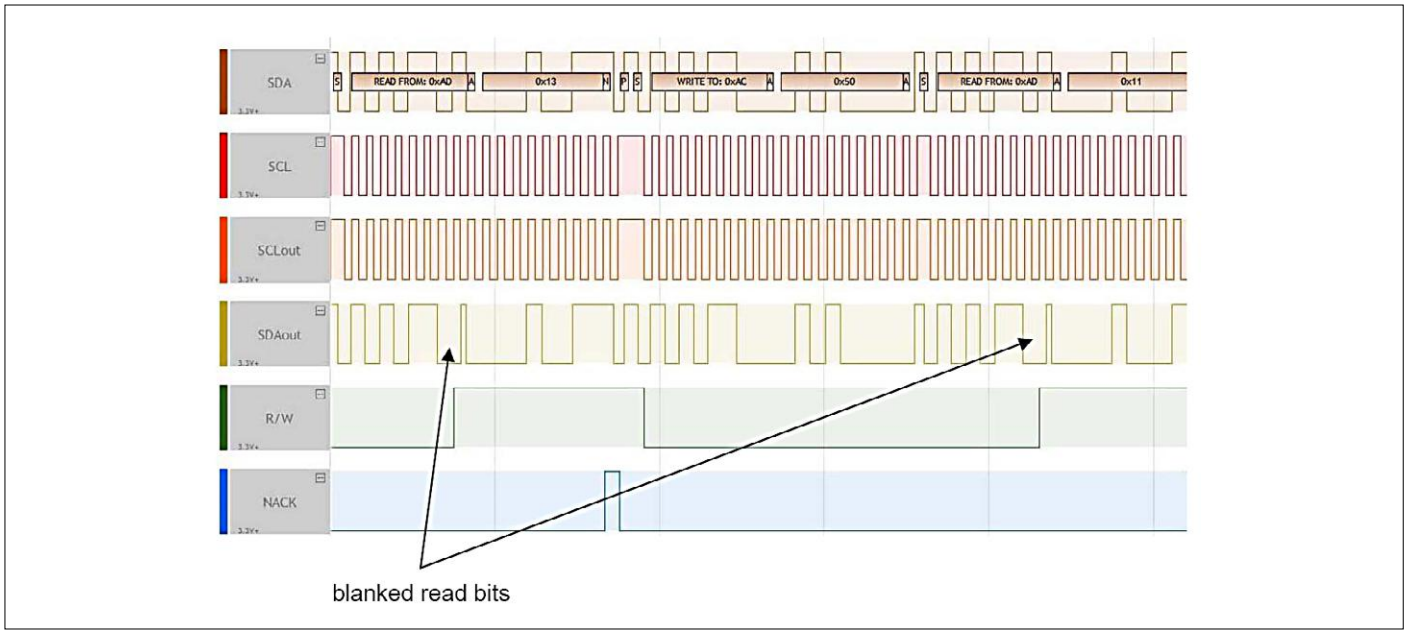


Figure 4. The signals around U1 indicating where the read-bit is removed.

RA4 is an open-drain output with an external pull-up resistance. This is a requirement because U2 always sends back a confirmation signal (ACK) after receiving one byte of the preprocessor. The input of U2 becomes an output during one bit. The read information is available at the output of CLC3 (see bottom left in Figure 2) and is routed to output RA0.
Because the SDA signal is slightly delayed by the CLC, to keep the timing correct, the SCL signal must also be delayed. We do that here with CLC4. A small program controls the timer and reads the confirmation bit for RA2.
Figure 5 shows the graphical user interface (GUI) of the corresponding PC software. The status line below indicates whether the I2C data bus reader is connected and recognized or not. The "Start Acquisition" button starts reading the I²C data. The LED of the reader lights green. The start button now becomes the stop function. The I²C bus data is displayed in the window, where according to the standard S stands for Start, R for Read, W for Write, A for Acknowledge, N for No Acknowledge and P for stoP. The data bytes are displayed in hexadecimal format. The software permits to filter in hexadecimal or binary form on the first two bytes. When these bytes are filled, only data starting with these bytes will be displayed. If "Post Acquisition" is checked, the data acquisition only starts after receipt of the specified bytes. When filtering with the binary form it

Listing 1.

This source code extract shows the configuration of CLC2 (Figure 2b). Here the read-bit from the SDA data stream is set to zero. Timer0 is started when the start bit is detected.

```
banksel   CLC2GLS0 ; TMR0 ANDed with SDA clears bit 8
movlw     2
movwf     CLC2GLS0
movwf     CLC2GLS1
movwf     CLC2GLS2
movlw     4
movwf     CLC2GLS3
movlw     1        ; SDA input signal
movwf     CLC2SEL0
movlw     0ch      ; TMR0 overflow
movwf     CLC2SEL1
clrf      CLC2SEL2
clrf      CLC2SEL3
clrf      CLC2POL
movlw     82h
movwf     CLC2CON
```

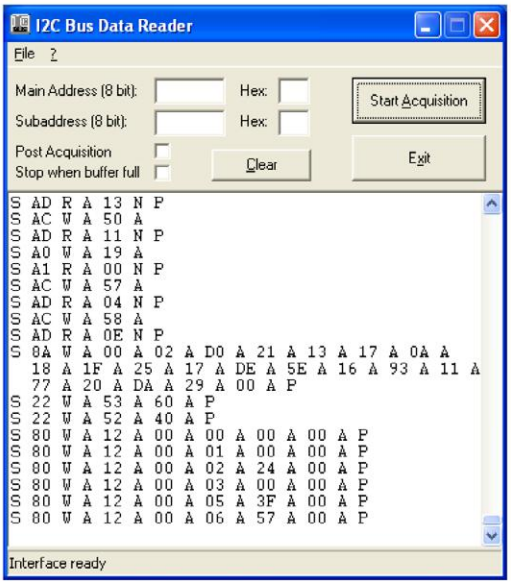


Figure 5. The user interface is very clear and equipped with a few handy features.

is also possible to work with don't cares, at the place of the don't care you enter an 'X'.
The size of the buffer is about 40 kbytes. If the buffer threatens to fill up, part of the data at the beginning of the buffer is deleted. Checking "Stop when buffer full" prevents this by stopping the data acquisition at that moment. The green LED goes out as soon as the acquisition stops.
If too much data is present, the data acquisition stops and the LED lights red. The software is then automatically returned to the starting situation. The data in the log window will remain until deleted or until the buffer is full and a part is deleted. Via the menu "File" you can save or print all or part of the data in the window on the hard disk.
I have noticed that the continuous display of the data is a heavy load for the Windows operating system. It is therefore recommended not to run other programs on the PC at the same time.

In use

Before you can use the I²C data bus reader, you must install the USB driver. When you connect the reader to the PC via the USB cable, the PC sees a new device. In the download with this article [1] a .inf file is included in the driver directory. Select this file when the PC asks for a driver and the driver will be installed automatically. To install the user interface, double-click on the Setup.exe program, which is also available in the download at [1], and follow the on-screen instructions.

Conclusion

The special feature of the described I²C data bus reader is its ability to read data as long as it is needed, i.e. for an unlimited time. This allows us to detect errors that only rarely occur. By rewriting the PC software, it is possible to store received data on the hard disk, which we can then analyse later.

180601-04

Web link

[[1] Software and drivers: www.elektormagazine.com/180601-01

@ WWW.ELEKTOR.COM

→ Book: Elementary Course BASCOM-AVR
www.elektor.com/elementary-course-bascom-avr

→ Book: Make: AVR Programming
www.elektor.com/make-avr-programming

Advertisement



Interference Sources in Automotive Applications



And how to get rid of them

By **Tam Hanna** (Slovakia)

It is easy for anyone whose background is in the world of industrial control to fall into one of the traps for the unwary when turning their hand to designing a circuit for use in a vehicle. This article will look at the ways in which things can go pear-shaped, and at the risks in store for designers and their hardware.

Just a couple more words before we get down to the nitty-gritty: the author knows of a large number of low-volume designs that in some cases have been in working use for years despite implementing few if any of the protection measures we will be discussing. One of the reasons for this can be understood by thinking of an automotive system as comprising a network of filtering elements with an exponential decay characteristic as shown in **Figure 1**, with some circuits being in effect closer to the sources of trouble others: these circuits take the brunt of the interference, while others are left relatively unscathed. However, providing a design with the appropriate protection measures generally does not cost the earth and thus in any case makes economic sense.

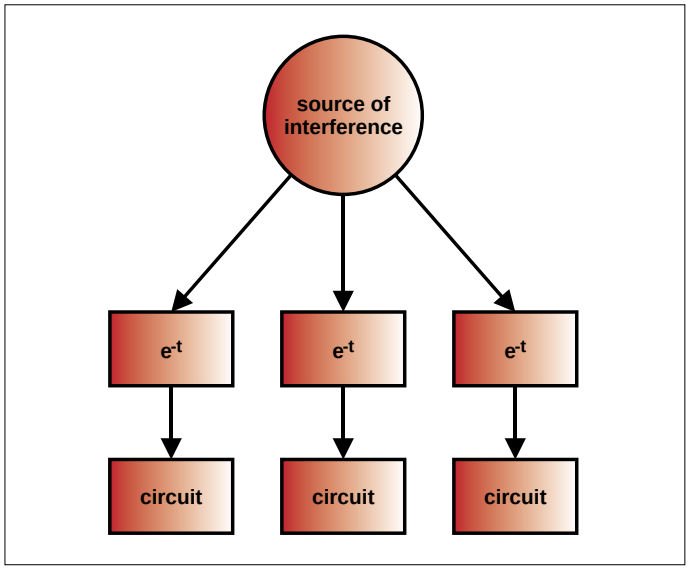


Figure 1. The circuits have different levels of exposure to the sources of interference.

The savage alternator

Let us start with the most pernicious of all types of interference, known as 'load dump' in the trade. Imagine a quad bike or similar vehicle being driven hell-for-leather along a dirt track. Unfortunately the integrity of the connection between the alternator and the battery is not all that could be wished for: perhaps the manufacturer has been trying to cut costs, the mechanic was asleep or the dreaded tin worm has taken its toll on the terminals. And so, as you might expect, when the tyres hit an especially rocky patch what happens is that the connection between alternator and battery is briefly interrupted. Now the alternator wants, and indeed has to, unload its excess output energy somewhere. Fortunately this is not that common an occurrence in practice.

At this point we encounter for the first time a problem that we will frequently face in automotive applications: competing manufacturers specify more or less whatever they want to, following the approaches used by different standardization organizations. It's a jungle out there!

Back to load dump. There are two relevant ISO specifications: the first is ISO standard 7637-2, which can be downloaded at [1]. That was in force until 2010, but continued to be used subsequently by some car makers, including Ford. The second is the version currently in force, ISO 16750-2, which is not available to view free of charge. The semiconductor manufacturer Vishay has prepared a brief overview at [2], summarizing the requirements of the current standard. At [3] Texas Instruments reveals the form of the test impulse, as shown in **Figure 2**.

The bottom line is the observation that in a 12 V system we must be able to withstand a pulse of up to 101 V with a duration of 400 ms, with a source impedance of 0.5 Ω minimum and 4 Ω maximum. For 24 V systems the maximum duration is 350 ms, and the voltage is in the range 151 V to 202 V with a source impedance from 1 Ω to 8 Ω.

From these we can determine, with some not-too-taxing arithmetic, that a 12 V system has to be able to withstand an instantaneous power of 20.4 kW, and a 24 V system has to withstand 40.8 kW. Particularly tricky to deal with in this context is the very short rise time of just 10 ms or so. Any circuits we design need to react very quickly, which rules out from the start the use of components such as relays.

Just as tricky is the requirement laid down in the new standard that the circuit must be able to withstand a series of ten such impulses within the space of ten minutes. The older standard only required tolerance to a single pulse. However, the requirement of the new standard does make sense, as a vehicle with corroded battery contacts can easily produce a series of load dump impulses.

A classical approach to providing protection against such pulses, beyond just adding a huge smoothing capacitor with a series resistor, is to design in a protection switch that disconnects the circuit from the vehicle supply when emergency strikes (**Figure 3**).

Fortunately the semiconductor industry has fully realized how onerous complying with the standards can be, and offers a range of components designed to deal with transients. The company Littelfuse has been particularly successful in this department. Their TVS (transient voltage suppression) diodes, described briefly at [4], are components that act as a kind of crowbar, limiting the voltage when an exceptional transient occurs. STMicroelectronics have also been busy: their range of suppression diodes, going under the name 'Transil', is grouped according to the ISO test pulses, making it easier for designers to choose the right component (**Figure 4**). Vishay's ESD protection diodes go under the name 'TransZorb'.

Be aware that TVS diodes are almost always available in unidirectional and bidirectional variants. Unidirectional TVS diodes behave much like conventional Zener diodes, with a pulse of the 'wrong' polarity being clamped immediately. Bidirectional TVS diodes have a minimum threshold voltage in each direction before they start to conduct.

The author is a fan of the LM2576HV switching voltage regulator from Texas Instruments' 'simple switcher' range, which is somewhat tolerant of high voltages (although its cold-start voltage performance leaves something to be desired). Using this reduces the burden on the diodes as they can be arranged to start acting at 35 V: the power they are required to dissipate is then considerably lower.

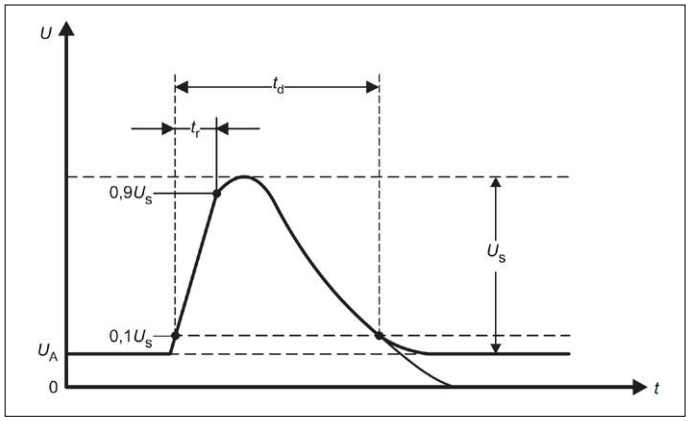


Figure 2. It's no fun dealing with a spike like this (source: Texas Instruments [3]).

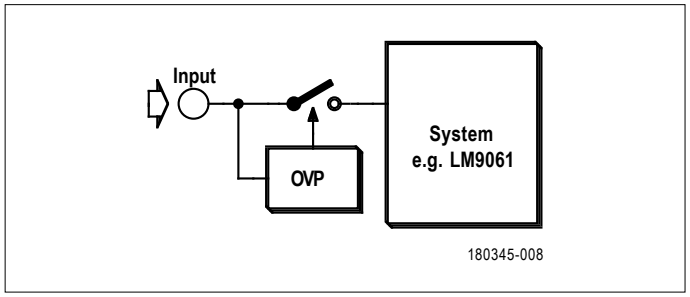


Figure 3. Sometimes it pays to be cowardly!

TVS diodes are of course not the only weapon that can be used in the battle against voltage spikes. For example, varistors (voltage-dependent resistors, or VDRs) are bidirectional devices that in principle exhibit characteristics similar to diodes. One way they differ from TVSs is in their considerably higher energy absorption capacity; on the other hand they switch more slowly and degrade over the course of time (even though

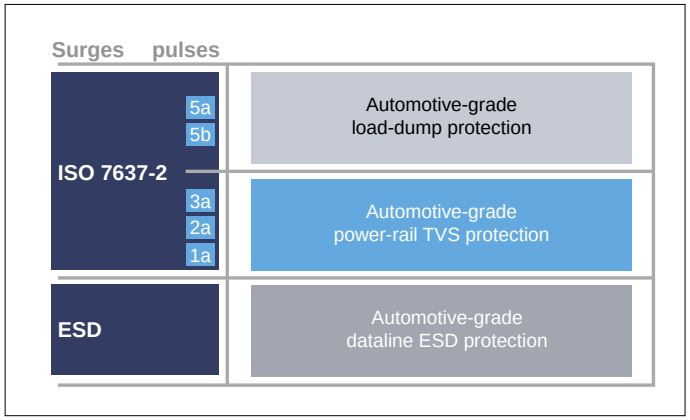


Figure 4. STMicroelectronics categorizes its Transil family of devices according to the level of ESD protection offered (source: STMicroelectronics [5]).

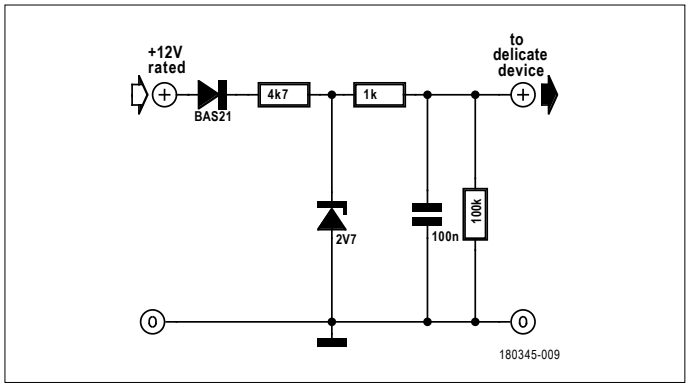


Figure 5. Unoriginal, but simple: a voltage limiter using a Zener diode.

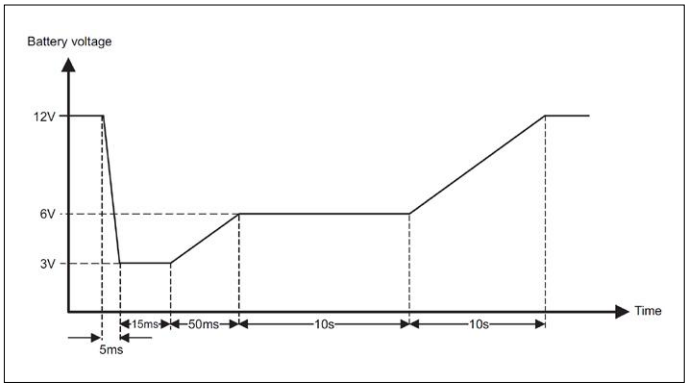


Figure 6. The unpleasant effect of cold cranking (source: Texas Instruments).

Vishay claim at [6] a 'negligible wearout within safe operating area'). Another potential problem in this regard is the tendency of metal-oxide varistors (MOVs) to suffer 'punch-though' failure: the author has seen several systems where aging MOVs have shorted out one or more power rails.
In practice one often finds a MOV paired with a TVS diode with a rather higher rated breakdown voltage. The idea behind this is to combine the benefit of the fast switching characteristic of the TVS diode with the higher energy capacity of the MOV. But note that the IEC 60950-1 standard prescribes the use of MOVs for over-voltage protection and forbids the use of TVS diodes!

Vicious voltages

As well as protecting a circuit from its main power supply, it is important also to look at adequately protecting sensitive components, and in particular their inputs, using voltage limiters. A standard and highly effective, although not exactly elegant, configuration is the Zener diode circuit shown in **Figure 5**. When using this kind of protection circuit it is necessary to ensure that the components are correctly dimensioned: the resistor and the Zener diode must be able to absorb the energy in any applied voltage spike without overheating: the resistor must be able to limit the current through the Zener diode to a safe level without itself being destroyed.
In theory in a 12 V system we might reckon with a maximum continuous supply voltage of at most 14 V or perhaps 15 V. Unfortunately in practice things are not so simple, as the wetware behind the wheel cannot always be relied upon. One example is when a vehicle's battery is left to run completely flat (whether through negligence or stupidity) and then an attempt is made to jump start it from another vehicle. It is surprising how often people will try to jump start a 12 V car from a truck with a system voltage of 24 V. Also, in the English-speaking world at least, it is not uncommon to wire the two batteries in series when jump starting. At any rate, it is worth just making sure that the protection circuits in your design will not see a continuous 24 V supply as a voltage spike that should be clamped or shorted out. For a switching regulator an input voltage of 24 V for a brief period should not present a grave problem, as long as the circuit is reasonably carefully designed.

Don't trust the voltage!

Getting voltage spikes out of the way is only half the battle: it is not just impulses that cause interference. A further source of aggravation is voltage drop-outs, which can arise for a variety of reasons. One is the sudden application of a heavy load to the system. Starting an internal combustion engine places extreme strain on the battery, especially at low temperature ('cold cranking') when the performance of the battery is in any case degraded. The behavior of the system voltage can be as extreme as that shown in **Figure 6**.
At first sight the designer might be inclined to ignore the cold crank phenomenon. A voltage regulator (such as the LM2576 mentioned above) might easily be able to operate from voltages as low as 8 V. According to the ISO standard, however, systems must be able to cope with voltage dips down to 3 V, depending on their criticality.

There are several ways to attack this problem. The first, of course, is to fit as large a capacitor as possible across the input, which will also have the effect of helping to smooth out rapid rises in voltage. The second possibility is to arrange things so that the system shuts down when the voltage falls below a certain level during cold cranking, and the restarts when the voltage becomes high enough once more. This approach is not always recommended, in particular in the case of controllers running complex operating systems that can take a long time to start up, as this can lead to further significant problems.
Another, rather more interesting, solution is to use buck-boost technology. Here we use a switching regulator which is capable of either stepping down or stepping up its input voltage. Application Note 728 from Texas Instruments [7] gives more information. If the regulator can operate with a 3 V input voltage, then the problem of cold cranking is eliminated entirely.

Fighting negative voltages

As a designer you might think it amusing to imagine a vehicle owner or mechanic accidentally connecting the battery the wrong way around, but don't laugh: it really does happen! Fortunately protecting against this kind of stupidity is not particularly onerous, and it does not take a genius to design the protection circuit shown in **Figure 7** using a single diode. In practice however, the demands on this protection diode are far from light. This is because a vehicle is not just a collection of exponential-decay elements connected in series, but instead is a complex agglomeration of different inductive loads that generate all sorts of spikes when switched on and off.
As well as positive-going spikes, against which we install TVS

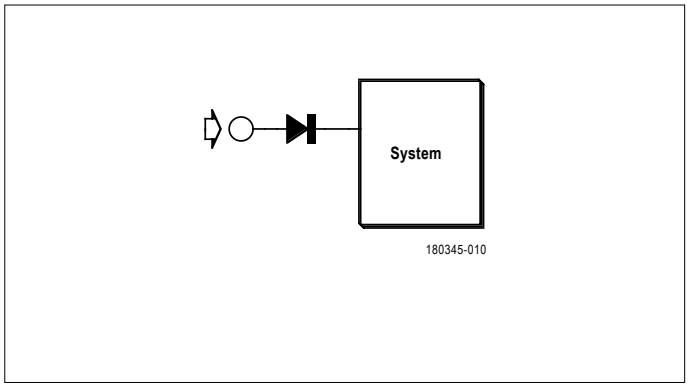


Figure 7. Sometimes the simplest solutions are the best.

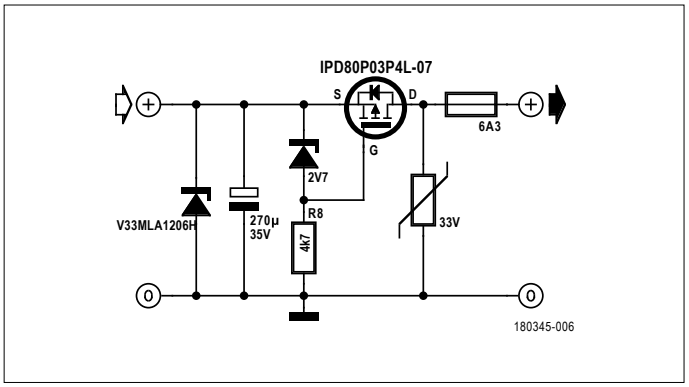


Figure 8. A somewhat unusual application for a MOSFET.

diodes, we can also expect negative-going spikes at the inputs to our circuits, sometimes going as low as –200 V! This means that our choice of diode is rather critical. For the main supply the author has had good experiences with the Diodes Incorporated S5BC-13-F, while the BAS20 comes in very handy for protecting voltage attenuators and similar circuits.
Standard diodes have long ago lost their monopoly in this kind of application scenario. Their relatively high forward voltage drop means that at high currents it is very desirable to replace them with more efficient components. Here designers have found a new field of applications for both P-channel and N-channel field effect transistors. **Figure 8** shows schematically how one kind of protection circuit can be constructed.

What more is there to say?

This article has confined itself to 12 V and 24 V systems. As vehicle electronics becomes more and more complicated it is becoming increasingly common to see vehicles with 48 V system supplies. The specifications for these systems are not yet completely finalized, and so if in doubt consult the manufacturer of the vehicle.
Automotive is far from being a niche market, and so it will come as no surprise to anyone that there is a huge amount of literature on the subject of automotive electronics. The ISO documents mentioned in the introduction to this article are certainly the standard references, although they are rather difficult to get hold of. In some countries it is possible to inspect standards for free in libraries or colleges, although this will often involve a long journey.
A handy alternative to obtaining full paper copies of the standards is offered by the American car manufacturer Ford at [8]. This is a specification document aimed at suppliers who wish to produce components for the company, but it holds interest also for other designers as the document is almost completely self-contained and makes it more or less unnecessary to read the original standards.

As we said in the introduction: there are thousands of systems that crassly flout some or even all of the rules that we have discussed, but which nevertheless function without problems. Protect your design with a TVS diode and a couple of other components, and the likelihood is that everything will work fine. ⏮

180345-02

Web links

[1] ISO 7637-2: www.smd.ru/upload/medialibrary/a3d/iso_7637_2.pdf
[2] Load dump protection: www.vishay.com/docs/49748/49748.pdf
[3] Load dump and cranking protection: www.ti.com/lit/an/snva681a/snva681a.pdf
[4] Littelfuse automotive TVS diodes: www.littelfuse.com/~/media/electronics/application_notes/littelfuse_tvs_diode_automotive_circuit_protection_using_automotive_tvs_diodes_application_note.pdf.pdf
[5] Automotive protection devices: www.st.com/content/st_com/en/products/protection-devices/automotive-protection-devices.html
[6] Transient suppressors, a competitive look: www.vishay.com/docs/88444/tvscomp.pdf
[7] Output voltage regulation during automotive cold crank: www.ti.com/lit/an/snva728/snva728.pdf
[8] Ford electromagnetic compatibility specification: www.fordemc.com/docs/download/FMC1278.pdf

Interference from LED Lamps

Do interfering LED lamps conform to EMC standards?

By **Johan Jongbloed** (The Netherlands)



Back in April 2018 we asked our readers to send us their suspect LED lamps so we could investigate how much interference they actually generated. This resulted in a decent number of different brands of lamp in the post. The EMC-EMF Commission from VERON (Association for Experimental Radio Research in The Netherlands) kindly offered to subject these lamps to an EMC pre-compliance test to see if the LED lamps conformed to the EN 55015 European standard.

First, we'll take a closer look at the EMC directive and standards. The EMC directive has been incorporated in the Electromagnetic Compatibility Regulations — the standards have not been included — and describes "emission" as follows in appendix I:

The electromagnetic disturbance generated does not exceed the level above which radio and telecommunications equipment or other equipment cannot operate as intended.

But how should this be interpreted? When, according to the directive, a radio no longer functions properly, meaning that it can no longer receive the required transmitters, it obviously suffered from interference. But should there be a minimum distance between the interference source and the radio?
The limits and measurement procedures were created back in the seventies, when the most common sources of interference were electrical motors in vacuum cleaners and coffee grinders, or the ignition in internal combustion engines. In those days we didn't have personal computers, mobile phones, switch-mode power supplies, LED lighting, or PV (solar panel) systems. It was generally assumed that the source of interference was with the neighbours, or in any case, it was not possible for you to turn off the source of interference. For this reason, a distance of 10 meters was chosen in the seventies for the measurement distance between a device and the source of interference.

Has there been a progressive insight?

Not really. Times have changed, but the basis for the standards seems to have been stuck in the seventies. Nowadays there are so many sources of interference under one roof that it's become almost impossible for radio amateurs to have a decent conversation via the ether, even with an outdoor aerial. About 70% of complaints about interference are caused by devices belonging to the complainant. These can of course be turned off, but that's certainly not the case for those of your neighbours! You also have to take into account that the sources of interference are no longer turned on for short periods these days, which used to be the case with vacuum cleaners and coffee grinders. These new sources of interference are often turned on permanently, or are simply not turned off.

Haven't the EMC standard committees done anything?

The EMC committees didn't predict that radio communications would suffer from as much interference as is the case now. It has happened very gradually, without anybody really noticing it. The first people to notice that something was amiss were predominantly radio amateurs. They often work at or close to the noise floor and they were the first to notice that something was changing in the frequency spectrum.
Another problem is that the EMC regulations assume that each product should satisfy the standards in isolation, whereas there are often several devices in operation at the same time, all generating interference. It's just like a busy cafe, where many people are talking at the same time, causing the background noise to increase significantly.
Perhaps it's time to look at a modified standard that takes the device density per m2 into account, for example, and for the product limits to be reduced by several dB for a single product.

The test: why EMC pre-compliance?

For equipment designers it is advantageous (particularly finan-

cially) to carry out a pre-compliance test during the development stage. A full compliance test or formal test is costs a lot (from € 1,000 to € 4,000). This way you can find and correct any EMC problems early on in the development of a product, before it needs to have a full compliance test, giving it a better chance of passing later.
An EMC pre-compliance test can tell you a lot about the emissions from an electrical device, and it's sufficient in this case to give a meaningful result regarding the emissions from the LED lamps.

Types of emission

During the tests we look at two types of emission: Conducted Emission (CE) and Radiated Emission (RE). The first is the electromagnetic energy that is generated inside a device and which finds a way out via the 230 VAC supply cable of the device. The electricity supply network in the home then acts as a long aerial, which can transmit signals. Radiated Emission (RE) is the electromagnetic energy that is generated inside the device, and which finds a way out via the air. This electromagnetic energy is often generated by devices that haven't been properly designed, or which have defects. Any devices that generate such emissions can cause problems in other devices that are in the vicinity.
The EN 55015 standard, which applies to the radiated energy, has an alternative method for the measurement of the Radiated Emission. With this method the emission measurement is carried out using a decoupling network. We also call this a Conducted Emission measurement in this report.

Results

In a CE measurement the measured spectrum is divided into two bands: from 9 kHz to 30 MHz and from 30 MHz to 300 MHz. Each frequency band has its own measurement setup. For the RE measurement an aerial is used at a distance of 3 m or 10 m from the (interference) source in the frequency band of 30 MHz to 300 MHz.
Since the interference can have an adverse effect before it exceeds the limits, we have sub-divided the interference level into four bands. To be clear, these aren't mentioned in any standard, they just give us a better insight. Instead of stating whether a lamp conforms to the norm or not, we use four bands to show how far the emissions are below the maximum permitted by the standard. **Table 1** shows the results of all the CE measurements.
We were somewhat surprised to find that the emissions of all the tested LEDs were within the CE measurement limit. Although it should be noted that for the CE measurements in the 9 kHz to 30 MHz band most of the LEDs came very close to the limit — just 0–5 dB below it. But over half the LED lamps had emissions that were more than 10 dB under the limit. In the CE test for the 30–300Mhz band, there were only a few LED lamps that came near the limit, but none of them exceeded it. In **Figure 1** we have the measurement for one of the two LED lamps that came above the limit in the RE measurement. The red line indicates the limit, the green line shows the emitted signal according to the RE measurement. In this instance the peak value exceeded the maximum permitted level by over 10 dB. However, this was the exception that proved the rule: there were only two lamps that exceeded the limit, and one of those was less than 5 dB above the limit. The other lamps were so well-behaved that they didn't seem to warrant a measurement. Some of them didn't even show an increase in the noise floor.

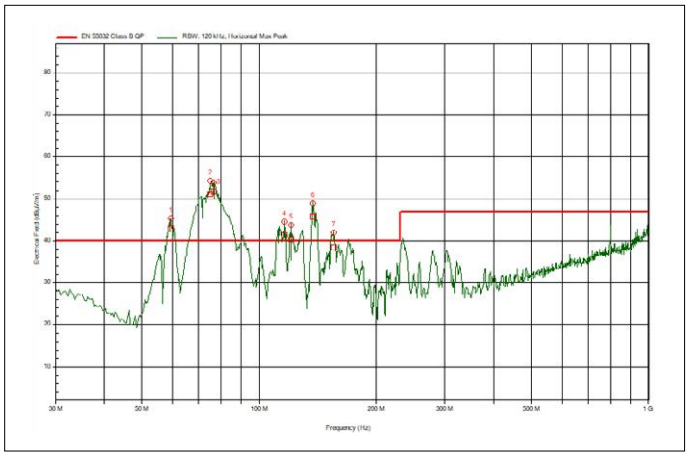


Figure 1. RE measurement for an LED lamp where it's clear that the standard is exceeded.

Table 1. Measurement results for the LED lamps		
CE 0.009 - 30 MHz	37.5%	0-5 dB under limit
CE 0.009 - 30 MHz	6.25%	5-10 dB under limit
CE 0.009 - 30 MHz	25%	10-15 dB under limit
CE 0.009 - 30 MHz	31.25%	>15 dB under limit
CE 30 - 300 MHz	6.25%	0-5 dB under limit
CE 30 - 300 MHz	12.5 %	5-10 dB under limit
CE 30 - 300 MHz	12.5 %	10-15 dB under limit
CE 30 - 300 MHz	68.75%	>15 dB under limit

Conclusions

Although we only carried out EMC pre-compliance tests, we can conclude that most of the LED lamps conform to the standards. But where does the interference come from? Does it depend on the specific circumstances (consider a length of 230V cable acting as an aerial and amplifying the signal)? Does it depend on the simultaneous use of several LED lamps or other sources of interference? Is it because some devices are more sensitive to interference? Was the research exhaustive enough? Do tell us if you know the answer (via editor@elektor.com; subject: LED lamps). ⏮

180608-03

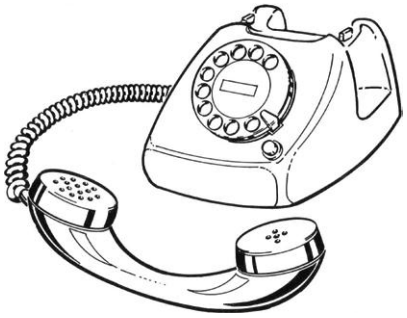
Web link

[1] Article support page: http://www.elektor.com/180608

@ WWW.ELEKTOR.COM
→ Book: EMC for Product Designers
www.elektor.com/emc-for-product-designers-5th-edition

From the Graphics Department, with Fun

Elektor's hand-crafted electronics artwork and cartoons



By **Jan Buiting**, Resident Editor, Elektor Retronics

Today, graphics, artwork, and text we view on paper and on screens can be safely assumed to be of digital origin. That's not implying the absence of human intelligence, but coming from a computer 'somehow'. Due to CAD, the craftsmanship of making schematics, PCB layouts and wiring diagrams by hand in a truly educative way using a drawing table, masking tape, rubbers and an array of pens, has disappeared entirely. For reasons not entirely nostalgic, I found some of Elektor's works of art worthy of reproducing and describing in this graphics-dominated Retronics instalment. Enjoy the pictures!

After yet another "reallocation" of staff, functions and office spaces within the Elektor organisation last year, and the subsequent cleaning out of cupboards, my colleague Patrick Wielders showed me three fat and heavy office binders from the former Elektor drawing department. The binders had miraculously survived all moves, reorganizations and general down structuring of the Elektor graphics department over the past 25 years, and even escaped the obliteration of the graphics department's complete paper archive back in 2005.

Historically

Shortly after its birth in 1961, Elektuur as a publishing house started to employ dedicated staff for the creation of 'engineering' graphics like schematics and wiring diagrams. These lads (and girls on occasion) were engineers with a background in electronics, internally trained, and instructed to uphold the standards as far as the educational aspect of the schematics was concerned. From a modest two or three in the mid 1960s, staff in the drawing department increased steadily and peaked to about 10 around 1980 when not everyone was lucky to have their own desk let alone a drawing table! I must hasten to add though that the number of 10 also included page layout staff, a 'cover artist', and draughtsmen assigned to material of a more illustrative nature, like exploded views and, admittedly, more frivolous things. Their legacy is the main subject here.

The style of the Elektor schematics came about naturally but got formalised to a degree by around 1970 when the now famous circuit symbols and even PCB track and pad shapes were created and printed on rub-off sheets made for internal use. Written in stone so to speak, and the envy of many of Elektor's competitors to this day. Every technical draughtsman was expected to use the symbols sheets to turn a 'circuit sketch' delivered by a lab worker into an Elektor-style schematic for reproduction in the magazines. Original schematics were made by rubbing off component symbols and connecting lines from Letraset sheets. New parts were drawn on the fly and pasted in by hand. Usually the work was done at a scale of 200%.

While jointly producing the magazine article in draft, lab workers and selected editors had to check the scribbled schematics found right beside the solder iron against the 'educational-level' work of art produced by Graphics and reduced to 100% as kind gesture. Errors got spotted (or not), correction rounds followed, you probably know the drill. The complete process was based on in-house lithographic reproduction and paste-up edits using hot beeswax to secure paper pieces onto carrier sheets.

During a company reorganization at around 1985, much of the page layout and typesetting staff left Elektor's graphics department to spin out to "Grafisch Bureau Sinke" (GBS), with Elektor as its main client and within walking distance from each other. Elektor however retained its draughtspersons and cover artist.

Enlivening the articles

I was delighted to discover that the three binders contained hundreds of 200%-size originals of the more artistic drawings that got produced (I estimate)

ESTD 2004
www.elektor.tv

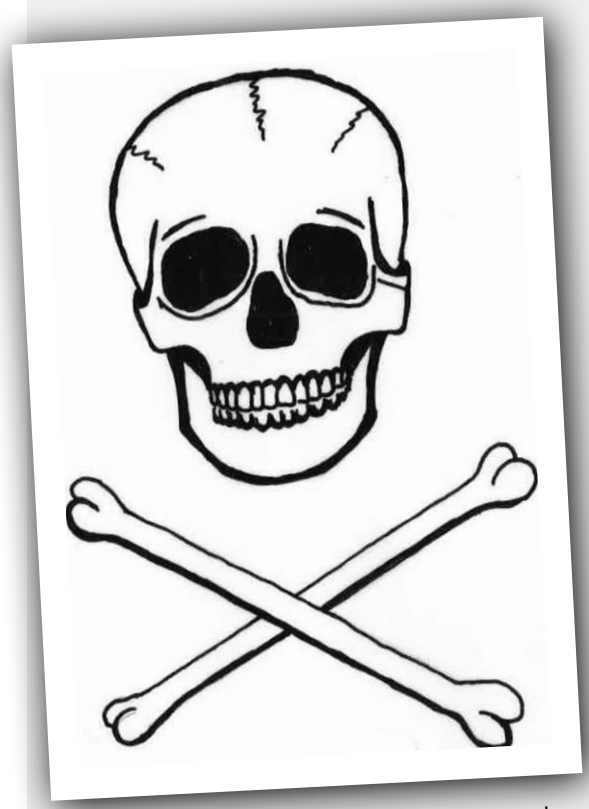
Retronics is a regular section covering vintage electronics including legendary Elektor designs.
Contributions, suggestions and requests are welcome; please telegraph editor@elektor.com

between 1980 and 1990. Their purpose was enriching Elektor magazine, and generally adding a lighthearted note to what must have been tedious and difficult to understand matter to many beginning readers. The circuit symbols and some '3D views' of components and electronic assemblies were made for all Elektor publications to use, i.e. books and magazines in all languages, however at the discretion of the editors. The bulk of the freehand drawings that were true "artist's impressions" were used for a magazine called Elex, which appeared between 1980 and 1992 and was aimed at young Dutch readers, and French as well for a brief period, later. Due to its more frolicking audience, the Elex editorial staff had a need to get their message across in a rather more playful way than the Elektor magazine staff supposedly serving a more 'highbrow' audience (depending on the language editions). Of the less serious artwork produced across a period of about 20 years, the cartoons had the most impact.

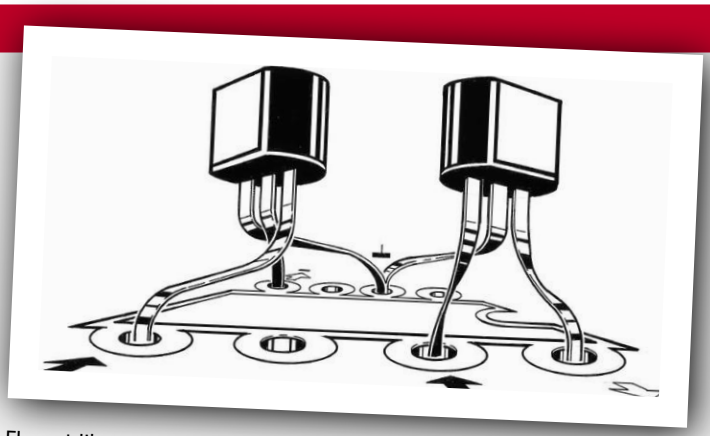
Further than Mr. Laurent Martin for the 3D component and assembly method views, and Mr. Jaap Kuiper for the cartoons, I have been unable to positively identify the creators of these wonderful illustrations, all made by hand in glorious black and white using an array of Rotring pens of which I remember a ticking sound due to the artist lightly shaking it to liquefy the ink. In some of the drawings you may espy the 'LM' and 'JAAP' signatures respectively. With compliments!

180574-B-01

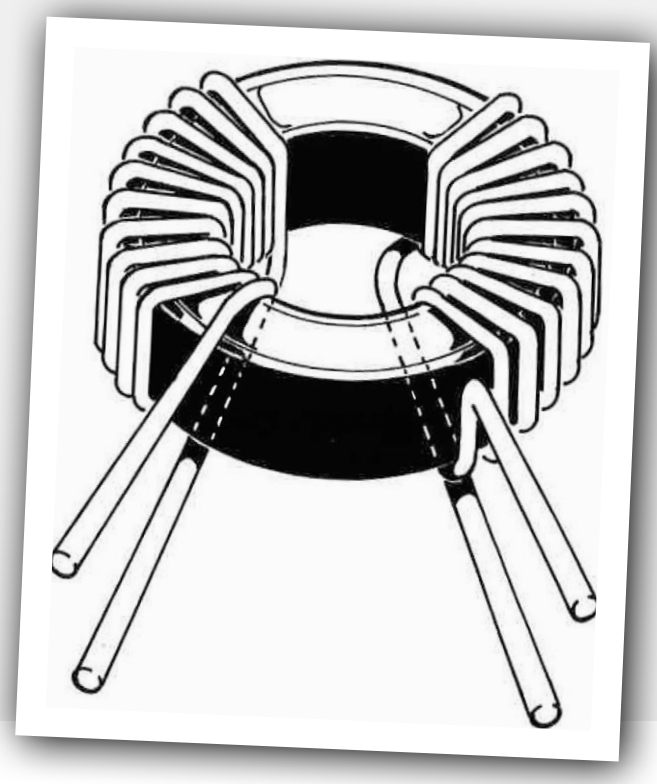
The Components



Every now and then Elektor felt the urge to warn readers of instant death and other discomforts due to high voltages present in the published circuits.

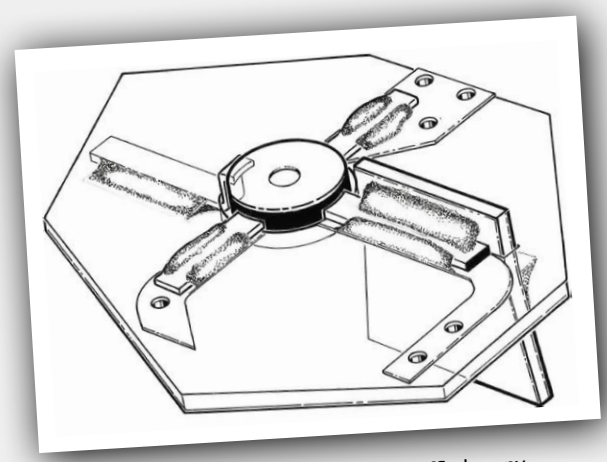


Elegant like a dancing pair, don't you agree? How to mount two TUNs (or TUPs?) in what appears to be a PCB shape for an integrated circuit. Is the arrangement a current source, a current mirror, or a poor man's Darlington? To be investigated!

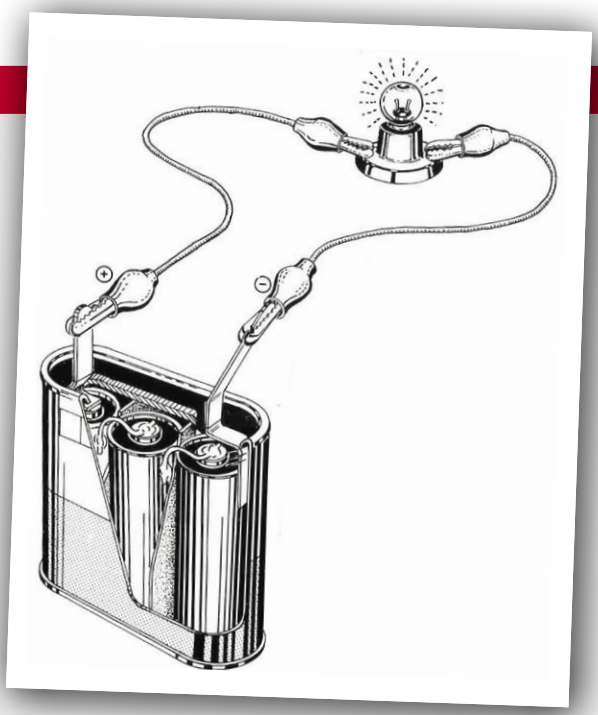


Inductors and home winding them was black magic to many Elektor readers who were DC minded overall as opposed to radio amateurs, who loved "The L". When the ferrite ring core arrived on the DIY market, the winding direction remained important, hence a nice drawing was produced to tell the story.

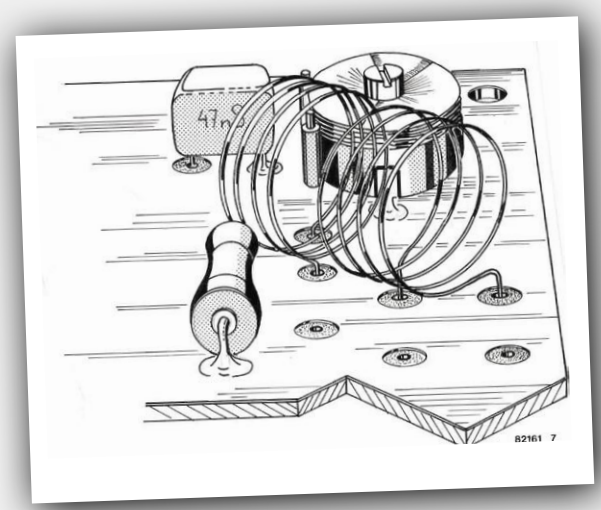
Assembly Methods in 3D Renderings



The BF9xx series of dual-gate MOSFETs baffled many readers due to their 'sunken' mounting in a hole. To add to the confusion Elektor prescribed a 'coffin' style, ceramic, wireless, decoupling capacitor fitted in a slot in the board, straight on the source terminal. Help! This artist's impression saved the editors a good 100 words of writing, and still failing to convince.

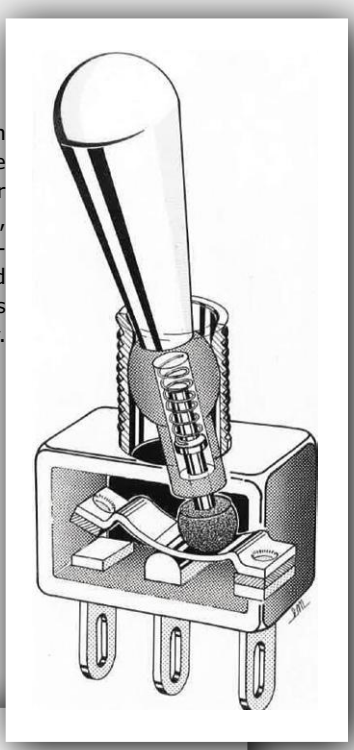


An iconic image of what must have been the start of many a career in electronics: lighting a 6-volt bulb on a 4-5 V flat battery. Note the croc clips and the lamp base! This gem was used in Chapter 1 of the incredibly popular *Electronics 1-2-3* book series published by Elektor in the 1970s and 80s. Today, I would advise every CEO and CFO in the electronic industry to perform the experiment.

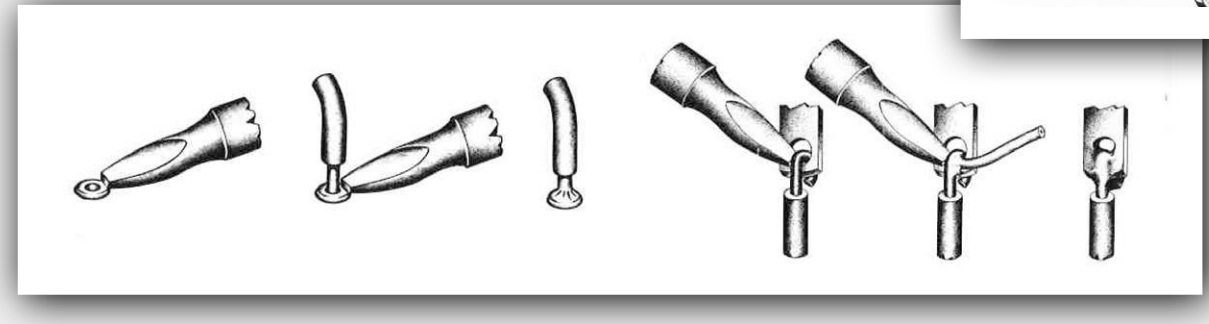


Tell a hobbyist about 6-pF PTFE (Teflon) trimmers, linked inductors, and decoupling capacitors, and he or she will benefit more from a drawing than from 1 Kwords on paper.

The secret of the flip switch revealed. As you may have discovered by debugging your own wiring jobs in the past, surprisingly and counter-intuitively… the contacts closed are **not** at the side the lever is flipped to! Here's why.

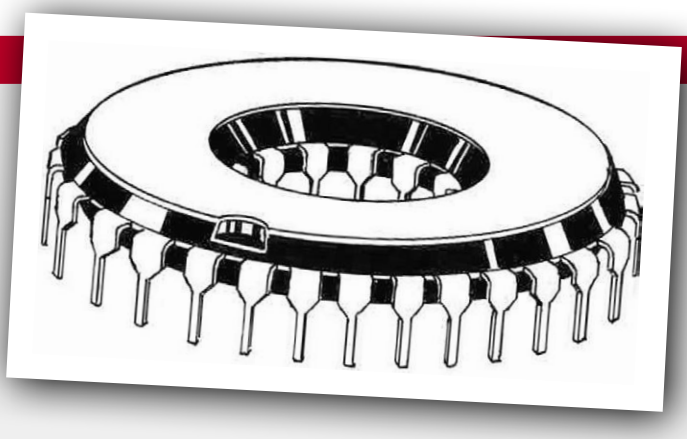


A miscellany of Good and Bad Practices for wire-to-terminal soldering (according to Elektor). Not sure if the industry would have agreed!



Did it Ever Exist?

When surround sound was introduced, hobbyists clamoured to build their own systems. The ICs were unobtainable in many cases, but with some ingenuity and plenty of phantasy, for sure that 'sound' can be shaped and bent a bit to achieve a surround-ish effect?



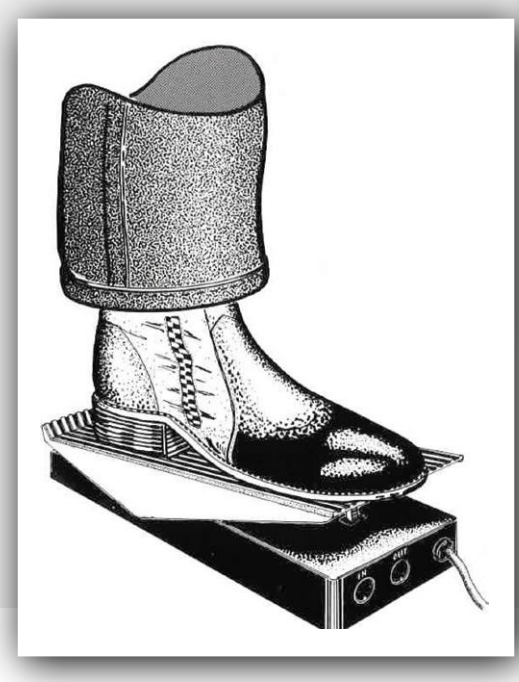


Elektor was suddenly into pressure sensors in IC form, and this might well be one of the earliest. In any case, the nozzle on top of the IC body had great artistic value. Later, this drawing was the source of inspiration for a totally fake photo of the "CD4711 Programmable Odour Generator", one of Elektor's most successful Joke Circuits contained in an annual *Summer Circuits* edition.

The Music Scene

Circuits and projects to support audio and musical instrument reproduction have been strong if not dominant elements in Elektor for decades. To pull 'outsiders' into a magazine about dry electronics requires picturing things they recognize. This is called "bait". At least in Elektor, no subject area has drawn more 'outsiders' into the electronics hobby than audio amplifiers, and especially hi-fi. Sadly, many audiophools came out of it.





In the 1980s everyone was after building guitar effects gear, probably to impress girls and to mask total inaptitude to music. Here's a belated hippy foot pressing what I believe to be a wah-wah pedal. Note: no flared trousers.

The Cartoons

To “explain” a cartoon is sacrilege. Still, most cartoons were aimed to the Dutch Elex market and some elucidation may be necessary for international readers of Elektor, 20 to 30 years after they were made. Disclaimer: Any resemblance between persons pictured in the cartoons, and Elektor staff is purely coincidental.



Hmm, how to play this score on the electronic instrument?



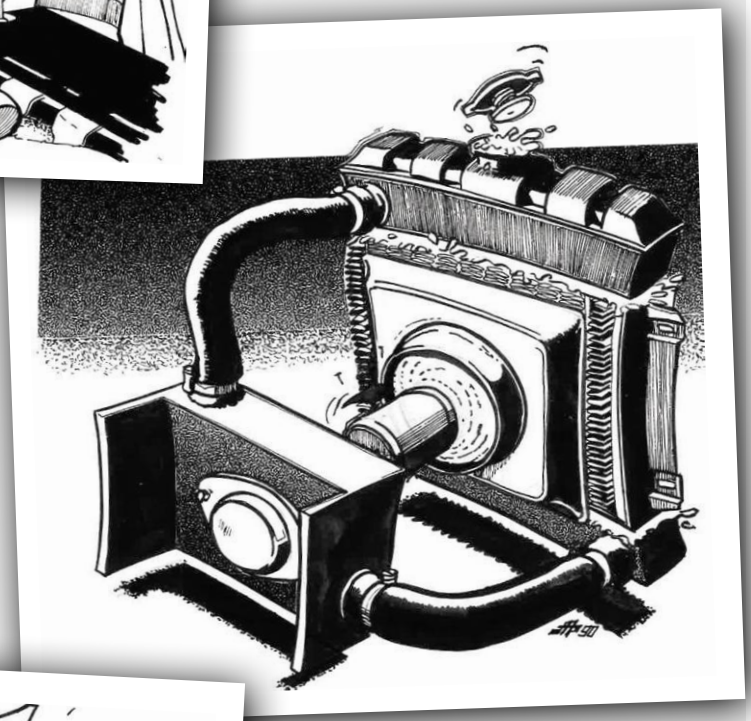
The Video Splitter!



Running Lights!
‘AAN’ is Dutch for on; ‘UIT’, for off.



I say, to conquer & explore electronics, much soldering, courage and test gear is needed!



Heatsinking and radiator-assisted cooling of the power semiconductor!



Frying, toasting, blowing up, components and gear is... fun sometimes, but actually disconcerting and undesirable. For example, never switch the poor multimeter to the ohm range when measuring thick amps!



How Can You Implement Ethical Values Concretely in Your Project?

By **Tessel Renzenbrink** (Netherlands)

The ethical aspects of technology are receiving more and more attention, which is good news. But how can you make ethical values a practical part of your project? This was the main question at a meeting in Amsterdam where three speakers shared their experience in this area. Gijs Boerwinkel from the Waag Society described how you can determine ethical values when you are dealing with a large and diverse group of users. Tada manager Douwe Schmidt talked about anchoring the values. The Tada Manifesto on the responsible use of data has been incorporated in the coalition agreement of the Municipal Council of Amsterdam. Mart van Santen, CTO of Greenhost, explained that maintaining principles means not only taking a critical look at your own organisation, but also engaging in a dialogue with your supply chain partners.

How do you determine ethical values?

In order to manage your project with regard to ethical values, you must first determine what the values are. One way to do this is to listen – listen to as many people as possible. That is one of the methods used by the Digital Identity Lab [1]. The Lab is a long-term project of the Waag Society, a knowledge centre that gets ordinary people and experts involved in giving form to technology.

Gijs Boerwinkel, Community Manager at the Waag Society, explained that the Lab carries out research on digital identity, which is becoming increasingly important in our lives. Everything from the user IDs for logging in to the websites of our health insurance providers to the text messages we send to those near and dear to us is a part of our digital identities. Right now we have little control over this. Our digital identities are fragmented and stored in the databases of government bodies and companies. We do not know what happens with the trip data collected when we use our public transport travel passes, or what Facebook does with our data.

On behalf of the Ministry of Internal Affairs and the Association of Netherlands Municipalities, the Digital Identity Lab researches methods and applications that could give us more control over our online identities. For example, an app that allows you to prove your identity online without having to submit a copy of your passport. The Lab checks if apps like this fulfil the wishes of users. For that purpose it is first necessary to find out what users want. In order to do so, the Lab organises meetings in various parts of the country. Not only with experts and software developers, but above all with ordinary citizens who will be using the apps. They also take to the street to interview passers-by. With input from a representative sample of the Dutch population, the Lab can identify the values that citizens consider important with regard to their online identity. Their findings indicate that these values include more transparency (insight into the data) and control over the data. Those are values that you would expect in this context. However, another frequently mentioned value is ease of use. That might sometimes be overlooked in an expert meeting with app developers and privacy activists. To avoid blind spots, the Lab involved as many stakeholders as possible in the formulation of the values. Getting back to the question of how to determine the ethical values for a project. It's a good idea to not only think about this yourself, but also listen to others.

How do you implement the values?

The discussion of the Tada project [2] addressed the question of how to give concrete form to the values once they have been determined. Tada is about the use of data in cities. With the advent of the smart city, that is more and more important. According to the initiators of Tada, this is a good thing. Data can help to make cities cleaner, more efficient, and more democratic. However, use of data must be based on specific ethical values. Tada is an initiative of about fifty organisations, including the Amsterdam Economic Board and the Waag Society, as explained by Tada Executive Board member Douwe Schmidt. Like the Digital Identity Lab, Tada organised a number of meetings with stakeholders to determine the ethical values. That resulted in a manifesto with six principles that must be met for the use of data in the city. One of these principles, for example, is inclusivity, which means that the use of data must not lead to the exclusion of particular groups.

The next step is the implementation of the values. Schmidt asked, "How do you prevent the manifesto from ending up at the bottom of the drawer? To ensure that people can actually do something with the manifesto, we have launched a

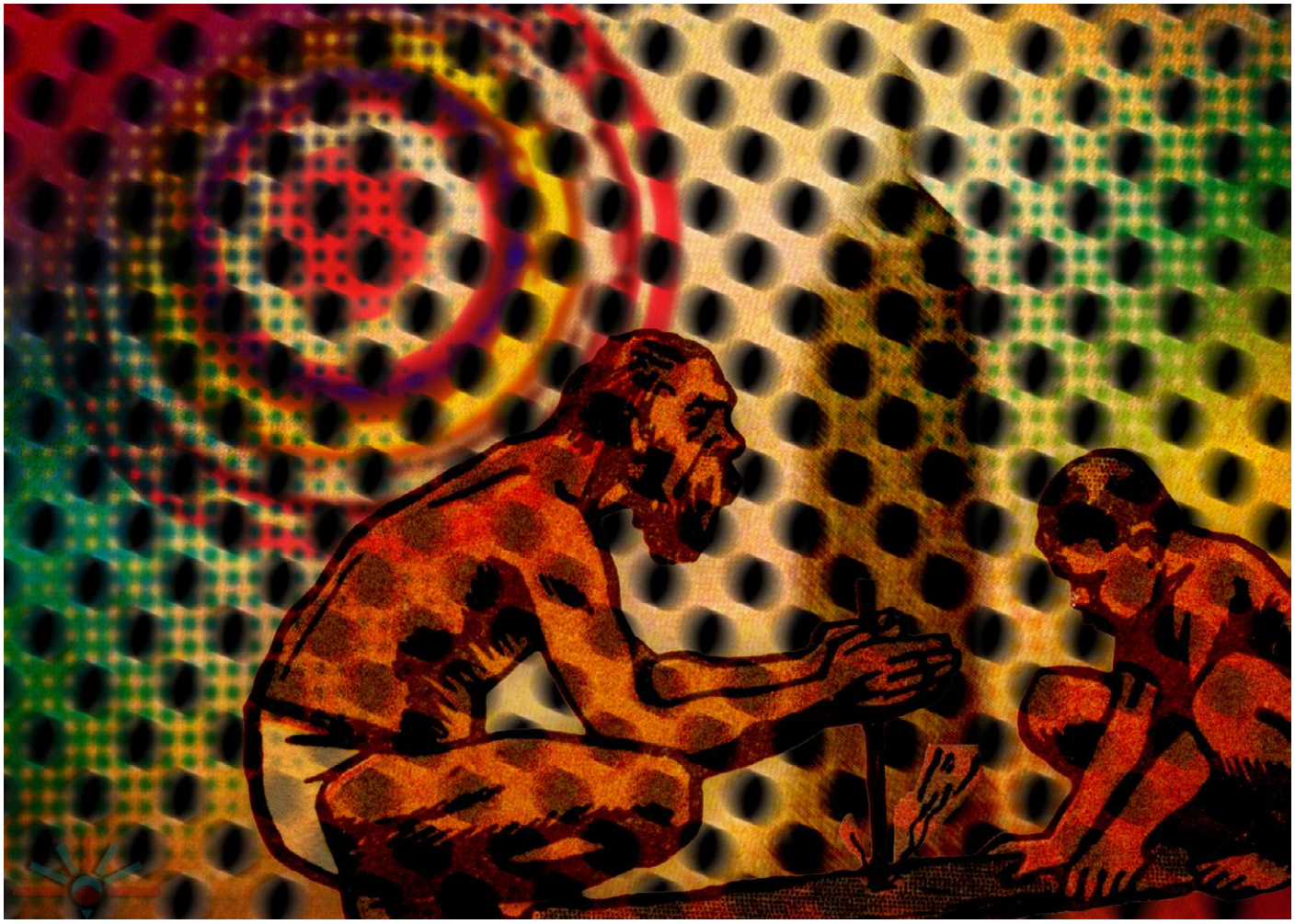


Illustration: Dawn (A.K. Rockefeller). Creative Commons licentie BY-SA 2.0 / Flickr: https://is.gd/Gbbqwe.

campaign. We call on everyone to sign the manifesto: citizens, companies, social organisations, and so on. That's because data in the city is not the business of the municipal authorities alone. It belongs to and affects everyone. We have also organised workshops for members of municipal councils." Those meetings have borne fruit: the Amsterdam Municipal Council that convened in 2018 has incorporated the manifesto in the coalition agreement.

How do you put the values into practice?

Mart van Santen, CTO and co-founder of web hosting provider Greenhost [3], explained how to make ethical values a permanent part of your project. Greenhost has adopted sustainability as a leading principle for their business activities. Internet technologies consume a lot of electricity. To limit the environmental impact as much as possible, Greenhost opts for green power and energy-efficient equipment. That is not always easy, since servers must meet specific technical requirements in addition to being durable. "It's always a question of finding a balance", says Van Santen.

Doing business in a responsible manner is not limited to your own business processes; you also have to take a critical look at the rest of the production chain. Greenhost chose a data centre that claimed to run entirely on green power, but they found out later that the data centre worked in part with 'fake green' power, which means power that is legally allowed to be called green but does not contribute to increasing the share of renewable electricity generation. Greenhost brought that up with the data centre operator. They learned that the data centre was already working on an improvement by switching to European wind power. In addition, they promised to purchase Dutch wind power for the share of electricity consumed by Greenhost.

The meeting 'Principles in Technology: How do you make them concrete?' [4] took place on 12 December 2018 in Amsterdam. It was organised by the Digital Identity Lab, Greenhost and Tada.

(180572-B-03)

Web links

[1] Digital Identity Lab: https://digitaleidentiteit.waag.org/

[2] Tada: https://tada.city/

[3] Greenhost: https://greenhost.nl/

[4] Principles in Technology meeting: https://digitaleidentiteit.waag.org/event/principes-in-technologie-hoe-maak-je-ze-concreet/

welcome in your ONLINE STORE

EDITOR'S CHOICE



MonoDAQ-U-X – Multifunctional USB Data Acquisition System

With its galvanically isolated USB-C port and sturdy aluminium enclosure the new MonoDAQ-U-X is a robust and versatile data acquisition device. Multiple devices can be connected to the same computer.

The MonoDAQ-U-X features 8 ports individually configurable as analogue or digital input or output. Analogue inputs can also be grouped to create differential inputs. Besides these 8 configurable ports a two-way digital port is available that supports the 1-Wire protocol. An analogue output to drive sensors or to power a subsystem (up to 24 V, 1 W) has not been forgotten either. Furthermore, integrated in the MonoDAQ-U-X are a temperature and humidity sensor and a 6-axis IMU.



Clemens Valens
(Elektor Labs)

www.elektor.com/monodaq

Elektor Bestsellers

1. Miniware TS100 Soldering Iron
   www.elektor.com/miniware-ts100



2. Android App Development for Electronics Designers
   www.elektor.com/android-app-development
3. Mastech MS8911 Smart SMD Tester
   www.elektor.com/mastech-ms8911
4. Raspberry Pi 3A+
   www.elektor.com/rpi-3a-plus
5. PoE HAT for Raspberry Pi 3B+
   www.elektor.com/poe-hat-rpi-3-plus
6. Bakeey UM25C USB Multimeter Logger
   www.elektor.com/bakeey-um25c

Weller WT 1014 Digital Soldering Station (90 W)



The Weller WT 1014 soldering station set includes the WT 1 supply unit, the WSP 80 Robust soldering iron and the WSR 201 safety rest. It is stackable and thus guarantees more space at the workplace. With an integrated usage sensor, the soldering tool switches off automatically.

Member Price: £314.95 • €350.10 • $401

www.elektor.com/weller-wt-1014

PeakTech 8200 Measuring Accessories Set



This comprehensive collection of test accessories for the electronics area ensure the best suitable measuring adapter for the desired application for each digital multimeter. All clips and cables are manufactured to the latest safety standards and provide a safe implementation in every application for the user. The material used is a flexible and non-slip plastic that can endure the harshest kind of usage.

Member Price: £36.95 • €40.46 • $47

www.elektor.com/peaktech-8200

JoyPi – Experimenter's Briefcase for Raspberry Pi

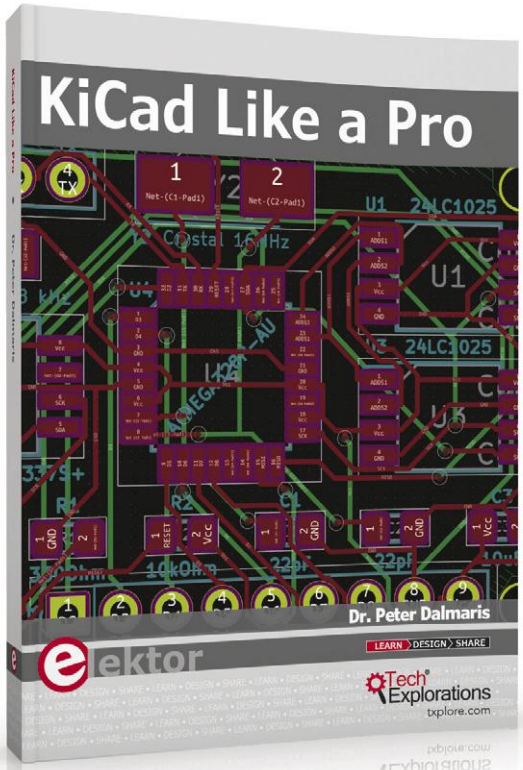


The JoyPi is an experimental case based on the Raspberry Pi and is ideal for getting started with electrical engineering and programming. The sophisticated case system offers a perfect all-in-one environment and puts an end to many fiddly small parts solutions and cable chaos on the workbench. The JoyPi has a large number of sensors and modules which can be operated independently of your own previous knowledge

Member Price: £201.95 • €224.10 • $257

www.elektor.com/joypi

NEW



KiCad Like a Pro

This book will teach you to use KiCad. Whether you are a hobbyist or an electronics engineer, this book will help you become productive quickly, and start designing your own boards. This book takes a practical approach to learning. It consists of four projects of incremental difficulty and recipes.

The projects will teach you basic and advanced features of KiCad. If you have absolutely no prior knowledge of PCB design, you will find that the introductory project will teach you the very basics. You can then continue with the rest of the projects. You will design a board for a breadboard power supply, a tiny Raspberry Pi HAT, and an Arduino clone with extended memory and clock integrated circuits.

Member Price: £32.95 • €35.96 • $42

www.elektor.com/kicad-like-a-pro

JOY-iT DPS5005 Programmable Power Supply



This power supply combines analog and digital technologies in an advanced design, offering outstanding precision and accuracy. The adjustable output reaches up to 50 V or 5 A and can be precisely configured in 10 mV or 1 mA steps. The operation of the device has been optimised for particularly simple operation and the colour display offers a particularly detailed and comprehensive presentation of information.

Member Price: £34.95 • €38.66 • $45

www.elektor.com/dps5005

Electronics for Space



The space industry is continually growing and new products and services will be required. Innovation is needed for the development of this industry. Today it is no longer possible to follow all the events in field of space. The space market is growing and activities are increasing, especially the market for small-satellites. This book wants to help close the gap and encourage electronic engineers to enter into the fascinating field of space electronics.

Member Price: £20.95 • €22.46 • $26

www.elektor.com/electronics-for-space

NixieCron-M4 – 4-digit LED Nixie Clock



The NixieCron M4 offers a high-precision watch chip, temperature compensated with a deviation of only a few seconds per month. It comes with a sound module, that plays hour gong and announces the current time at the push of a button. You can also connect it with the DS18B20 temperature sensor and configure it via a rotary knob.

Member Price: £120.95 • €134.10 • $154

www.elektor.com/nixiecron-m4

Hexadoku The Original Elektorized Sudoku

Traditionally, the last page of Elektor Magazine is reserved for our puzzle with an electronics slant: welcome to Hexadoku! Find the solution in the gray boxes, submit it to us by email, and you automatically enter the prize draw for one of five Elektor book vouchers.

The Hexadoku puzzle employs numbers in the hexadecimal range 0 through F. In the diagram composed of 16 × 16 boxes, enter numbers such that **all** hexadecimal numbers 0 through F (that's 0-9 and A-F) occur once only in each row, once in each column and in each of the 4×4 boxes (marked by the thicker black lines). A number of clues are given in the puzzle and these determine the start situation.

Correct entries received enter a prize draw. All you need to do is send us **the numbers in the gray boxes**.



Solve Hexadoku and win!

Correct solutions received from the entire Elektor readership automatically enter a prize draw for five Elektor Book Vouchers worth **$70.00 / £40.00 / €50.00 each**, which should encourage all Elektor readers to participate.

Participate!

Ultimately March 18, 2019, supply your name, street address and the solution (the numbers in the gray boxes) by email to: **hexadoku@elektor.com**

Prize Winners

The solution of Hexadoku in edition 1/2019 (January & February) is: **6A258**.

The €50 / £40 / $70 book vouchers have been awarded to: Siegfried Keep (Germany); Mehmet Can Cakar (Turkey); Per Troelsen (Denmark); Lucien Zadikian (France); Vladimir Koci (Czech Republic).

Congratulations everyone!

		0			E		9	3		D			4		
		3	2				6	9				B	7		
D	4				1					2				A	E
	8			F		2	4	6	A		7			9	
			3		5	F	A	C	2	1		7			
1		F		3		E			6		9		C		0
			B	6	7	9			8	A	E	3			
7	9		A	1							0	E		4	6
9	6		7	A							2	4		C	1
			C	7	8	6			4	9	A	0			
3		A		0		1			D		5		8		2
			5		F	4	2	E	C	3		9			
	1			9		5	F	2	B		3			6	
4	7				6					E				2	A
		C	6				1	8				D	0		
		E			D		7	A		6			3		

3	8	5	A	C	9	E	F	7	1	6	0	D	2	B	4
6	7	F	1	A	D	B	4	8	3	2	9	E	5	0	C
9	4	E	B	2	0	1	7	D	5	A	C	F	6	3	8
0	C	D	2	3	5	6	8	E	F	4	B	7	1	9	A
7	A	C	8	6	B	D	E	9	4	F	2	1	3	5	0
2	5	0	E	4	3	8	9	1	B	C	6	A	7	D	F
F	9	B	D	1	C	7	0	3	E	5	A	6	4	8	2
1	6	3	4	F	2	5	A	0	7	D	8	9	E	C	B
B	E	1	9	0	8	4	5	2	C	7	F	3	D	A	6
A	F	2	0	D	1	3	C	4	6	B	E	5	8	7	9
4	D	7	C	9	6	A	2	5	8	1	3	B	0	F	E
5	3	8	6	7	E	F	B	A	9	0	D	2	C	4	1
8	1	9	F	5	7	C	D	6	A	E	4	0	B	2	3
C	0	6	5	8	A	9	1	B	2	3	7	4	F	E	D
D	B	A	7	E	4	2	3	F	0	8	1	C	9	6	5
E	2	4	3	B	F	0	6	C	D	9	5	8	A	1	7

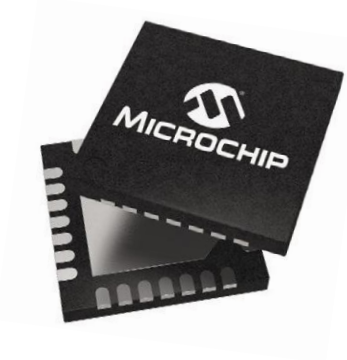
The competition is not open to employees of Elektor International Media, its subsidiaries, licensees and/or associated publishing houses.



Discover Your Power

Flexibility to Choose the Desired Power Solution

As a leading supplier with a comprehensive power management and monitoring portfolio, Microchip gives you the power, flexibility and confidence to choose the right solution for your design.



Managing your system's power usage is crucial to achieving the performance that your design requires. Our product portfolio of power monitoring devices allows you to accurately measure active, reactive and apparent power, Root Mean Square (RMS) current and voltage, line frequency, and power factor. Our broad selection of power management devices including DC-DC controllers and regulators, MOSFETs and MOSFET drivers, voltage supervisors and references, and power modules enables you to efficiently design a solution to manage the power requirements of your system.

From reference designs to evaluation boards to simulation tools, you'll reduce design-in time and minimize risk with the comprehensive support Microchip can offer.

Discover your power at
www.microchip.com/PowerSolutions

MICROCHIP

The Microchip name and logo and the Microchip logo 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.
© 2018 Microchip Technology Inc. All rights reserved. DS20006065A. MEC2230Eng11/18

reichelt elektronik

Technology connects.

- ✓More than 100,000 products
- ✓Top price/performance ratio
- ✓High availability and reliable delivery
- ✓Strong brands and choice quality

PROFESSIONAL TECHNOLOGY FOR WORKSTATIONS AND LABORATORIES

FULL POWER AT THE WORKPLACE!

Weller®

The Weller WE1010 Education Bundle

70 watts of power for professional requirements!

The most powerful soldering station in its class! This economy bundle will equip you for any task. In addition to the WE1010 soldering station this set includes 170 mm Weller Xcelite side cutters, a roll of Weller WSW SCN N1 solder, and an additional ETB 2.4 mm soldering tip.

- ESD-safe soldering station, soldering iron and heat-resistant silicone cable
- Temperature stability and lock for protection of soldering tip and component
- Password protection saves all settings



Order no.: WELLER WE1010EB

NEW £177.53 (£ 147.95)

AC power cable, 1.8 m
- UK plug > IEC socket



NKSK UK 180 **£ 2.67** (£ 2.22) **ORDER NOW**

Laboratory power supply

0 - 30 V, 0 - 10 A

Due to its very handy design and low weight, this switching power supply unit is ideally suited for applications in development and education.

- Voltage and current preselection with switchable output
- Modes for constant current and constant voltage with connected load
- Automatic switch-off in case of short circuit
- finely adjustable output values: 10 mV, 10 mA

PeakTech®



QUALITY FOR TRAINING, LABORATORY & SERVICE

Order no.: PEAKTECH 6226

£85.60 (£ 71.33) **BEST SELLER**

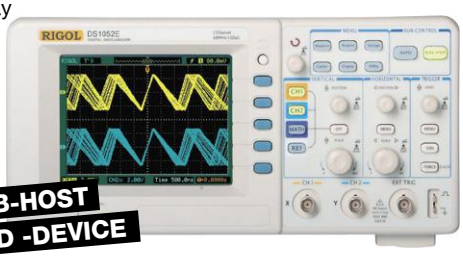
Digital storage oscilloscope for beginners!

50 MHz, 2 channels

RIGOL Innovation or nothing

Versatile digital storage oscilloscope, also ideal for mobile use.

- illuminated full-colour TFT display
- Sampling rate: 1 GS/s
- various trigger types,
- incl. edge, video, pulse width
- with Ultra Sigma Software



USB-HOST AND -DEVICE

Order no.: RIGOL DS1052E

£267.63 (£ 223.03)

Compact multimeter

with 2,000 counts

CHAUVIN ARNOUX

- Voltage measurement: 200 mV - 600 V AC/DC
- Current measurement: 200 µA to 200 mA
- Resistance measurement: 200 Ohm to 20 MOhm
- continuity (buzzer) and diode tests



Order no.: CHAU P01191740Z

£44.60 (£ 37.16) **NEW**

Daily prices! Price as of: 30. 1. 2019

Payment Methods: VISA MasterCard Maestro PayPal

The statutory cancellation provisions apply. All prices are stated in £ including statutory value added tax, plus shipping charges for the entire shopping basket. Our Terms and Conditions (available at https://rch.lt/TERMS or on request) apply exclusively. Actual appearance of products may differ from photos. We accept no liability for misprints or errors; prices subject to change.

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

www.reichelt.co.uk

ORDER HOTLINE: +44 203 808 95 25