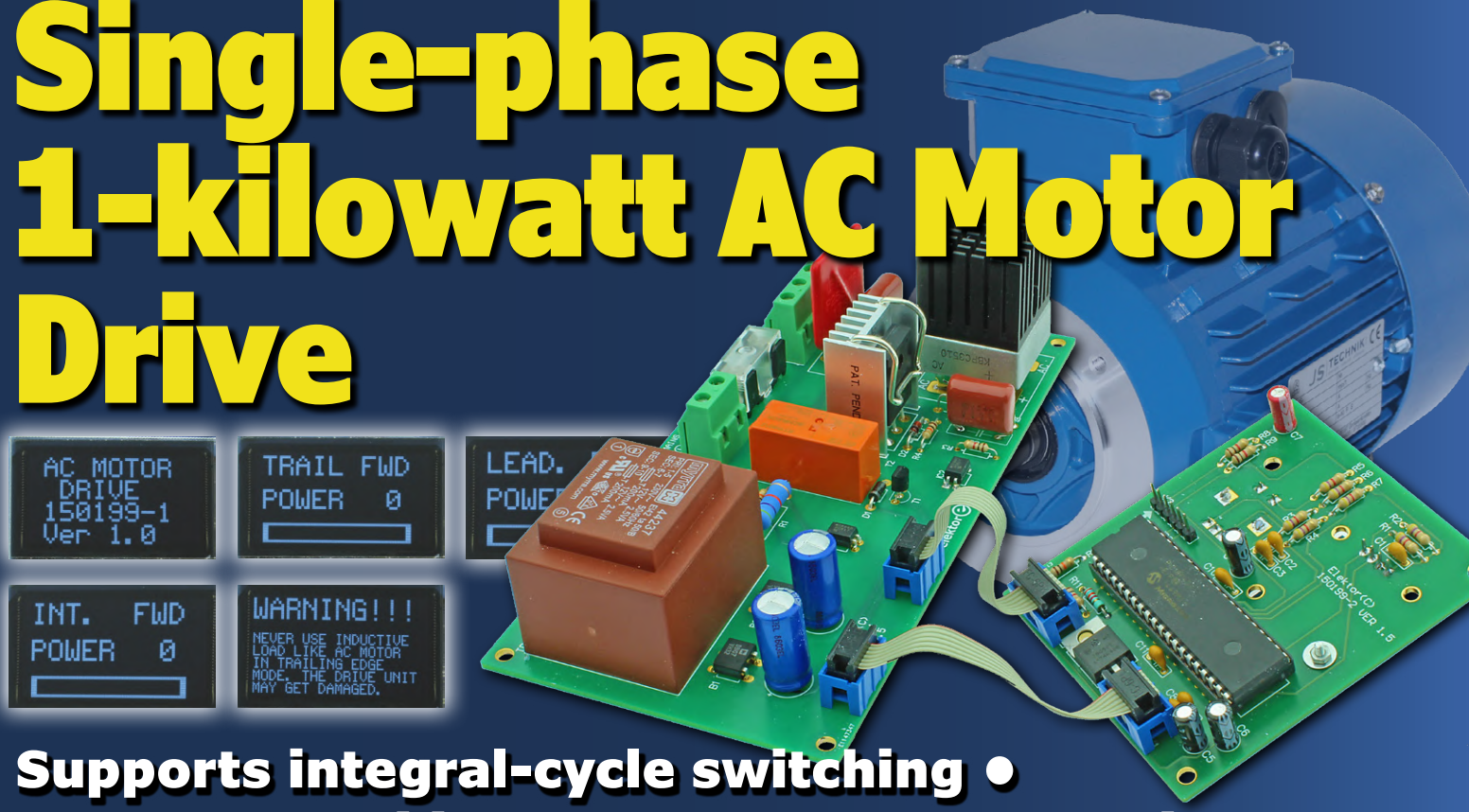
Edition 1/2020 | January & February | www.elektormagazine.com

elektor magazine

DESIGN BUILD SELL ELECTRONICS

Single-phase 1-kilowatt AC Motor Drive



Supports integral-cycle switching • phase-cut trailing edge • phase-cut leading edge

GUI with Touch — for ESP32, RPi & Co.



Graphical user interfaces with the LittlevGL library

Extendable Environmental Monitoring System

based on a Geiger-Müller tube



Analogue Electronics Design • Arduino Pro IDE - First Impressions • Automotive Headlight Tuning • Beyond Electronics • Capaci-Meter • Developer's Zone • Elektor Labs Challenged: a DIY LoRa Tracker • First Steps with RISC-V • Focus On: Autonomous Driving • How-To: Debounce a Mechanical Contact or Switch • In a Nutshell: Text for Microcontroller • Lab, Sweet Lab • Lego Electronic Train Set from 1968 • Monsanto MAN1 LED Display • "Not Just Any Guy's Project" • Practical ESP32 Multitasking • Review: Toolcraft Digital Soldering Station • ... and more

The Elektor Community

82	203637	1040	546	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

- ✘ 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

- ✔ 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 Magazine,
English edition

Edition 1/2020
Volume 46, No. 499
January & February 2020

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

www.elektor.com
www.elektormagazine.com

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

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

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

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

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

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

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

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



The Next Ten Years

A new decade of electronics has arrived. Technologies such as SigFox and LoRa are ideal for the expanding Internet of Things, and there are innovative processor architectures such as RISC-V. Developers are expecting to prepare ready to go applications using affordable hardware and software. The modular approach, which is becoming increasingly common, is the quickest way to reach one's goal. But for fine-tuning, know-how on the interaction of individual components remains indispensable. At the same time, a growing number of electronics engineers are taking an interest in marketing privately developed solutions.
An established and class-leading electronics magazine, Elektor should aim at being a faithful companion to its readers in all these developments. That is why we introduce a few new sections as of this issue. We intend to introduce new controller platforms on a regular basis (p. 74). We shed light on the analogue interaction of components in our expert section on p. 88. In our interview, the minds behind successful (open-source) projects have their say — you will also learn a lot there about how you can make your own projects known to a wider audience (p. 14). There are also pages for professional developers and those who aspire to become one (pp. 26, 56). We especially like to respond to the many requests of our readers to do more for beginners (pp. 23, 53, 86, 92).
The core of our magazine remains our projects, in which we also highlight the software more than before (as on p. 7). And last but not least, we subject the most interesting products from the Elektor Store to a practical test (p. 54).
From now on, our editorial team is truly international with myself as International Editor-in-Chief (previously I was responsible for the German edition only). Without the experience and knowledge of my colleagues who have managed their language editions for many years, naturally I would not be able to manage the task. Only as a team can we offer you a magazine and an online presence covering all areas of electronics in the best possible way. We are all looking forward to the next 10 years with you!

Jens Nickel

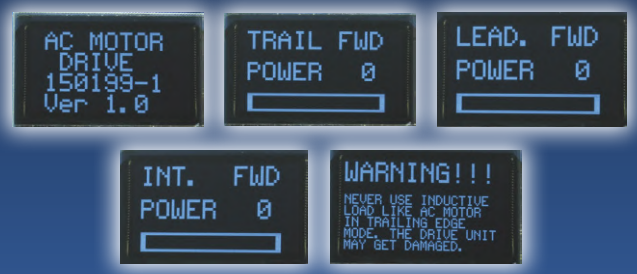
The Circuit

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

This Edition

Volume 47 – Edition 1/2020
No. 499 January & February 2020

Single-phase 1-kilowatt AC Motor Drive





Three-mode design: integral-cycle switching,

GUI with Touch — for ESP32, RPi & Co.

Graphical user interfaces with the LittlevGL library



Regulars

23 How-To: Debounce a Mechanical Contact or Switch
A switch is either open or closed, isn't it?

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

42 Interactive
Corrections & Updates || Questions & Answers

56 Steeped in Electronics
Outfitting the e-lab and e-workshop

58 Peculiar Parts, the series
Monsanto MAN1 LED Display

86 Starting Out in Electronics
Easier than imagined!

88 Analogue Electronics Design
Case Study #1 — Section 1: MEMS Microphone… *Testing 1-2-3*

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

106 Lego Electronic Train Set from 1968
50 years on, electronic toys still fascinate

110 Beyond Electronics
The MX3D Bridge Senses the City

112 Elektor Store
what's available @ www.elektor.com

114 Hexadoku
The original Elektorized Sudoku

Features

14 "Not Just Any Guy's Project"
Interview with Gábor Kiss-Vámosi, the developer of LittlevGL

44 Automotive Headlight Tuning
Legal, illegal — it matters!

50 Review: Toolcraft Digital Soldering Station

53 Arduino Pro IDE - First Impressions

54 Two Thermal Imaging Cameras Compared
Seek ShotPro and Seek Compact

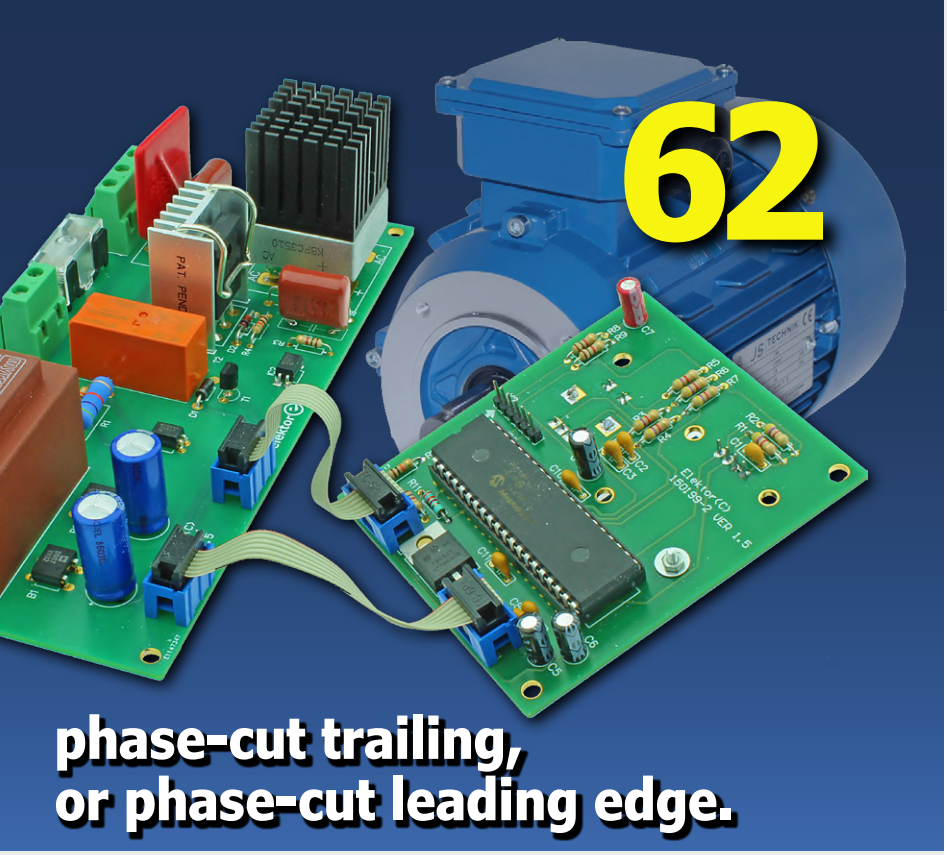
72 Lab, Sweet Lab
A glimpse of The Holy Place — no unauthorized entry.

98 Focus On: Autonomous Driving
State of the art at a glance

104 Start-Up Leaders and Innovators Talk "Innovation 4.0" in Munich



Interview with Gábor Kiss-Vámosi, the developer of LittlevGL



elektor
DESIGN BUILD SELL ELECTRONICS

59 Thermoelectric Tea Timer
First dealings with energy harvesting

62 Single-phase 1-kilowatt AC Motor Drive
Three-mode design: integral-cycle switching, phase-cut trailing, or phase-cut leading edge.

70 RPi Gets a Fast 3.5" Touch Display
More power at no extra charge

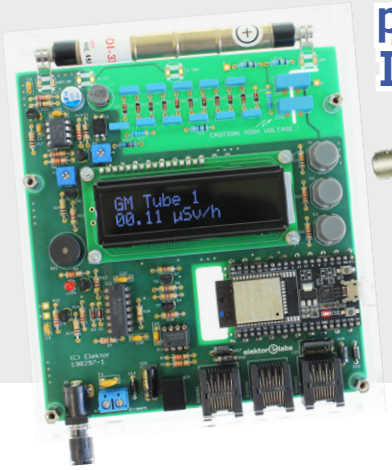
74 First Steps with RISC-V
Trying out the LoFive Board

78 Elektor Labs Challenged: a DIY LoRa Tracker
Glitches and results in electronics development

80 Sigfox and the IoT (2)
Registration in the Sigfox network

94 In a Nutshell: Text for Microcontrollers
Using compression to save memory

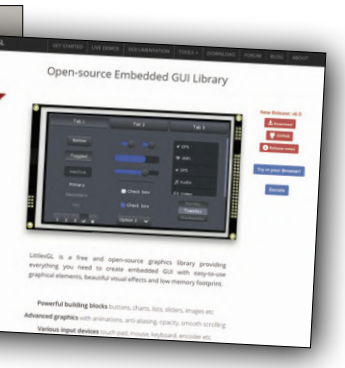
Extendable Environmental Monitoring System



publishes data on IoT platforms



28



14

Projects

6 GUI with Touch — for ESP32, RPi & Co.
Graphical user interfaces with the LittlevGL library

18 Capaci-Meter
with two-digit Dekatron-style LED display

28 Extendable Environmental Monitoring System
publishes data on IoT platforms

36 Practical ESP32 Multitasking
Task programming with FreeRTOS and the Arduino IDE

Next Editions

Elektor Magazine Edition 1/2020

Elektor LoRa Controlled Switch • "My-IoT" Button • How-To: Calculate the Prospective Short-Circuit Current or PSCC • Review: Andonstar AD407 • FreeRTOS for ESP32 (2) • ESP32 Doorbell via Telegram • Analogue Electronics Design (2) • Optical Probe for Oscilloscopes • DIY PC for the Electronics Home Lab • Radar Board • Sigfox and the IoT (3) • Small Circuits & Tricks • Retronics.

Elektor Magazine edition 2/2020 covering March & April 2020 is published on 27 February 2020. Delivery of printed copies to Elektor Gold Members is subject to transport. Contents and articles subject to change.

Elektor Industry Edition 1/2020

Elektor Industry issue 1/2020 is a special edition for the **Embedded World Exhibition & Conference 2020** in Nürnberg, Germany (February 25–29, 2020). The magazine will cover microcontrollers, embedded technology, programming and tools, with a miscellany of contributions from companies, industry specialists, Elektor editors, and free-lance authors.

Elektor Industry edition 1/2020 is published around 14 February 2020 to Elektor Gold Members in print and to Elektor Green Members as a pdf download. The edition will also be available for purchase at www.elektormagazine.com.

GUI with Touch — for ESP32, RPi & Co.

Graphical user interfaces with the LittlevGL library

By **Mathias Claußen** (Elektor Labs)

Many projects with a microcontroller inside have to display something. In the past, 2-line text displays were common and sufficient, but today graphic LCDs (or OLED displays) are increasingly used. There are several libraries for displaying attractive graphics as well as for touch operation. LittlevGL is a library under the liberal MIT license, which can be easily adapted to various displays and controllers. In this article, we demonstrate the whole thing using an ESP32 board and a touch LCD that displays data from a weather station.

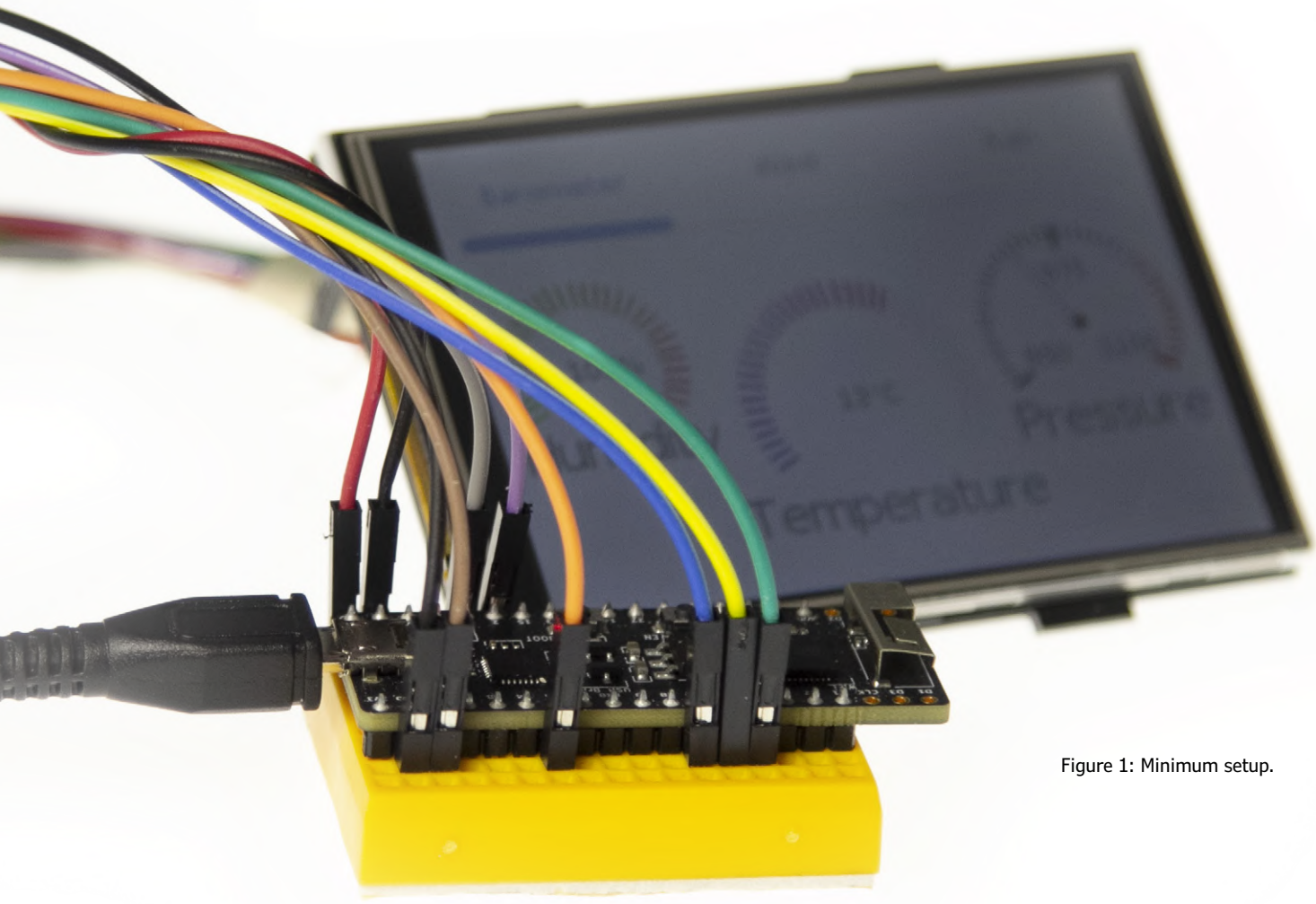


Figure 1: Minimum setup.

Graphics user interfaces often require much more programming effort than the 'actual code' that does the important functions of the project. To be able to concentrate on the essential aspects of the project, ready-made solutions or the use of ready-made libraries for the graphical output of information are becoming more popular and important. Well-known names are µGFX, emWin or TouchGFX for STM32 boards, to name just a few. They all have advantages and disadvantages, such as commercial

licensing or the connection to certain controller manufacturers. Of course, you could also develop a library yourself, but that's a huge amount of work and presents many pitfalls, not to mention the many bugs in the homebrew, extensive code. Things look better with a library like LittlevGL by Gábor Kiss-Vámosi [1] because it comes with a very project-friendly MIT license. A GUI developed with it is well suited for touchscreens, but can also be operated with a mouse, keyboard or some buttons. The code runs on popular 16-, 32- and 64-bit microcontrollers. The minimum criteria to meet are 16 MHz, 64 kB Flash and 16 kB RAM. Thus, the library fits perfectly to small boards such as the ESP32 or ESP8266 and has now been included by Espressif in their IDF. Below you'll find support for getting started and putting together test hardware. LittlevGL also offers the opportunity to develop and test GUIs on the PC, which is not to be sniffed at. The code created on the PC for a GUI can be transferred to the target microcontroller without major adjustments.

Libraries & the ESP32

You learn most when you try something hands-on. Therefore, the use of the library within Elektor's weather station [2] is demonstrated here. The goal is a GUI suitable for touch operation. We will even realize a multi-page display for data. But this requires hardware.
An ESP32 module is easily available. You can use modules like an ESP32 PICO D4, an ESP32 DevKitC, or a derivate board. The display offers a selection between a serial interface and parallel control, which then uses almost all IOs of the ESP32. Since the price also plays a role, a common 3.5" LCD for the Raspberry Pi is used. Most cheap displays like the 3.5" JOY-iT are connected via SPI and work with 3.3-V levels on the signal lines. They are therefore perfectly suited for a pin-saving link to an EPS32 board. In addition, they already have an integrated touch controller that can be connected via SPI.

The SPI displays intended for RPi, however, are limited in terms of speed at which data can be transferred to the display. Anyone who has ever experienced such a display in action on an RPi already feels the slightly sluggish screen update.
Note: LittlevGL does not provide display drivers, only 'higher' functions for drawing objects. It is up to the creator to develop the appropriate hardware-related routines. But even that doesn't require reinventing the wheel, since most display controllers benefit from ready-made libraries. Here the Arduino TFT_eSPI [3] library is used, which also supports 3.5" displays for the RPi.

Hardware

The following hardware is required to replicate the project (**Figure 1**):

- ESP32 DevKitC 32D or ESP32 PICO kit V4
- 3.5" Touch-Display for Raspberry Pi from JOY-iT
- Small breadboards & jumpers

Software

The required software is also orderly. In addition to the compulsory Arduino IDE and board support for the ESP32, the Arduino versions of the LittlevGL and TFT_eSPI libraries are needed.
In order to install and manage both libraries comfortably, the following two address lines must be entered in the Arduino IDE under *Preferences Additional Boards Manager URLs*:

```
https://github.com/littlevgl/lv_arduino/library.json
https://github.com/Bodmer/TFT_eSPI/library.json
```

These allow searching and installing LittlevGL and TFT_eSPI via the Library Manager. Then check if the Arduino library folder contains 'TFT_eSPI' and 'LittlevGL'.
As already mentioned, both libraries are needed. LittlevGL takes care of the UI, i.e. the animation and arrangement of objects, the management of several scenes, and the rendering of the displayed graphics. The result is a bitmap. This data is then transferred from TFT_eSPI to the display hardware in a suitable way. These libraries therefore abstract from the actual display used.

More displays

TFT_eSPI not only supports SPI displays for the RPi, but others too based on the following controllers: ILI9341, ST7735, ILI9163, S6D02A1, HX8357D, ILI9481, ILI9486, ILI9488, ST7789, and R61581.
This supports many common colour displays. If a RAiO-based controller such as the RA8875 is used, the RA8875 library from Adafruit [4] can be used instead. Adjustments are necessary to connect LittlevGL. In addition to colour displays, monochrome types can also be used in conjunction with the *u8g2 library* [5]. However, the following text refers to the common 3.5" SPI LCDs for RPi.

A homebrew driver

If you are using a display with a controller lacking Arduino driver support, you need to know what functions need to be provided. This knowledge is also helpful for porting and linking. Basically, it is sufficient to write your own driver capable of setting individual pixels on the display in a defined colour. LittlvGL expects a function like this:

```
/*************************************
* Function    : disp_flush_data
* Description : Sends pixels to the display
* Input       : lv_disp_drv_t *disp, int32_t x1,
   int32_t y1, int32_t x2, int32_t y2, const lv_
   colour_t *colour_array
* Output      : none
* Remarks     : none
*************************************/
void disp_flush_data(lv_disp_drv_t *disp,
 const lv_area_t *area, lv_colour_t *colour_p){
   /* Here: grab the pixel and send it to the display
   */

   /* Inform the library after job is done */
   lv_flush_ready(disp);
}
```

This function is passed to LittlevGL as a function pointer. For parameters it receives the start and end coordinates of the drawing area to be filled as well as a pointer to the image data. The actual setting of the pixels depends on the driver for the respective display. However, it can happen that the colours desired by LittlevGL have to be converted for the

display (e.g. from RGB to BGR). That's as far as the abilities of the drawing routine have to go. On the other hand, there are solutions for hardware acceleration, such as the DMA2D engine of some STM32 controllers.

Touchy

This should clarify how a graphic is displayed on the screen. What's missing is the processing of the data from the touch controller. For that, we have a LittlevGL function which reads and processes the coordinates of a possible touch depending on the device. RPi displays are usually equipped with an XPT2046 touch controller, which is connected to the SPI bus as an additional slave. Unfortunately, this cannot be read with a clock rate higher than 2.5 MHz. Since the display and its controller run at a clock rate of 20 MHz (at room temperature, up to 26 MHz), the bus clock rate needs adjusting during access and then get reset to the original clock rate. The TFT_eSPI library also helps here, because it not only offers XPT2064 support, but also automatically takes care of the necessary clock switching. If you do not use touch operation, you can also operate the UI through a mouse, keyboard, rotary encoders, or buttons. Of course, these input methods also require suitable drivers. They have to be registered in LittlevGL.

Image composition

First, a word about the strengths of LittlevGL: it is advantageous that the source code is also available. This facilitates the debugging of your own code. In addition, the library is well documented and developed out actively. Even beginners can use the examples to quickly achieve their first sense of achievement and design interfaces. From simple labels, buttons and tables to dropdown lists and gauges — a wide range of controls is provided. Windows and note boxes as well as themes for the appearance are also provided in order to be able to adapt the GUI even better to one's needs. The documentation describes all elements in detail. At [1], functions of the library and their interaction are demonstrated. LittlevGL does not yet offer any basic functions for setting pixels or drawing lines. That's due to the type of image generation: If an element changes, the new screen area to be drawn can be determined and the buffer prepared in the internal RAM. The image is then sent to the display. Consequently, not just a line needs to be present as an object, but even individual pixels. This restriction makes it much easier to redraw areas on the screen.

The technical details of image generation and display are now available: If you want flicker-free and interference-free animations or screen updates, you could first prepare the complete image in the RAM of the controller and then transfer this data to the display. In this case, we would be dealing with 307 kB of data. You could also transfer all elements directly to the display and use less RAM. The latter complicates a flicker-free display and hinders effects like antialiasing, transparency and shadows. A compromise is to mirror a part of the screen in RAM. With only a little more than 10% of the memory required for the entire image, you get all the features mentioned. For a display with 480 × 320 pixels at 16-bit colour depth that would be 'only' 30.7 kB of RAM — for an ESP32 that's a lot, but it can handle it. In the current version 6 of the library, the memory area is not communicated by a #define, but instead has to be provided by code. This procedure is especially useful if additional external RAM is available and is to be used.

Table 1. ESP32 connections

Function	Display Pin	ESP32 Pin	Note
MISO	21	19	
MOSI	19	23	
SCK	23	18	
DC	18	02	
CS	24	05	
RST	22	EN	saves 1 GPIO Pin
T_CS	26	04	
VCC (5 V)	02	5 V	
GND	14 / 25	GND	
T_IRQ (optional)	11	34	ESP32 Pin input only

In our demo we limit ourselves to a static allocation of an area in the ESP32 memory, which keeps the code simple:

```
//Memory for the displaybuffer
static lv_disp_buf_t disp_buf;
static lv_colour_t buf[LV_HOR_RES_MAX * 10];
```

This memory is allocated to the display in the function `hal_init()` with the following line:

```
lv_disp_buf_init(&disp_buf, buf, NULL, LV_HOR_RES_MAX
   * 10);
```

With other microcontrollers the possibilities and priorities need to be considered, because many devices either have considerably less RAM available, or would have to address external RAM through a bag of tricks. In addition to the available RAM, other aspects such as the available computing power or multi-threading are also relevant. LittlevGL unfortunately does not allow multi-threading, so all access must be from the same thread as the `lv_task_handler()` function. The required computing power depends on how much interaction and drawing takes place on the display and whether and how animations are used. Thanks to its two processor cores, an ESP32 has enough computing power for a GUI.

Experiments

If you want to experiment now, be wary of the pitfalls. A setup example is described below for easy-peasy commissioning. An ESP32 D4 PICO board occasionally has slight startup problems due to the additional load of a display. An additional 10-µF capacitor between 3.3 V and GND delays booting until the voltages are within a defined range.

The connection of a display to an ESP32 board should follow the allocation in **Table 1**. This would prepare the hardware. Now the configuration and the software testing follow. First, we deal with the library TFT_eSPI as the actual display driver and then with the configuration of LittlevGL.

Concerning the display driver, to be able to adapt the file `User_Setup.h` to the display used, you have to look for the folder TFT_eSPI in the library directory of the Arduino IDE. The following `#defines` have to be present for the display used:

```
#define RPI_ILI9486_DRIVER // max. 20 MHz SPI
#define TFT_MISO 19
#define TFT_MOSI 23
#define TFT_SCLK 18
#define TFT_CS   05 // Chip select control
#define TFT_DC   02 // Data Command control
#define TFT_RST  -1 // set TFT_RST to -1 if display
   RESET is connected to ESP32 RST
#define TOUCH_CS 04 // Chip Select (T_CS) of touch
   screen
#define SPI_FREQUENCY  20000000

// An XPT2046 requires a lower SPI clock rate of
   2.5 MHz:
#define SPI_TOUCH_FREQUENCY 2500000
```

Meaning: the GPIOs used are defined and a 20-MHz SPI clock serves as safe initial value for the display. 2.5 MHz is suitable for the touch controller. For testing we use an example (selection: TFT_eSPI 480x320 Rainbow480), which outputs the rainbow colours once. If everything is compiled and connected correctly, the display should look like **Figure 2**. At this point, we have the hardware basically ready for use.

The next step is linking LittlevGL to the display driver and creating its own HMI (Human Machine Interface). In order to use LittlevGl, the configuration should be adapted first. To do this, look for the littlevGL folder in the Arduino library directory. In the file `lv_config.h`, adaptations to the display used are made and the available elements of the library are set. At the start of the file you will find the settings for the memory management. The line:

```
#define LV_MEM_SIZE (64U * 1024U)
```

defines the RAM reserved for GUI objects. At the specified value of 64 kB, the linker will later complain that such an amount cannot be provided. With static reservation (at compile time), the non-continuous memory area of the ESP32 becomes noticeable. You could now reserve blocks at runtime through malloc and free. Since this measure holds other dangers, the matter is approached differently. Change the line like so:

```
#define LV_MEM_SIZE (24U * 1024U)
```

This is enough for our first steps.
The display has a resolution of 480×320 pixels. The corresponding #defines are:

```
/* Horizontal and vertical resolution of the library */
#define LV_HOR_RES_MAX (480)
#define LV_VER_RES_MAX (320)
```

The resolution in DPI (dots per inch) is calculated according to the following formula:

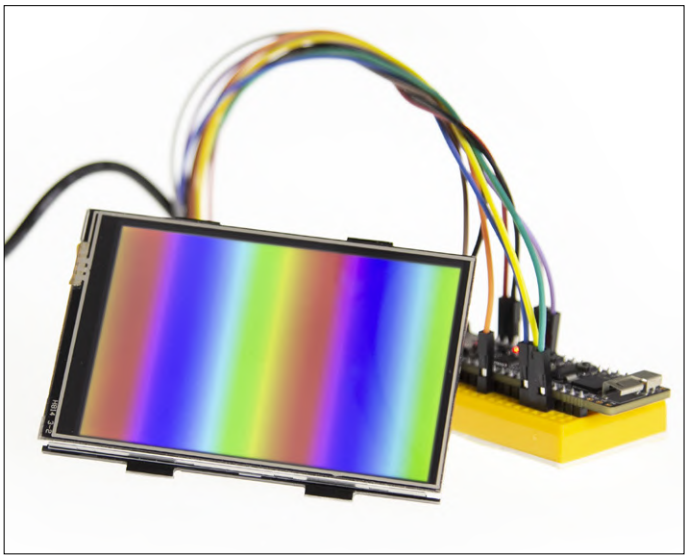


Figure 2: First test: the display shows rainbow colours.

$$DPI = \frac{\sqrt[2]{(horizontal\ resolution)^2 + (vertical\ resolution)^2}}{screen\ diagonal\ in\ inch}$$

Substituting:

$$DPI = \frac{\sqrt[2]{(480)^2 + (320)^2}}{3.5} = 164.83$$

With integers being output:

```
#define LV_DPI 164
```

So much for the basic setting. For first tests, the remaining settings remain untouched; the changes are saved. In the Arduino IDE you can now select the example *ESP32_TFT_eSPI* under LittlevGL and upload it to the ESP32 board. If everything is configured correctly, “Hello Arduino!” should show up on a white background on the display.
So, the driver and LittlvGL cooperate well. However, the touch controller of the display has not yet been read and this data has not been transferred to the library. The following are the basic code parts allowing you to can create a basic framework for your own application. For this purpose, the example *ESP32_TFT_eSPI* from the LittlevGL library, which just got loaded into the ESP32, is examined more closely.
In the `setup()` function, after the initialization of the library in line 63 using `lv_init()` and the TFT in lines 69 and 70 with `tft.begin()` and `tft.setRotation(1)`, lines 73 and 74 with the initialization of the structure `lv_disp_drv_t` follow. This structure is given a function pointer for writing to the display, and is subsequently registered in the library.
A similar procedure can be found in the dummy touch driver in lines 80–84. The last step is to provide a time base for the library using ‘Tickers’. Here, a function is called in the specified interval of 20 ms. Each time, a timer is increased by 20 ms.

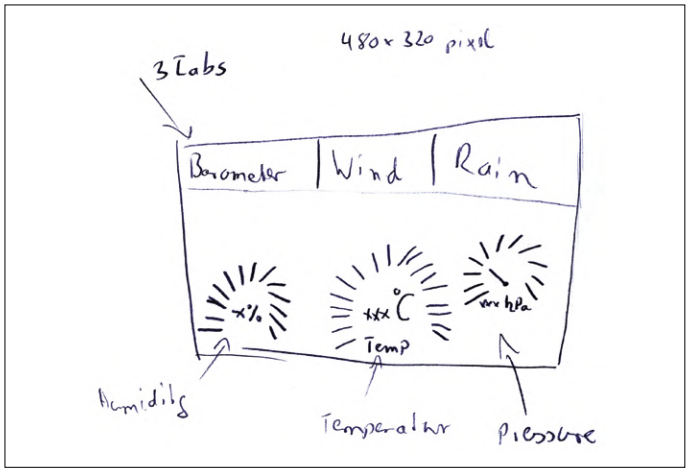


Figure 3: Example of a manual sketch for the display of air pressure, temperature and humidity using three tabs.

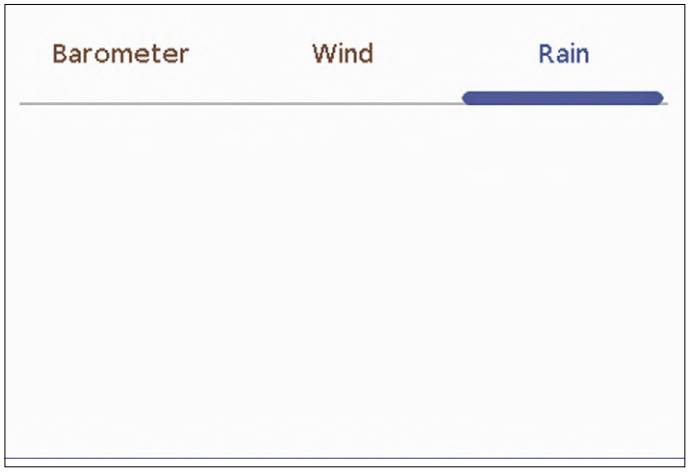


Figure 4: Screen with three tabs without further content.

Finally, a button will be created and assigned the text “Hello Arduino!” (lines 90–92).
Within the loop function, `lv_task_handler()` is now called so that the GUI can react to input or redraw the screen.

To avoid having to start from scratch with every project, the author has created a basic project in which the settings for the display of JOY-iT and its touch controller as well as the initialization of the components are made. In line 139 of the sketch, the orientation of the display is set with `tft.setRotation(3)`. The image is thus rotated 270° from the starting position. If another display needs e.g. a rotation of 180°, the parameter should be set to 1.

GUI creation

With this basic framework, you can start to create your own GUI. This can be done directly on the ESP32 hardware, but compiling, uploading and testing takes time. The alternative is a PC simulator. Its installation requires familiarity with Eclipse. The installation is described in [6]. It is a bit more difficult in Windows than in Linux or OS X. In the simulator, you can now try out the first steps without having to upload modified code to the hardware every time.
The first step is the design of the surface, which is best done using pen and paper (or tablet and pen) first, because sketches should be made before the first lines of code. **Figure 3** shows an example of such a hand-drawn sketch. This makes it clear where which object gets placed, and which objects are used to navigate.
Since we’re talking about a weather station, a simple surface with three tabs is created for the display of the weather data. To keep the Arduino sketch clear, the functions and the creation of the GUI elements are stored in a separate file.
First, the scene is prepared and a tabview element is created, to which three tabs are added: Barometer, Wind and Rain. The following code is responsible for the preparation of the scene:

```
lv_theme_set_current(th);

/* Next step: create a screen */
lv_obj_t * scr = lv_cont_create(NULL, NULL);
lv_scr_load(scr);
```

First, the theme is loaded, which was passed as a function parameter. An empty scene is then created and loaded. The Tabview element is assigned the complete screen size. The screenshot in **Figure 4** shows the three empty tabs with the font settings specified by the theme. If you click on the name of the corresponding tab, a change of the tabs is indicated by the blue marker. Since the tabs are empty, you can’t see more than this.
The first three tabs are created with the following five lines of code.

```
/* And now populate the four tabs */
lv_obj_t * tv = lv_tabview_create(scr, NULL);
lv_obj_set_size(tv, LV_HOR_RES_MAX, LV_VER_RES_MAX);
lv_obj_t * tab0 = lv_tabview_add_tab(tv, "Barometer");
lv_obj_t * tab1 = lv_tabview_add_tab(tv, "Wind");
lv_obj_t * tab2 = lv_tabview_add_tab(tv, "Rain");
```

At first, they have no content yet and are therefore provided with titles afterwards.

Weather display

Let’s start with the barometer: three values for humidity, temperature and air pressure should be displayed here. For humidity and temperature, `lv_lmeter` and `label` are used, indicating the value and name of the measured variable. For the humidity, `lv_gauge` is used. Conveniently, you can still influence the look of the elements at runtime by styles and so individualize each element.
The Autofit function of the library must be taken into account when arranging elements. Consequently, you should arrange the elements appropriately or disable Autofit. Elements can be positioned from several original coordinates — an overview can be found at [7]. The individual elements can have parents, i.e. objects on which their position depends. This results in elegant dependencies of the positions, which also realigns all children during a parent shift. In the code, the created elements cannot be accessed directly later. For `LMeter` and `Gauges` we therefore

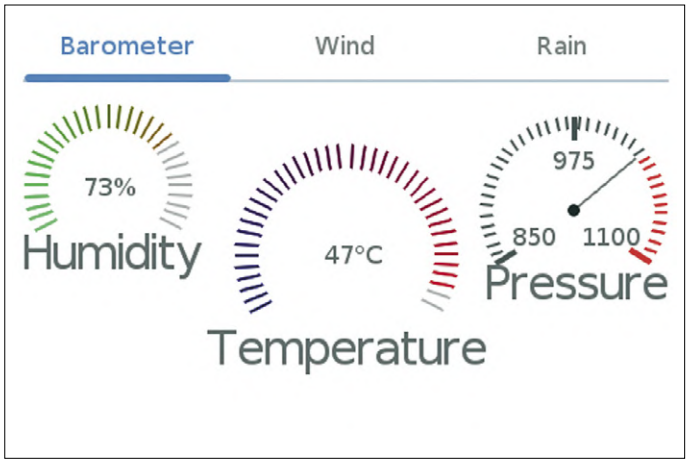


Figure 5: Screenshot of the first tab with barometer values.

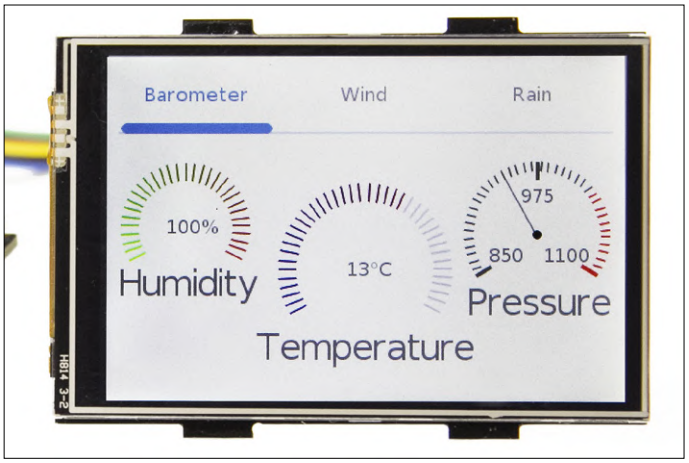


Figure 6: Barometer on real display.

use pointers that can be accessed globally. In the example code

```
lv_obj_t* humidity_lmeter
lv_obj_t* humidity_label
lv_obj_t* temp_lmeter
lv_obj_t* temp_label
lv_obj_t* air_pressure_gauge
```

it is noticeable that functions like `lv_ L-meter` create only return pointers. The question now is where memory is reserved. This is buried a bit deeper in the library. The expression:

```
# define LV_MEM_SIZE (24U * 1024U)
```

creates a fixed storage pool for the graphic elements. Each time a Create function is called, some memory is taken from the pool and reserved for the graphic object. The result of the operation is a pointer to the memory address, which is used to change the properties of the object. If the memory runs out at some point — this can happen with dynamic surfaces — an error is triggered in the library and an endless loop is created. The ESP32 stops completely.

The pointers are only valid in the function you are currently in. If you want to access an element later without detours, the pointers have to be stored outside the function. For the sake of simplicity, we use a few global variables. In serious applications, however, it is not advisable to use global variables.

Through Pointer, you can then write new values to the displays. One example is the `UpdateTemperature` function. The display element `Lmeter` expects a value between 0 and 100, but the size has a value range of ±50°. The temperature must therefore be offset by 50. 0° then corresponds to a `Lmeter` value of 50. In addition, the current temperature is displayed as text. This is done by `snprintf` and a small local buffer, which is written as new text into the text field. If the text length changes, the text is not automatically realigned. The alignment must be done again after setting the text. To do this, `lv_obj_align` is called again with the parameters for the label. Humidity and air pressure are treated very similarly. **Figure 5** shows a screenshot of the finished tab and **Figure 6** shows what it 'really' looks on the LCD.

The first tab is now filled with content. The second tab is similar, but the wind direction display as a compass dial requires a little more effort. Here a `Gauge` acts as Parent for four labels. In the code, a `Gauge` with 0–359° is created first. This is followed by four labels, which are forwarded to the compass as a Parent. The labels are defined in relation to the centre of the compass. This results in the four points of the compass. The pointer points in the direction from which the wind comes. With the `Gauge`, 0° does not result from the value 0, but from 180.

Web Links

[1] LittlevGL: https://github.com/littlevgl/lv_arduino

[2] Weather station with ESP32, ElektorLabs Magazine 1/2019: www.elektormagazine.com/180468-02

[3] TFT_eSPI: https://github.com/Bodmer/TFT_eSPI

[4] RA8875 Library: https://github.com/adafruit/Adafruit_RA8875

[5] u8g2 Library: https://github.com/olikraus/u8g2

[6] PC Simulator: https://docs.littlevgl.com/#PC-simulator

[7] Object Positioning: https://docs.littlevgl.com/#Base-object

[8] Monster LED Clock, ElektorLabs Magazine 4/2019: www.elektormagazine.com/180254-02

[9] Article support page: www.elektormagazine.com/190295-02

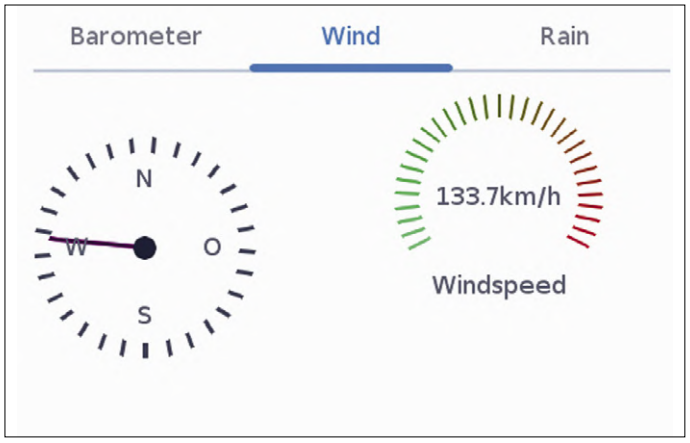


Figure 7: Tab with wind direction and speed.

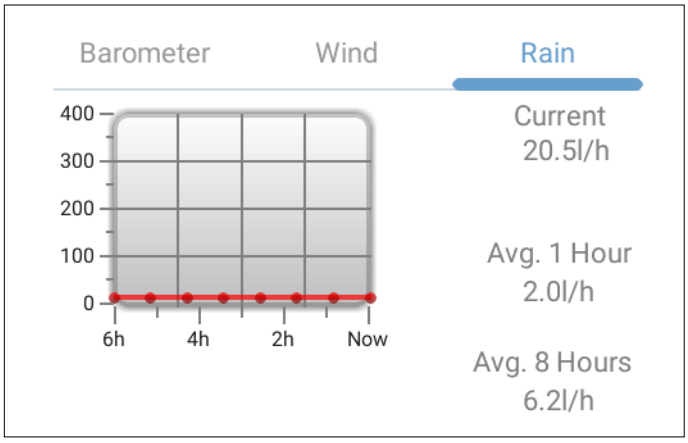
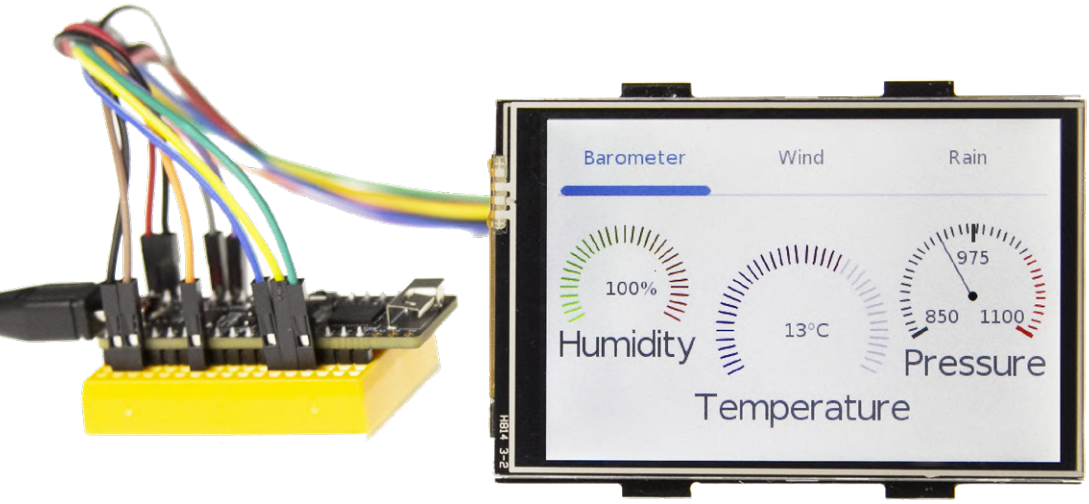


Figure 8: Screenshot of the tab with precipitation history and values.



For the display of the wind speed, again an `Lmeter` with similar structure is used as with the barometer. Notice that processes are repeated for the creation of the elements. First, a style is created for the object, then the object is created, and finally its properties get assigned to it. The finished result can be seen in the screenshot of **Figure 7**.

In the case of rain or precipitation, the values are displayed differently. Here the values are represented in text form as well as in a diagram, from which the temporal process results. The texts are realized as described before: First, styles and objects are created — then the values are assigned. For the course of the rain volume a line diagram is suitable, which requires no more tricks to label the axes since version 6. To update the values, it is not necessary to move each data point individually, because `lv_chart_set_next` does this. A new value is passed to the diagram once per hour. The update of the precipitation volumes is done just as with other texts as well as by an internal function. **Figure 8** shows a screenshot with pseudo data of the progression and the precipitation values.

For the data connection of the display, code from the Monster LED Clock [8] project got recycled, since data already coming from a broker via MQTT is processed here. The code expects the broker to send a JSON string containing the values for humidity, temperature, air pressure, wind direction, wind speed and precipitation. When new data comes from the broker, it is entered into the corresponding elements. However, you need to make sure that this does not happen through different threads. With the configuration, there are no big differences to the LED Clock project [8] except for the missing time setting. WLAN and MQTT settings were simply adopted; only the weather station and the display have to be set to the same topic. From then on, the values are displayed directly on the screen. An exception at this point is the rain quantity, because here only the current rain quantity of the weather station is output. The calculation of hourly values and the trend are unfortunately not yet implemented with the weather station. As soon as this has materialized, however, these values are also updated on the display.

Conclusion

With this example, some basic functions of LittlevGL were demonstrated hands-on. The corresponding code for this demonstration can be downloaded free of charge from the ElektorLabs Magazine website [9]. As already mentioned, LittlevGL provides much more functions and animations as well as design possibilities with tables, lists and dropdown menus. If you like the simplicity of using this library, you might want to try LittlevGL in your own projects. After all, an EPS32 module in combination with a touch display is a platform that can be used quite universally and offers a lot of performance for little money. ⏮

190295-02

@ WWW.ELEKTOR.COM

→ SJOY-iT 3.5” Touch Display for Raspberry Pi
www.elektor.com/18145

→ ESP32 Pico Kit V4
www.elektor.com/18423

→ Mini Breadboards & Jumper Wires
www.elektor.com/18430

OUR RND RANGE SUITABLE FOR EVEN THE TIGHTEST OF BUDGETS.

The best part of your project: www.reichelt.com/rnd

High quality, low prices.

RND is a reichelt brand that offers customers easy access to affordably priced components, without compromising on quality. The range comprises more than 8000 products covering components, automation, cables, housings, tools and laboratory equipment:

RND lab · RND power · RND components · RND connect · RND cable

Power soldering iron with 150 W

RND lab

Automatically sleep after 10 minutes

For high safety, the soldering iron automatically switches to standby mode after 10 minutes of inactivity and switches off completely after 20 minutes.

- Temperature controlled - Adjustment at the push of a button
- Temperature range: 250 - 520 °C
- Ergnomic handle with integrated LCD for temperature display



BESTSELLER ~~£ 43,39~~

Order no.: **RND 560-00216**

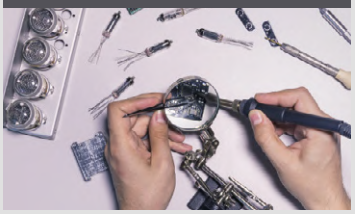
£ 34,70 (£ 28,92)

150 WATT

THEME SPECIAL:
Things to know about soldering

Soldering is an important part of working with and on electronic components. Successful projects require on the one hand the right equipment, on the other hand the right soldering technology.

Inform now
▶ www.rch.lt/MG282e



LED illuminated shop magnifier

5-step dimmable

- Glass 3 dioptre lens
- Colour temperature 6000 - 6500 K
- 950 lm luminous flux



A+ (A++ - E)

Order no.:
RND 550-00122 **£ 73,79** (£ 61,49)

RND lab

Soldering tin with copper portion

100 g, Sn99.3/Cu0.7

- Proportion of flux 2.5 %
- Filler wire diameter 1 mm
- Melting point +217 °C



PRICE TIP

Order no.:
RND 560-00169 **£ 6,34** (£ 5,28)

RND lab

Payment Methods: VISA · MasterCard · Maestro · PayPal

DAILY PRICES! Price as of: 25. 9. 2019

- Top price/performance ratio
- Over 110,000 selected products
- Reliable delivery - from Germany all over the world

Order Hotline: +44 203 808 95 25

www.reichelt.com

reichelt elektronik – The best part of your project

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

“Not Just Any Guy's Project”

Interview with Gábor Kiss-Vámosi, the developer of LittlevGL

Questions by **Mathias Claußen** and **Jens Nickel**

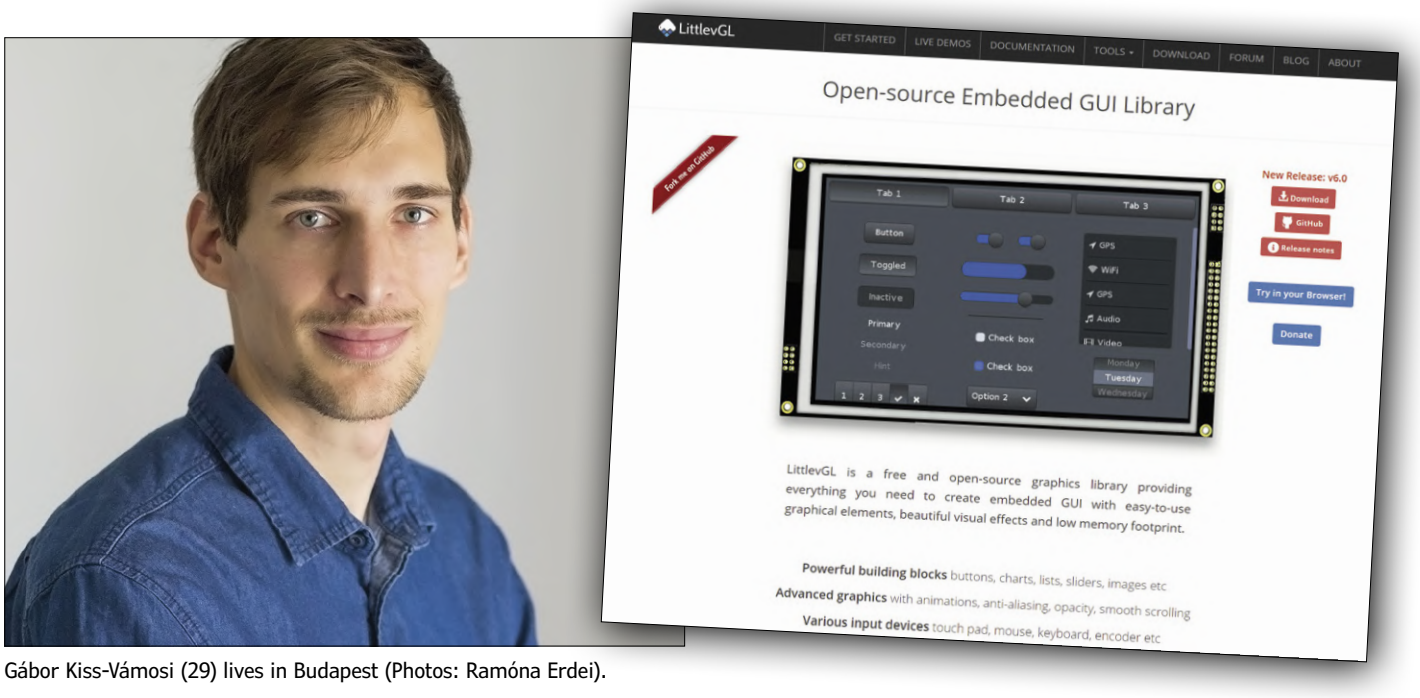
Nowadays, software development cannot work without open-source libraries and frameworks. Some of these libraries are born when young private software enthusiasts are developing some code they need regularly for themselves. Then the project is getting bigger and more professional. One example is the LittlevGL library by Gábor Kiss-Vámosi, which makes graphical user interfaces with touch accessible even for 16-bit-Microcontrollers with small RAM (see extra article in this issue). In this interview, Gábor talks about the background, some interesting details and the future of his popular library.

Elektor: Gábor, why should we use your GUI library since there are some out there already that are open source and free to use for non-commercial purpose like µGFX?
Gábor: µGFX and LittlevGL have some shared advantages, like open source and platform independence, but LittlevGL can handle smooth animations, anti-aliasing, opacity, and shadow drawing without double buffering. This way only one frame buffer is required in the display controller chip or in the MCU and a small graphics RAM for LittlevGL. Another advantage is LittlevGL has built-in support for navigation and control with a keyboard or even an encoder.
Finally, LittlevGL has MicroPython support. So you can write Python3 code to create a user interface. Because of that, you can change the UI real-time without rebuilding and flashing the MCU.

Elektor: What is your background, your profession? How comes that you invest so much time in a software project?
Gábor: I’m an electrical engineer and I’m now working on a hardware security-related project in my professional job.
During university, one of my friends and me developed a lot of hobby things in our free time, like amplifiers, instruments, and other gadgets. We were wondering if it would be so good to equip this stuff with a nice graphics display instead of 7-segment or 2 x 16 LCD displays. I said: “Okay, let’s buy a TFT display. I’ll try to draw something on it. If I can set a pixel, the rest should be easy”. Finally, it took just two hours to set a pixel, but I have been dealing with the “easy rest” in the last eight years!
In general, I love to learn and figure out new things, so it’s like a hobby. When the project became public, fortunately, I got a lot of feedback from other - more experienced - developers. They suggested how could it be done better or what they miss. And I can’t say no to the challenge. I feel a personal responsibility for my project, because people use it and trust in it.

Elektor: How much time do you actually invest on coding? How much time do you spend on other project tasks?
Gábor: It was easier a few years ago, but now I have a family, so I need to find the balance between the two. Usually, I get up early when my wife and child are still sleeping to answer emails and questions and to work on the current new features. In the afternoon, if I have free time, I still try to do something. If I count the hours, it is a part-time job besides my professional job. It’s quite challenging (but not impossible) to have time for work, LittlevGL, family, friends and other free time activities. Earlier, when I published LittlevGL, I needed to learn a lot of non-programming related things such as marketing, Search Engine Optimisation, web development, some graphic design and so on. I remember that it took months to read SEO tips and tricks, and I analysed websites all day long. But it was worth it, because if you now search for “embedded GUI”, probably LittlevGL will be the first in Google.

Elektor: Do you get help from the community or companies in the background, for coding and other tasks?
Gábor: Fortunately, as time passed, more and more contributors have joined and supported the project in various ways. Some of them share their personal improvement like “Hey, I’ve added a placeholder feature to the Text area object. Are you interested?” Some contributors add huge improvements; the MicroPython binding is from “amirgon” and the new font converter is from “puzrin”. Others help with answering users’



Gábor Kiss-Vámosi (29) lives in Budapest (Photos: Ramóna Erdei).

questions and with bugfixes. For example, “embeddedt” spends hours every day to support users. What they do is very important and valuable. LittlevGL couldn’t be here without them!

Elektor: Did you get help from the community to do the Arduino porting ?
Gábor: Actually, the Arduino port was a tricky part. The library’s structure needed to be changed to work with Arduino’s build system. I’ve done the initial changes myself but a few months ago, “Pablo2048” reshaped it again to release an Arduino package and he also helps to answer the related questions. He had way more experience with Arduino than I had**.**

Elektor: Users can donate money to your project. Are the donations a big amount of support for you? Or just a small one?
Gábor: The donations are enough to cover the expenses of the project such as web server, buying development boards and so on, but I can’t make a living from it.

Elektor: How do you get new ideas for the library? Do you get feedback from developers?
Gábor: Sometimes users explicitly ask a feature. In addition, I handle recurring questions as something needs to be improved either in the library or in the documentation. For example, v6.0 was a conceptual change in the library to mainly solve frequent questions and requests.
Besides users’ suggestions, I try to follow modern trends in graphics design and embedded UIs and provide a solution for them.

I also use the library myself. I made several projects for companies as a freelancer, but I also use it in my professional job right now.

Elektor: Do you also go to shows and events? Do you have the chance to talk to users of your library personally?
Gábor: To be honest, I miss the personal experience with the users. It’d be so good to meet them personally, but they are all around the world so it’s not that easy.
Some people already offered to visit them. They even want to host us if we visit their country or city. In fact, we’ve organised our next year holiday to a LittlevGL user. And I’m so happy about this opportunity.
LittlevGL’s background will be changed to become a really professional software and not just any guy’s project. As part of it, we planned to participate in conferences and hold webinars and seminars.

Elektor: To use LittlevGL, 16, 32 or 64 bit MCUs are required. This means no support for the 8-bit-MCU family?
Gábor: Theoretically there is no problem with 8-bit MCUs but usually, they have so few RAM that they can’t run LittlevGL. At least 16 kB RAM is required to run LittlevGL which can be tight even on 16-bit MCUs.

Elektor: Can you tell us a bit about the technical idea behind the library? From the first samples we can see that the UI is working with Screens / Scenes.
Gábor: The basic idea is that you create objects (or in other words widgets) such as buttons, labels, charts, sliders, etc.,

Web Tools for a Software Project

Coding is one side of an open source software project, but you also need documentation and a way to support the users. This is what Gábor Kiss-Vámosi tells about the website and other tools to support his library:

As an electrical engineer, I didn't know much about web development. First, I've tried WordPress but when I tried to customise it, I've found that it can't give the freedom I need. Later I've found Bootstrap, and I could create the website from scratch. It was pretty easy to deal with Bootstrap even as a novice web developer because it has good looking elements and built-in responsiveness.

Later, the "side topics" are moved to separate subdomains and now they work with different engines.

The blog is hosted on GitHub where the posts are written in markdown, and an engine called Jekyll automatically compiles a static website from the markdown files. This way, anybody can easily add posts to the blog, just a pull request needs to be sent.

The documentation is also hosted on GitHub but it's compiled offline with Sphinx. The docs can be translated by the users on an online platform, called Zanata.

The forum has a Discourse engine which runs on a DigitalOcean Virtual Server. DigitalOcean has a sponsorship program for open source projects. I applied and got a yearly sponsorship which can be spent as I want. So, I started a VPS to host the forum. "seyyah" from GitHub helped a lot with the VPS configuration.

My goal was to build a community where people can share their knowledge, speak out their projects and experiences. That's the reason why LittlevGL has an open blog where anybody can write posts. And there is a "My projects" category on the Forum to share if you created something interesting.

and you just set their properties such as size, position, styles, value or state and call-backs to notify the user if something happened (e.g. a button is clicked). This way, all the drawings and other underlying things are managed by LittlevGL under the hood and the user needs to deal only with the high-level things. The object can be created and deleted in real time. This way, you can optimise memory usage to keep only the required object alive. For example, if you need a message box, just create it and delete it when the user clicks the "Ok" button. The message box only consumed memory while it was shown. The objects have a parent-children hierarchy. For example, you can create a container (parent) and add three buttons (children) to it. If you move the container, the buttons will move with it, if you delete the container, the buttons will be deleted too. Another interesting concept of LittlevGL is it realises a limited kind of object orientation. Every object is derived from the base object which has only the most common properties such as position, size, parent, etc. This mechanism is quite common in C++ but unique in C.

Elektor: We can use various prefab Widgets for the UI, if we need something that is not already there, is there a Guide helping to create these?

Gábor: All the widgets work the same way. They have some custom data, a design function which draws the widget and signal function which handles the low-level internal events. Basically, if you check the code of widget, you can create your own in the same manner. There are template c and h files to make it easier to get started with your own widget.

Elektor: Is the RAM statically allocated for the UI Elements? Or do we have a sort of memory pool where we can run out of memory?

Gábor: LittlevGL has its own memory manager which dynamically allocates memory for the widgets and other related things. It allocates data into a big array whose size can be adjusted or even placed to external memory. Unlike the standard "malloc" and "free", LittlevGL's memory manager provides monitoring to see the usage and fragmentation of the memory.
So yes, you can run out of memory, but you can monitor the memory usage and fine-tune the size of memory reserved for LittlevGL. If you run out of memory, you will get an error log message with a line number and the program will halt there. This way, you can easily debug where the problem happened.

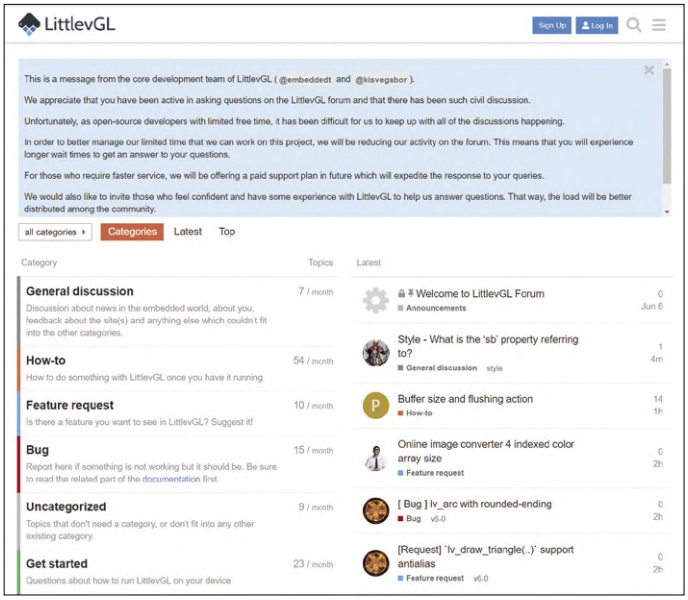
Elektor: How is drawing managed, are there some tricks involved to reduce multiple drawing over overlapping objects?

Gábor: A lot of tricks are involved to optimise drawing. On the highest level of drawing, LittlevGL accumulates which areas need to be redrawn (e.g. if a button is pressed) and in every few milliseconds (e.g. 30), it redraws those areas. Before redrawing, it searches from the background which widget is the first which covers the area to redraw. Therefore, if you change the text on the button, only the button and the label will be redrawn but the background below the button won't. But if you change the position of the button, the background, the button, and label also need to be drawn. But these are quite trivial optimisations.



Developing starts early in the morning.

The really interesting thing in LittlevGL is it doesn't draw directly to display (like a lot of simple GUI libraries) nor makes double buffering (like the advanced GUI libraries) but performs a tiled drawing. It uses a smaller (typically 1/10 screen sized) memory and draws there first. When the drawing is ready, the buffer with the ready image is flushed to the display. As it's a buffered drawing, there is no flickering while drawing, which allows smooth animations.
In the latest version of the library (will be published this year), the drawing engine is completely rewritten. Besides making the drawing engine more flexible and extendible and improving the quality in general (e.g. better anti-aliasing and shadow drawing), the main goal was to support masking. With masking the pixels, out of rounded corners can be clipped or texts can be written with an image background or a gradient colour. But some new features are also added like horizontal gradient, shadow offset and additive and subtractive blending modes.



User support is a big part of the work. On the forum, experienced users of the library are also answering questions.

Elektor: Fonts can be converted online at your page; are you offering an easy to use offline converter for fonts? What about symbols?
Gábor: The font converter is written in Node.js, so it can run online and offline too. We don't have a desktop font converter application yet, but you can clone the converter from GitHub and use it from the command line.
Symbols can also come from fonts. FontAwesome is a well-known very popular symbol font from which some symbols are included in LittlevGL by default. However, when you create your own font, you can pick multiple font files to merge into a single C file. For example, given ranges from Arial and some symbols from FontAwesome.

Elektor: The library is not thread-safe, are there plans to add RTOS support to your library?
Gábor: That's true, the library is really not thread-safe by default, however, it's easy to make it thread-safe. All you need to do is to take a mutex when you call LittlevGL related functions and release it when you are ready.

Elektor: LittlevGL has a nice Eclipse-based Environment for fast UI development without the hassle to flash the code to real Hardware. In the future, will there be a kind of GUI Designer, accelerating the Development phase even more?
Gábor: Absolutely yes. It was already requested several times. I was very happy when I saw that some users have started to spontaneously work on GUI designers. There are three independent developments and all of them made great progress so far. It is possible that one of them will become the official GUI designer of LittlevGL.

Elektor: LittlevGL is currently licensed under MIT License, which allows an easy integration in commercial products. From the last online survey you did, there were questions about commercial licensing options. What will change in the future?

Gábor: LittlevGL popularity is rapidly increasing. A very large company has shown interest in LittlevGL, but it was very difficult to negotiate as an individual. The next reasonable step is to create a company which can represent the interest of LittlevGL better than as an individual.
As the community is growing, it's more difficult to deal with the development, support, and marketing as a free time activity. The goal is to find a business model for LittlevGL which can provide a salary for some people who work on LittlevGL in full-time, but which will be still acceptable for the users too. Paid support with fast bug fixes and dedicated professional help, and some kind of licensing is also considered. We are still trying to find the right business model.

Elektor: Do you have any advice for other young people developing interesting open source software projects?
Gábor: My advice is to have a plan and vision when you publish or start an open-source project. Is it just a small tool which doesn't need to be really maintained? Or something big which will require a lot of time for years? What's your goal with the project? Why do you want to do this? Just for fun? For fame? For experience? Just to help others? You should be able to answer these questions in advance and make decisions accordingly. You might become frustrated if things just happen to you and don't know how to decide. Just say no if something is not what you want and welcome opportunities which brings you closer to your goals. As the proverb says: "There is no wind that blows right for the sailor who doesn't know where the harbour is." ⏮

190353-01

Web Link

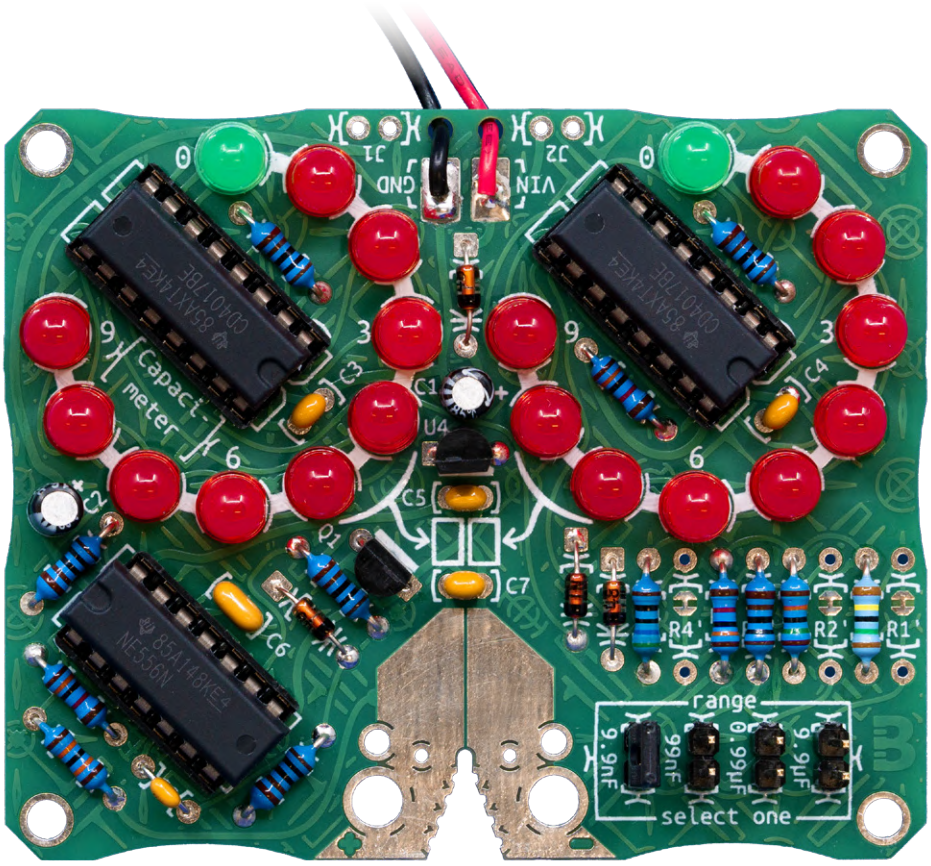
[1] LittlevGL Homepage: https://littlevgl.com/

eLABS PROJECT

Capaci-Meter

with two-digit Dekatron-style LED display

By **Jez Siddons** (projects@peakelec.co.uk)
PCB design by **Boldport** (saar@boldport.com)



The Capaci-Meter is based on a design that I created in 1984/1985 to submit for my UK O-level (now GCSE) technology course project. Instead of the 3-digit, 7-segment display used on the original version, I've opted for a more retro-style 'Dekatron'-type arrangement of LEDs to represent two digits.

With two digits for display, this is not designed to be an accurate measurement instrument. Instead it is aimed at establishing if a capacitor is close to the desired value, especially when it is difficult to read the values printed on the capacitor.

Quick Specifications

- Minimum measurable capacitance: 200 pF
- Maximum measurable capacitance: 9.9 µF
- Open circuit test voltage: 3.3 V
- Maximum allowed voltage on the test probes: -0.6 V to +5.6 V
- Input impedance: 130 kΩ
- Accuracy (unadjusted): ±5% of full scale
- Accuracy (range resistors fine-tuned): ±2% of full scale
- Current consumption: 15 mA
- Supply voltage: 7.5 V_{DC} to 15 V_{DC}

Measurement range and accuracy

The Capaci-Meter provides four measurement ranges (see **Table 1**). The minimum capacitance for each range is really a limit of the counter resolution (as well as jitter), hence the minimum is typically twice the measurement resolution. As with any measurement system, it's good to select a measurement range that gives you a reading that is as close as possible, but not over, the maximum full-scale value.

Typical accuracies are in the order of ±5%, but this can be improved for each of the four ranges by means of some

Table 1. The Capaci-meter provides four measurement ranges.
If a capacitance is too big for the currently selected range then the display will max out at '99', indicating a possible over-range. If possible, try a higher range to see if the capacitance was really over-range or perhaps sitting exactly on '99'.

Range	Capacitance Minimum	Capacitance Maximum	Capacitance Resolution
1	0.2 nF (200 pF)	9.9 nF	0.1 nF (100 pF)
2	2 nF	99 nF	1 nF
3	0.02 µF (20 nF)	0.99 µF (990 nF)	0.01 µF (10 nF)
4	0.2 µF (200 nF)	9.9 µF	0.1 µF (100 nF)

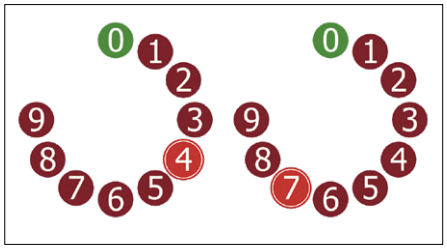


Figure 1. Here the value of '47' is being displayed. So, if the currently selected measurement range was 0.0 - 9.9 µF, then this display would represent 4.7 µF.

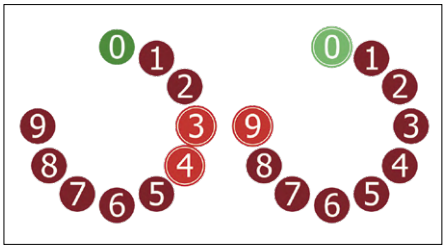


Figure 3. The display is jittering between '39' and '40'. It can appear confusing to start with, but just remember that jitter typically occurs between adjacent values (such as 39 and 40). This means that we can be sure that the display is not showing '30' and '49' because they are not adjacent values.

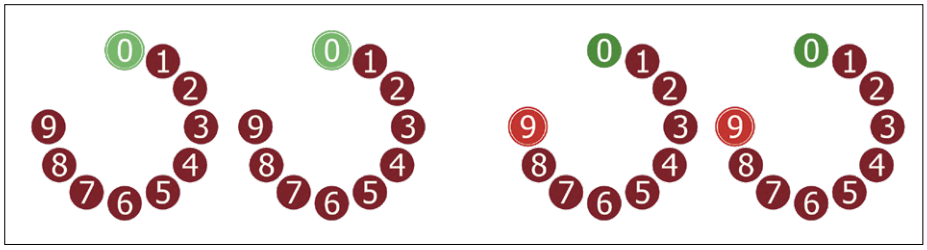


Figure 2. The display showing zero (left) and '99' (right). The display may show '99' if the value being measured is exactly full-scale, or perhaps over-range.

fine-tuning resistors. Bear in mind, however, that a 2-digit display can only provide a maximum theoretical accuracy of ±1% of full scale.
Furthermore, measurement quantization and jitter can add a further ±1 digits of error. So, you could realistically yield a best possible accuracy of ±2% at full scale if all four ranges are fine-tuned.

A Dekatron display

Two circles of ten LEDs make up the two-digit display of the Capaci-Meter. These circles are reminiscent of the Dekatrons used for counter/displays long before the advent of chips, LEDs and even transistors. Unlike the original Dekatrons, the LEDs here are arranged in the same way as the hours on a clock, so it becomes more intuitive to read the display (see **Figure 1** and **Figure 2**).

Jitter

Occasionally, it is possible that the display may be jittering between two values as shown in **Figure 3**. This can happen if the capacitance is just on the border between two measurement steps, or perhaps there is some electrical noise that causes some small variations. In most circumstances it is easy to see what value is being displayed even if there is some jitter. However, it can be trickier to interpret if the units are jittering between 9 and 0 which would also cause the tens to jitter.



The measurement principle

The Capaci-Meter repeatedly measures the time it takes for the capacitor-under-test (Cx) to charge up by a certain amount through a known resistor. The circuit comprises three main sections:

- Cx clock (where Cx is the capacitor under test): a simple 555-based pulse generator (U3A, half of a 556 dual-timer chip). The duration of the pulses are proportional to the capacitance Cx.
- Master clock: a square wave used by the counter for measuring time.

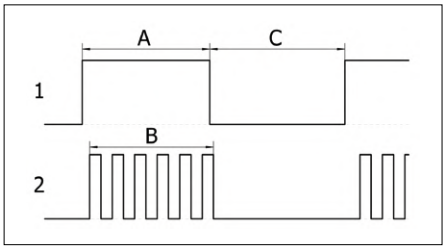


Figure 4. The duration of (A) depends on the value of Cx. The counter counts the pulses from the master clock (2) only when the Cx clock (1) is high (B). When it is low, the value is displayed (C).

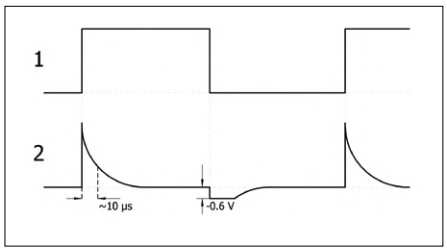


Figure 5. (1) is the Cx clock, (2) is the reset signal for the counters. Differentiator R12/C8 produces a positive pulse when it experiences a rising edge on its input and a negative pulse when it sees a falling edge. D4 limits the negative pulse to about -0.6 V to prevent damaging the counters. The values of R12 and C8 determine the rate of decay of the differentiated signal. Here we've chosen values that give about 63% decay in 10 µs, determined simply by T = R × C.

By changing the frequency of the square wave, we can change the effective measurement range of the whole instrument.

- Counter/Display: counts the rising edges coming out of the master clock. By doing so it can measure time. The counters also drive the LEDs which form the display.

The master clock generates pulses that are fed into the counter/display. When the Cx clock signal is high, the counter counts the pulses from the master clock. When the Cx clock signal is low, the master clock is disabled (**Figure 4**) and the counter stops. At the same time, the LEDs are switched on so we can see the counted value.

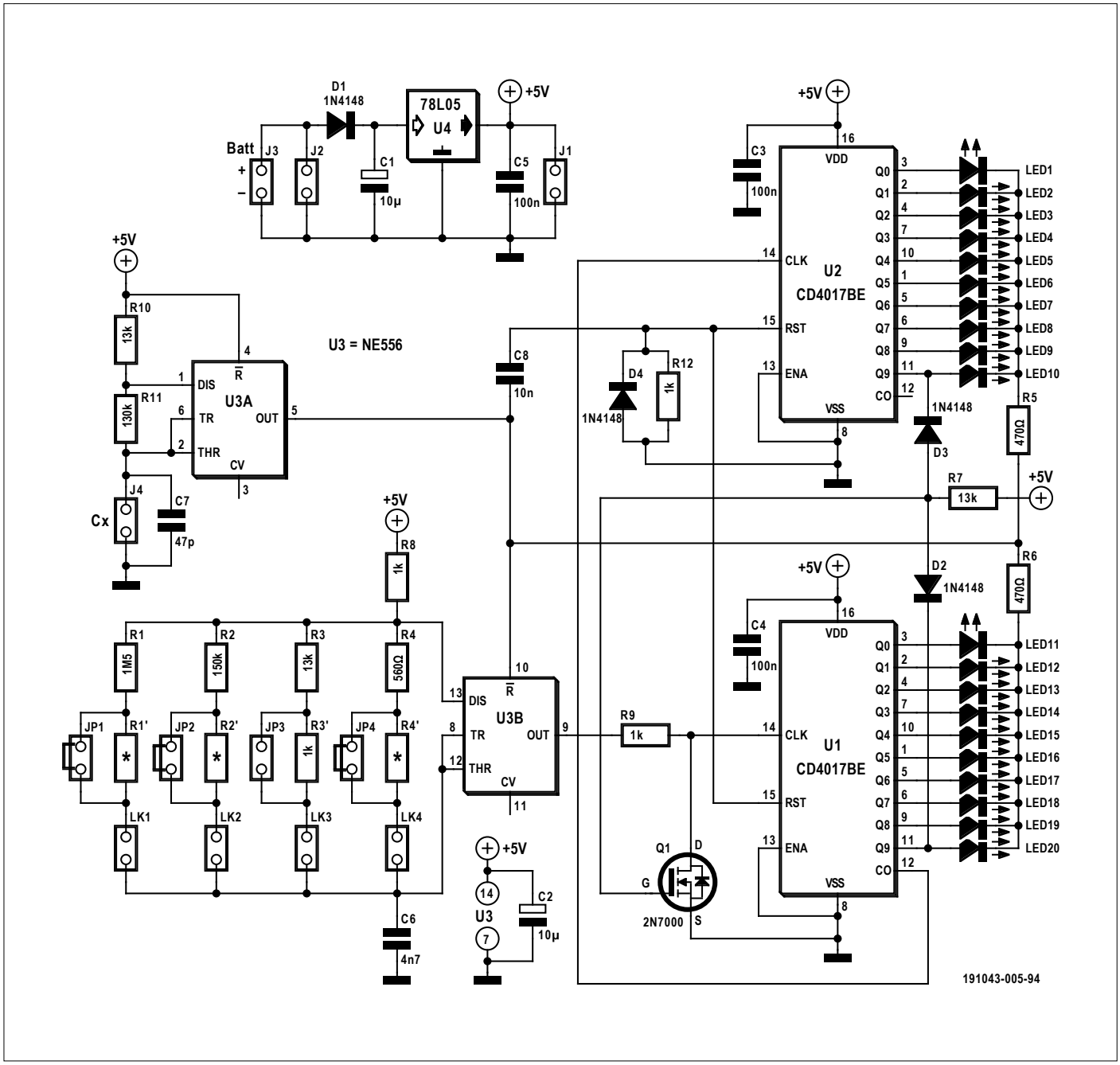


Figure 6. The complete schematic of the Capaci-Meter.

Starting the count at zero

To make sure that the counter starts from zero every time the counting starts, we need to reset the counter on the rising edge of the Cx clock signal. We do this using a simple differentiator (R12/C8) which gives very narrow pulses from each rising and falling edge of the rectangular signal (**Figure 5**).

To prevent the counters from experiencing signals that they are not designed to handle, diode D4 is added to remove the negative pulses.

The full schematic of the Capaci-Meter is depicted in **Figure 6**.

Preventing rollovers

Although not strictly necessary, the count is limited to 99 to avoid ambiguous values if the count exceeds 99 and rolls round. From a logical point of view, we stop the count when both the units AND the tens are 9. An AND gate is what we need. As only a single gate is required, we made one with a pair of diodes (D2 & D3) and MOSFET Q1. Now, when both the counters are outputting a high level on the '9' outputs, the master clock is pulled low by Q1 to prevent further counts occurring.

Master clock frequencies

We want the counter to count to 99 for a capacitance that is full-scale for the range that we're currently on. As an example, testing a capacitor of 9.9 µF on the top range, we want the count to reach exactly 99. In our circuit, the duration of the high pulse coming out of the Cx clock is determined by the values of R10, R11 and Cx, using the following formula:

T_{HIGH} =
0.693 × (R10 + R11) × (Cx + C7) [s]

With the given values for R10 and R11 and 9.9 µF for Cx, this gives us a high pulse duration of 0.981 seconds. During that time, we want to count 99 pulses from the master clock. So, the frequency of the master clock should be:

f_{CLOCK} = 99 / 0.981 = 100.9 Hz

That's the frequency required for the 9.9 µF range. If we reduce the capacitance range by a factor of 10 (to a full scale of 0.99 µF) and we still want to count to 99, then the master clock frequency needs to go up by a factor of 10, and so on.

These frequencies are the theoretical target values for each range, the actual frequencies may differ slightly due to component tolerances in the master clock. If desired, they can be fine-tuned by adjusting the values of the relevant pairs of resistors R1 & R1', R2 & R2', R3 & R3' and R4 & R4'. Without making any adjustments, the accuracy is likely to be within ±5% of full scale, and possibly better.

Measure the master clock frequency by monitoring pin 9 of U3 while the capacitor test leads are shorted. Note that fine-tune resistors R1', R2' and R4' (not R3') are shorted on the PCB, meaning that you must cut the short before you fit one of them.

Deviation from theoretical values

To cover all four measurement ranges, the master clock must generate frequencies from approximately 100 Hz to 100 kHz. According to the datasheet of the 555, the frequency is calculated as:

f = 1.44 / (C6 × (R8 + 2 × Rx)) [Hz]

Here Rx is either R1 + R1', R2 + R2', R3 + R3' or R4 + R4'. Unfortunately, as we approach higher frequencies, the actual output frequency deviates from the theoretical value. This deviation has been taken into account when selecting the range-setting resistors. However, there may be some benefit of further fine-tuning as detailed earlier.

Assembling the Capaci-Meter

As usual, start by mounting the small components like diodes and resistors and then work your way up to the taller parts. Note that R1', R2' and R4' should not be mounted, R3' on the other hand must be mounted. Make sure all polarized components (in this project everything except the resistors, the ceramic capacitors and the headers) are correctly oriented. Using sockets for U1, U2 and U3 is highly recommended. Thread the battery wires through two of the PCB holes before soldering them into the pads.

Testing it

Start by double-checking that the whole board looks OK and that there are no shorted joints. Check also that the ICs are inserted correctly.

Place a jumper across one of the range selection headers. Attach a 9-volt battery to the battery clip. You should see the green LEDs light, possibly in addition to a red LED on the right-hand circle of LEDs. Find a capacitor of a known

Advertisement

HAMMOND MANUFACTURING®

Ventilated enclosures

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

01256 812812
sales@hammond-electronics.co.uk



COMPONENT LIST

Resistors (all 1%, 0.25W)

R1=1.5MΩ
R2=150kΩ
R3,R7,R10=13kΩ
R3',R8,R9,R12=1kΩ
R4=560Ω
R5,R6=470Ω
R11=130kΩ
R1',R2',R4'= Not fitted (optional to adjust accuracy)

Capacitors

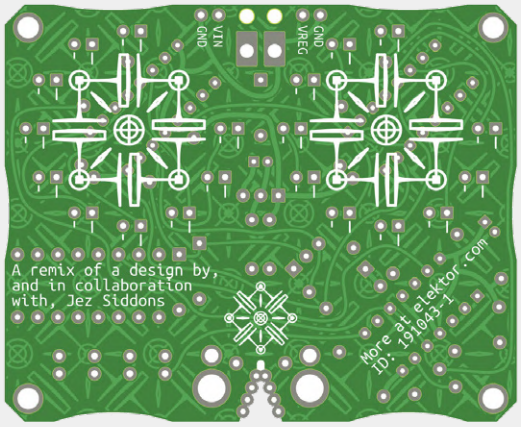
C1,C2 = 10µF, 10V
C3,C4,C5 = 100nF
C6 = 4.7nF, 1% or 5%
C7 = 47pF
C8 = 10nF

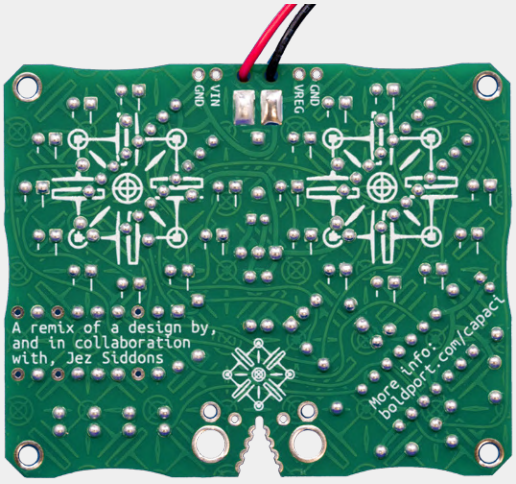
Semiconductors

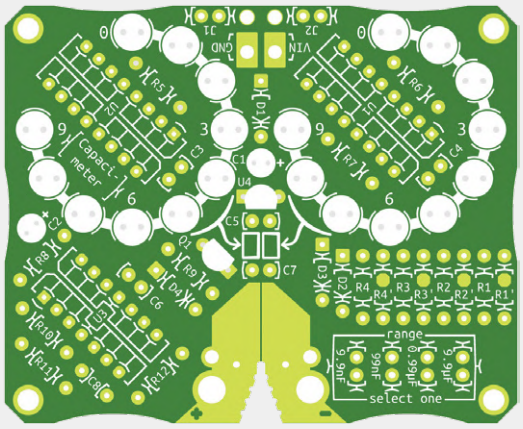
D1,D2,D3,D4 = 1N4148
Q1 = 2N7000
U1,U2 = CD4017BE
U3 = NE556
U4 = 78L05
LED1, LED11 = 2x green LED, 5mm, low current or high brightness
LED2 - LED10, LED12 - LED20 = 18× red LED, 5mm, low current or high brightness

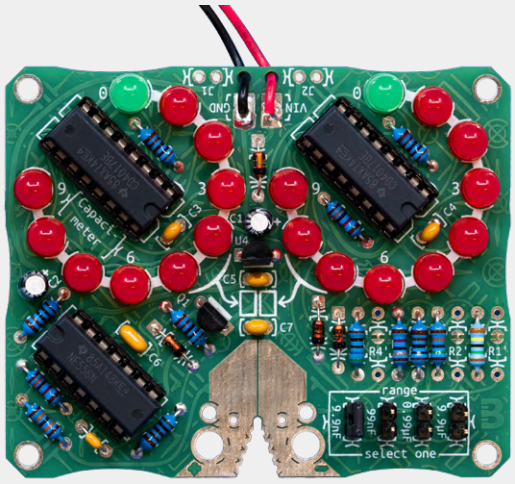
Miscellaneous

J1,J2,LK1,LK2,LK3,LK4 = 2-pin pinheader, 2.54mm pitch
J3 = PP3 9V-battery clip
J4 = Connections to test capacitor









JP1 = Jumper for range selection
2× 16-pin DIL socket, 7.62mm width for U1 & U2
1× 14-pin DIL socket, 7.62mm width for U3

Printed circuit board (PCB) 191043-1

value that sits neatly within one of the four measurement ranges, e.g. 47 nF for the 99 nF range. Apply the test capacitor across the two test pads or use test leads if you wish (the holes in the test pads are suitable for 2 mm and 4 mm banana plugs). The capacitor's value should appear on the display. If it doesn't, check the selected range and double-check your soldering.

A last note for completeness sake. The Capaci-Meter can be powered from either a 9-volt battery (J3) or an external 7.5 to 15 V_{DC} power source connected to J2. Do not connect them at the same time. J1 is for testing purposes only; do not use it as a power input.

Finally

I would like to thank Stephen Bernhoeft for his critical eye, advice and clarity with refining and testing this circuit. Thanks also to Saar Drimer from Boldport for designing the PCB for it.

I hope you enjoy building and using this project as much as I have. ⏮

191043-01

@ WWW.ELEKTOR.COM

→ **SKU19118 Capaci-Meter - Bare PCB (191043-1)**
www.elektor.com/191043-1

→ **SKU19119 Capaci-Meter - Kit of Parts (191043-71)**
www.elektor.com/191043-71

HOW TO: Debounce a Mechanical Contact or Switch

A switch is either open or closed, isn't It?



By **Clemens Valens**

Many engineers treat mechanical switches or contacts as simple, 2-state devices that are either open or closed. Although this view suffices most of the time, there are situations where it is too simplistic and causes trouble. Contact resistance is one source of problems, contact bounce another. In this installment of the new "HOW TO:" series we look at the second issue.

It's a matter of time

As everything in our universe, state changes take time to complete and switches, mechanical or not, are no exception. Most mechanical switches are rather slow devices and a change of state may take several tens of milliseconds to stabilise. During the state change the contact bounces a few times, like a ball hitting the ground. Unlike a ball, bouncing also appears when a contact is opened. Systems much faster than the switch may notice contact bouncing and be disturbed by it. As a result, instead of a single keypress the system may detect multiple presses and race through menus, leaving no chance to the user to select the desired option.

Thou shalt debounce

Debouncing is the technique of proofing a system against contact bounce. Several methods exist, from analogue RC filters through specialised ICs to software algorithms. They all have in common that they try to deliver a well-defined, glitch-free contact state to the system. Please keep in mind that a contact can be a human-operated switch or pushbutton or a machine-controlled contact like a microswitch, rotary encoder or relay.

RC network for debouncing

The idea here is to connect the switch to a circuit that's even slower than the switch itself. When this circuit is slow enough, it will simply not notice the bouncing at its input. The output of the circuit is a steady state, low or high, with slow transitions in between different states (**Figure 1**). This sounds like a nice & easy solution, but there are issues:

- When the input does not have enough hysteresis and noise is present, a slowly changing signal driving a fast input can produce undesirable effects somewhat similar to contact bounce. They can also result in excessive power consumption. Schmitt-trigger inputs at the receiving end are therefore highly recommended. If they are not available, add them to your filter;
- if made too slow, short but valid state changes may be missed;
- it requires extra components that eat up board space and which cost money;
- in the common switch-to-ground-with-pull-up-resistor-and-RC-network, pressing and releasing the button do not produce identical results. Adding a diode may overcome this problem (Figure 1) but adds yet another line to the bill of materials.

More Involved Debouncing Circuits

A set-reset (SR) latch or a D-type flip-flop can be used to eliminate contact bounce. However, to do so requires an SPDT or changeover switch which is more expensive than a basic SPST pushbutton. Note that this method can be implemented easily in software if you are willing to sacrifice two pins per SPDT switch. A one-shot multivibrator or timer can be a solution but will produce a pulse instead of a steady level.
Special debouncing ICs exist. A classic is the MC14490 (**Fig-**

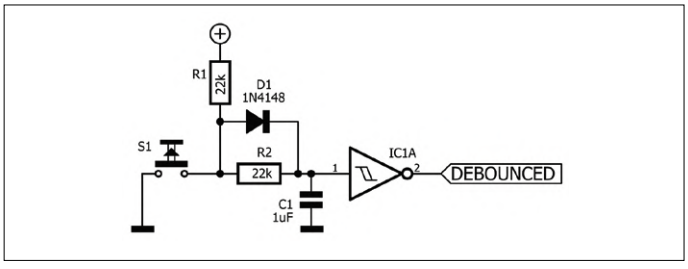


Figure 1: Debouncing a mechanical contact with an RC network. The values of R1, R2 and C1 depend on your application even though the ones given will probably work fine in many cases. D1 ensures that opening and closing the contact produce similar delays.

ure 2), but there are others like the MAX6816 (multiple channels MAX6817 & MAX6818) or the LTC6994. The company LogiSwitch seems to try to make a living out of debouncing.

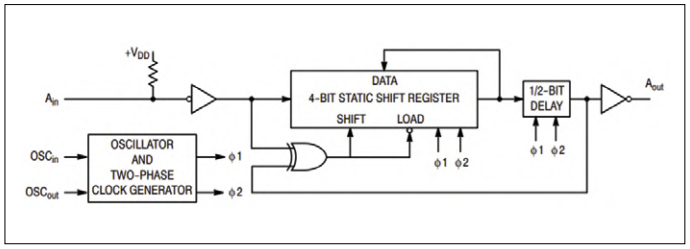


Figure 2: Inside the MC14490 six-channel contact bounce eliminator.

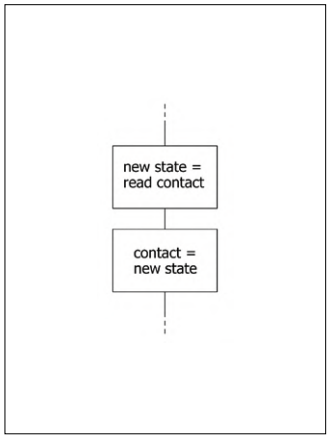


Figure 3: When a contact is read only every once in a while, it is possible to get away without more sophisticated debouncing techniques.

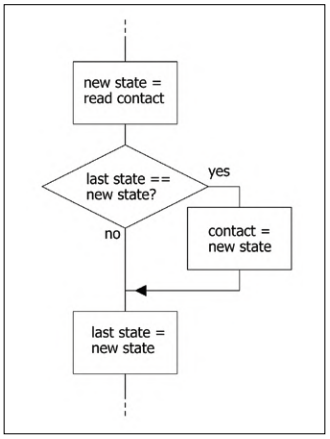


Figure 4: When polling is not too fast (<0.1 Hz or so) two consecutive identical contact state values can be enough to accept a new state.

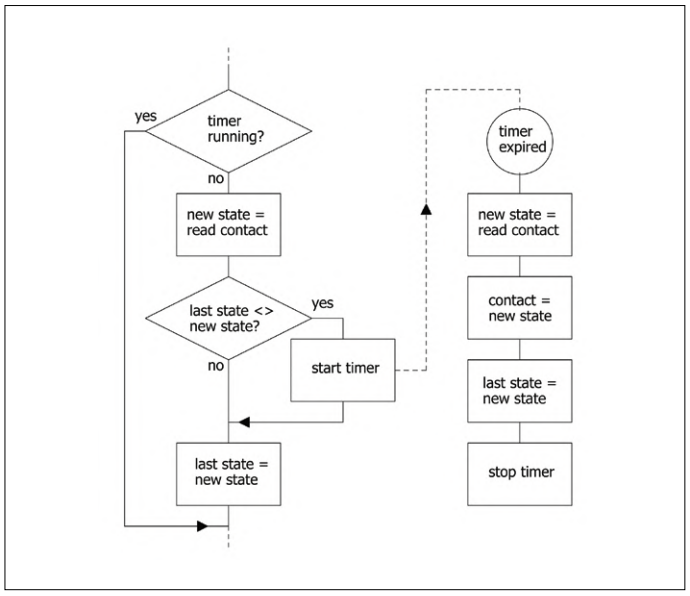


Figure 5. A contact transition starts a timer. The main program loop ignores the contact while the timer is running. The state of the contact is read again when the timer expires, i.e. when contact bouncing is expected to be over.

Furthermore, Maxim offers so-called 'Contact Monitors' that do debouncing.

The advantage of debounce ICs is that they can pack more than one channel in a single device and that they may add features like overvoltage and ESD protection. Their main inconveniences are, of course, extra cost and board space.

Debounce in software

The reason to debounce in hardware usually is because for some reason it cannot be done in software. Yet in many microcontroller-based applications the MCU has ample resources to take care of debouncing. But how? Use polling or interrupts?

When polling the MCU checks the contact input on a regular basis. This can be done under timer control or whenever the MCU has time for it. Polling may miss contact state transitions when they are too short or detect them late (high latency), but it does not really interfere with time-critical processes.

Interrupts, on the other hand, can detect contact state transitions almost as soon as they occur (low latency), resulting in a responsive system. However, when the MCU is fast enough, it may trigger several times due to contact bounce and hinder time-critical processes, produce false positives or both. Disabling pin interrupts during the bounce phase may help here.

Never say never

Some people say that you should never connect a switch to an interrupt pin because the bouncing may disturb the interrupt input's internal logic (or some other reason). This is, of course, nonsense as it all depends on the application. If contact bounce can lock up a system, then EMI and other noise can do so too. Noise filtering must be added to such systems anyway. Most, if not all modern microcontrollers have glitch filters on their inputs. If this is not enough, insert a noise filter between the contact and the input pin.

A few debouncing algorithms

Lazy polling

As soon as a transition is detected, accept the new state. This is shown in **Figure 3**. This is an easy method for low-speed (25 Hz or less) polling systems, but it may mistake a glitch for a transition if the glitch happens exactly at the sample moment. **Figure 4** shows a simple way of improving matters by requiring two consecutive identical samples before accepting the new state.

Use a timer

After the detection of a transition, start a timer to check the input again 20 milliseconds or so later. Use as contact state the value at the input when the timer expires (**Figure 5**). This works with both polling and interrupts but requires a timer. Keeping the pin interrupt disabled while the timer is running will avoid unnecessary stress due to bouncing.

X out of Y

A step further is to require two or more identical samples before accepting a state change, for instance three out of four or seven out of ten. This is the digital equivalent of the integrating RC debounce network, see **Figure 6**. Instead of requiring *x* out of *y* valid reads, you can also require *z* consecutive valid samples. Its responsiveness depends on the number of samples and the sampling rate.

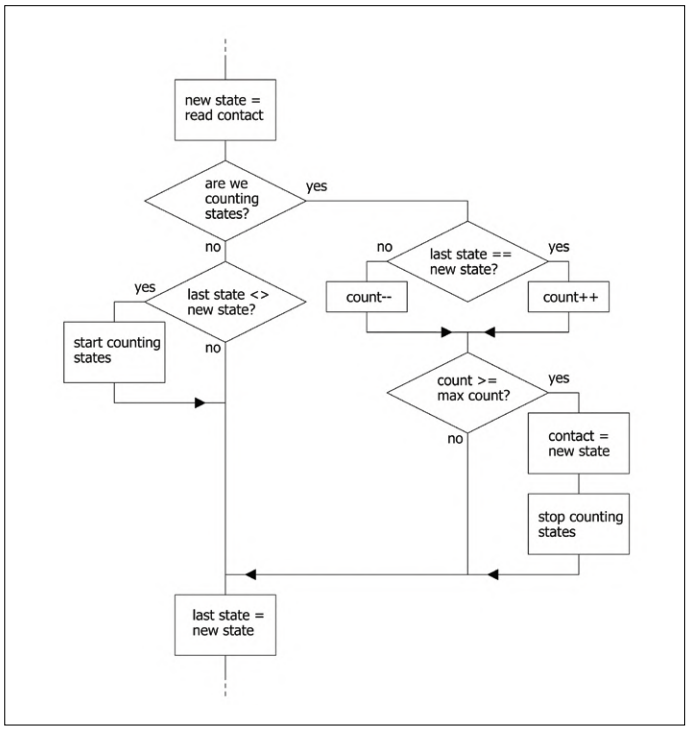


Figure 6. A flowchart-equivalent of an RC debounce network. This one will trigger when at least 'max count' valid samples have been read.

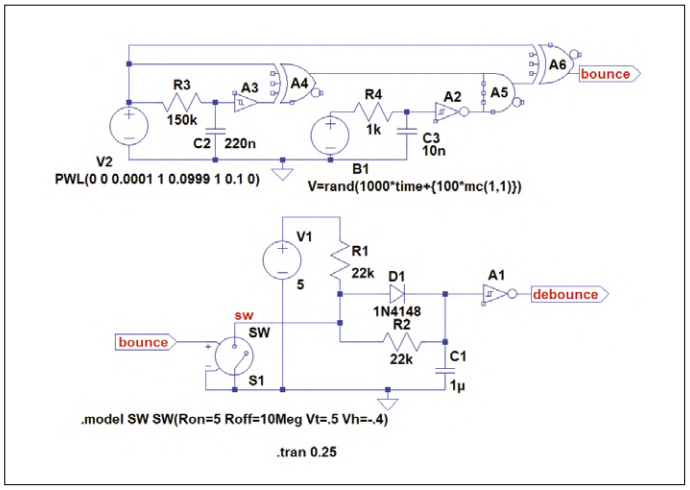


Figure 7. Contact bounce and debounce simulated in LTspice. Voltage source V2 produces the contact closure pulse, voltage source B1 adds bounce to it. The value of B1 determines the minimum bounce glitch length (here it is 1 ms). Change the values of R3/C2 and/or R4/C3 to modify bounce characteristics.

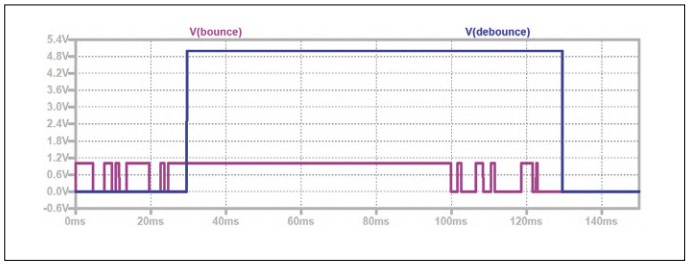


Figure 8. Simulation results show contact bounce in pink and the debounced result in blue.

Example code?

You now probably expect example code that you can cut & paste into your program. Well, there isn't. How to best implement debouncing algorithms highly depends on the cleverness of the programmer and the resources available. For inspiration refer to the flowcharts that illustrate this article.

Simulate it

Figures 7 and 8 show an LTspice simulation of the RC debounce circuit that we started this article with. To make the simulator produce different results for every run, you should check the option 'Use the clock to reseed the MC generator[*]' on the 'Hacks!' tab of the 'Control Panel' (the button with the little hammer on it) of LTspice (**Figure 9**). The '{100*mc(1,1)}' part in the value for B1 makes this possible. Play with the different R/C values to change the debounce characteristics.

So, what do I use?

As with every subject in electronics, in engineering and in general, there is much more to be said about it. Every switch bounces in its own way, and its bounce characteristics may change over time. Bounce behaviour may be different on opening or closing. Contacts wear and the surface properties change; contact surface has an influence on conductivity. Some systems require fast response times, others don't care and so on and so further. As a result, it is impossible to say which debounce method is best for you. Learn and understand your system first before deciding on which technique to implement. ⏮

191015-01

Web Link

[1] LTspice simulation files: www.elektormagazine.com/191015-01

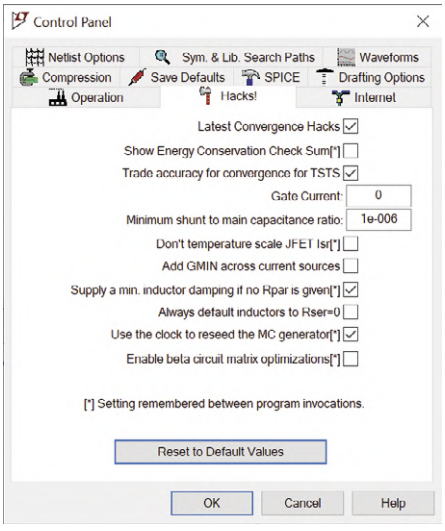


Figure 9. Checking 'Use the clock to reseed the MC generator[*]' will make LTspice produce different results for every run.

Developer's Zone

Tips & Tricks, Best Practices and Other Useful Information

By **Clemens Valens** (Elektor Labs)

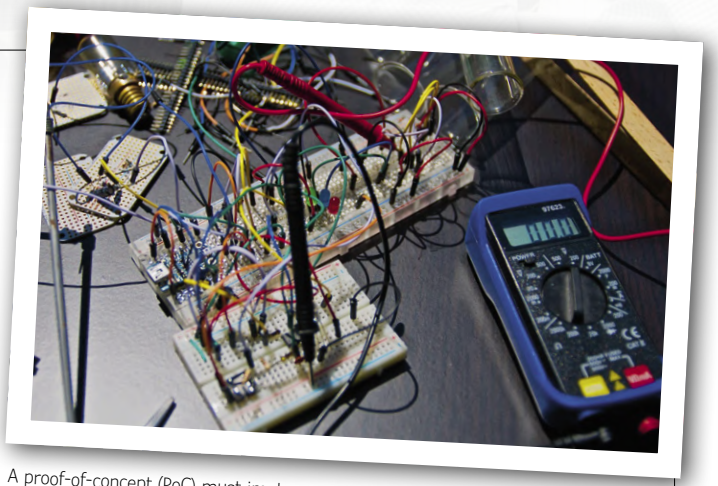
FROM IDEA TO PRODUCT

PART 3

In the previous instalments of this column we discussed a few things to investigate before you even start developing a product. From now on we will suppose that you addressed these issues and found suitable solutions. Let's start designing!

BUILD A PROOF OF CONCEPT

With the list of specifications in hand the first thing to do is come up with a proof-of-concept (PoC) of your product. This is a sort of prototype showing that what you have in mind is possible. It does not have to implement every function or option from the specs list, as long as the main functions work as intended. The goal here is to uncover any technical or physical impossibilities you may have overlooked before. If you can't get your PoC to work, you either give up now or revise your project from the start. Simply dropping the specs you can't meet without redoing market research doesn't make much sense as the final product no longer conforms to the potential user's desires.



A proof-of-concept (PoC) must implement at least the main functionality of your product.



Simulate the hell out of it!

SIMULATE THE HELL OUT OF IT

Many designers tend to stay away from circuit simulators. There are many reasons for this, but maybe these were valid twenty years ago? A good simulator can save you time and money; some even come for free. True, you should know how to operate it and how to interpret its results, but a simulator sure comes loaded with measurement tools that you just can't afford. It has power supplies with unlimited power. Visualising any waveform, voltage or current is just a mouse click away; probes never slip off pads and cause short circuits. Thousands of amperes through a transistor do not toast the device. And even if it did, you replace it without heating up a soldering iron.

SIMULATIONS NEVER WORK FOR REAL

If 'a' reality does not agree with its simulation, maybe that reality is not good enough? A breadboard with bad contacts and long wires is not a clean, noise-free environment inside a computer. By the same token, a simulation can be enhanced. Use SPICE models provided by parts manufacturers instead of generic models. Do not forget to test the models you use. Add tolerances, noise sources, and things like internal impedances.

STAY FOCUSED

Besides requiring good knowledge of how to work with a simulator, it's sometimes difficult to prevent getting away. "What if?" questions may result in hours spent investigating obscure circuit options and all sorts of components added. Stay focused and occasionally step back. Make sure your simulation remains valid.

STICK TO COTS

The PoC may ignore the specified cost limitations and can be constructed from common off-the-shelf (COTS) modules, parts and boards. Avoid using too much stuff from dismantled equipment that you happen to have lying around, as such parts may get you into procurement or obsolescence troubles later. This also holds for those "brand-new" boards, modules and parts that have been gathering dust on your desk for many years.



Using common of-the-shelf (COTS) products whenever possible makes a developer's life much simpler.

BEWARE OF 'SOMETHING SIMILAR'

If the final product needs a microcontroller in a microscopic package that you don't know how to handle, it's all right to replace it in the PoC by a large development board the manufacturer designed for it. However, be very careful though when such a board does not exist, and you replace it with 'something (highly) similar'. It's easy for a design to come to rely on the specific function of a part, but if that part turns out to be unsuitable for the final design you may be in trouble. The same is true for the dev board; avoid making it the centrepiece of your PoC.

SUBTLE ISSUES ARE THE TOUGHEST

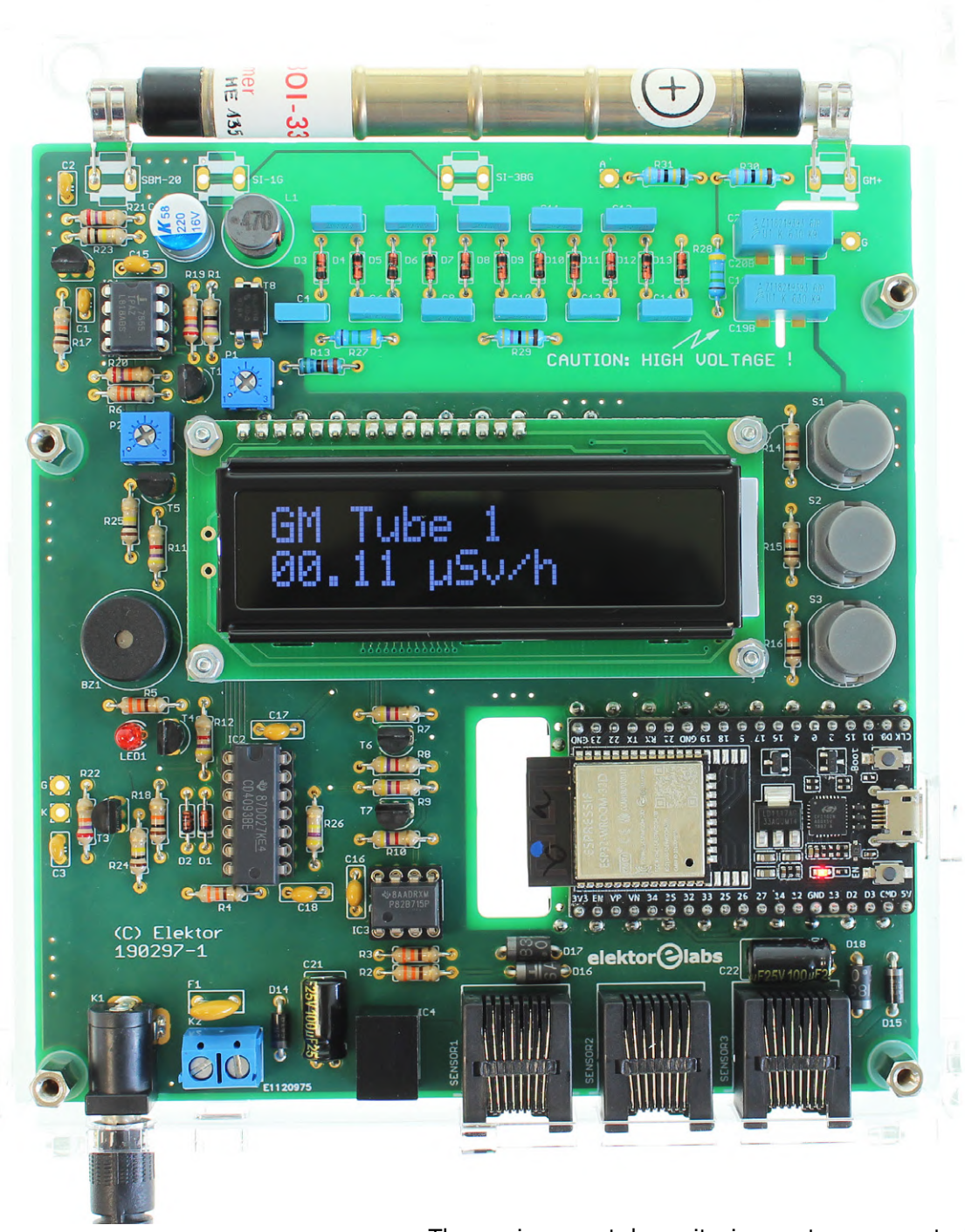
During PoC development many subtle issues may pop up like incompatible signal levels, synchronisation and timing differences, incomprehensible descriptions in datasheets, and interference. Sometimes they go away simply by replacing a part. A triple-A-grade, mil-spec'ed, space-rated opamp with terahertz bandwidth may solve a design problem, but is it a viable solution to reach your goal? On the other hand, if you discover you're needing a lot of parts to work around an issue, you may be on a dead end too. In both cases it may be worth revising some principles of operation first before continuing. Don't hesitate to go back to the simulator to figure things out.

WHAT'S NEXT?

Supposing that your PoC performs as intended, we will turn it into a schematic. Sounds easy, right? Well, it isn't.

Extendable Environmental Monitoring System

publishes data on IoT platforms



The environmental monitoring system presented in this article measures the background ionizing radiation level and publishes data on IoT platforms like openSenseMap and ThingSpeak.

PROJECT DECODER

Radiation | high-voltage | Geiger-Müller tube | Arduino | ESP32 | IoT

entry level
➡ intermediate level
expert level

4 hours approx.

Soldering iron, Arduino IDE, lab power supply

£65 / $80 / €70 approx.

Quick Specifications

- 9 – 20 V_{DC} supply voltage
- ESP32-based with Wi-Fi connectivity
- Up to two Geiger-Müller tubes
- Adjustable tube voltage (up to 1.1 kV)
- Measures gamma (and beta) radiation
- Display data on local LCD and remote IoT platform
- Highly configurable
- High-radiation-level alarm with adjustable threshold
- Three buffered I²C expansion connectors

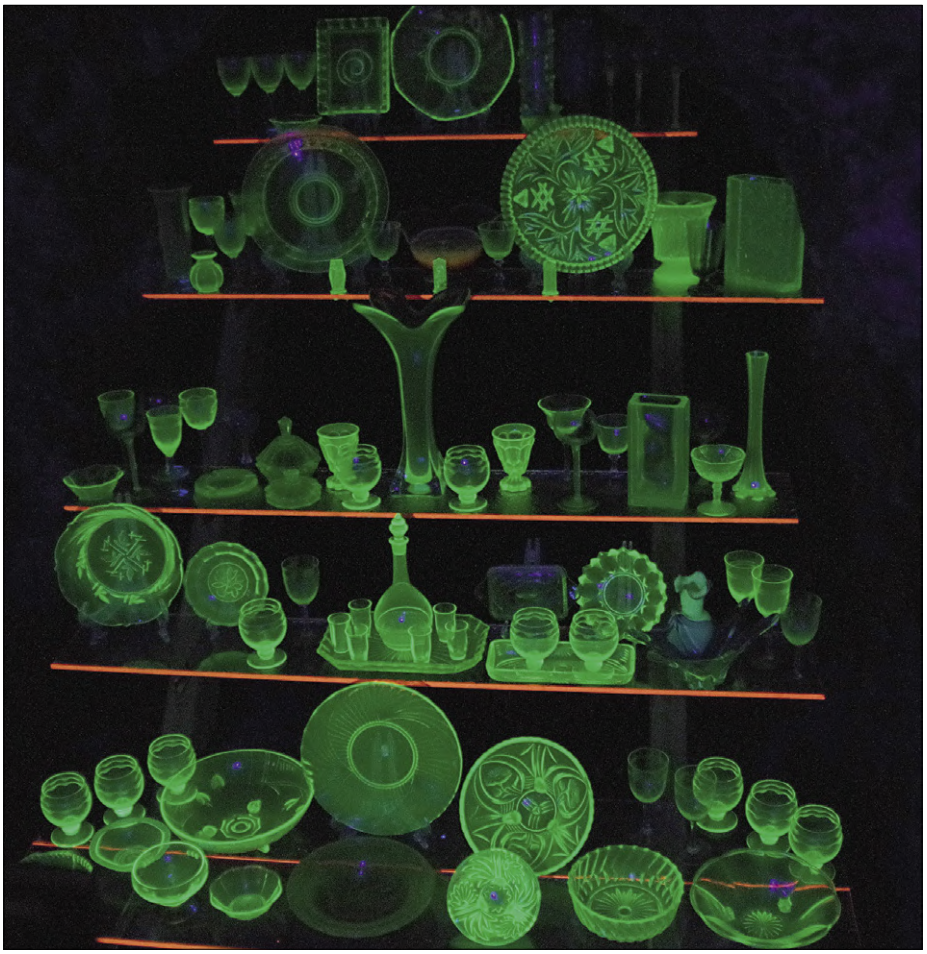


Figure 1: An interesting radioactive source to check the Environmental Monitoring System with is uranium glass. Beads of this glass, still produced in the Czech Republic, are cheap and safe to use as the uranium salts are contained by the glass.



Figure 2: Two examples of common metal-wall Geiger-Müller tubes.

Background radiation

Produced by natural as well as man-made sources, background radiation can be considered the level of ionizing radiation always present at a certain location. Ionizing radiation is any form of radiation capable of detaching electrons from atoms or molecules. It is not the same as radioactivity, but radioactive decay commonly results in ionizing radiation [1]. Virtually all materials found in nature produce such radiation due to the presence of naturally occurring radionuclides. Those that concern us are mainly uranium-238/-235, thorium-232, potassium-40 and rubidium-87. In the air surrounding us, we find mainly radon-222, its daughter nuclides, and cosmogenic radionuclides [2]. Finally, radioactive man-made items like building materials, consumer goods and household items (**Figure 1**) also contribute to (in-house) background radiation [3].

Measuring ionizing radiation

When monitoring background radiation, alpha and beta radiation are usually less of a concern as their range in air is very limited. For best results, a G-M tube with a high gamma sensitivity should be used. For an entry-level system, the commonly found Russian SBM-20 tube is a good choice. Be careful when ordering such a tube as many compatible tubes are sold

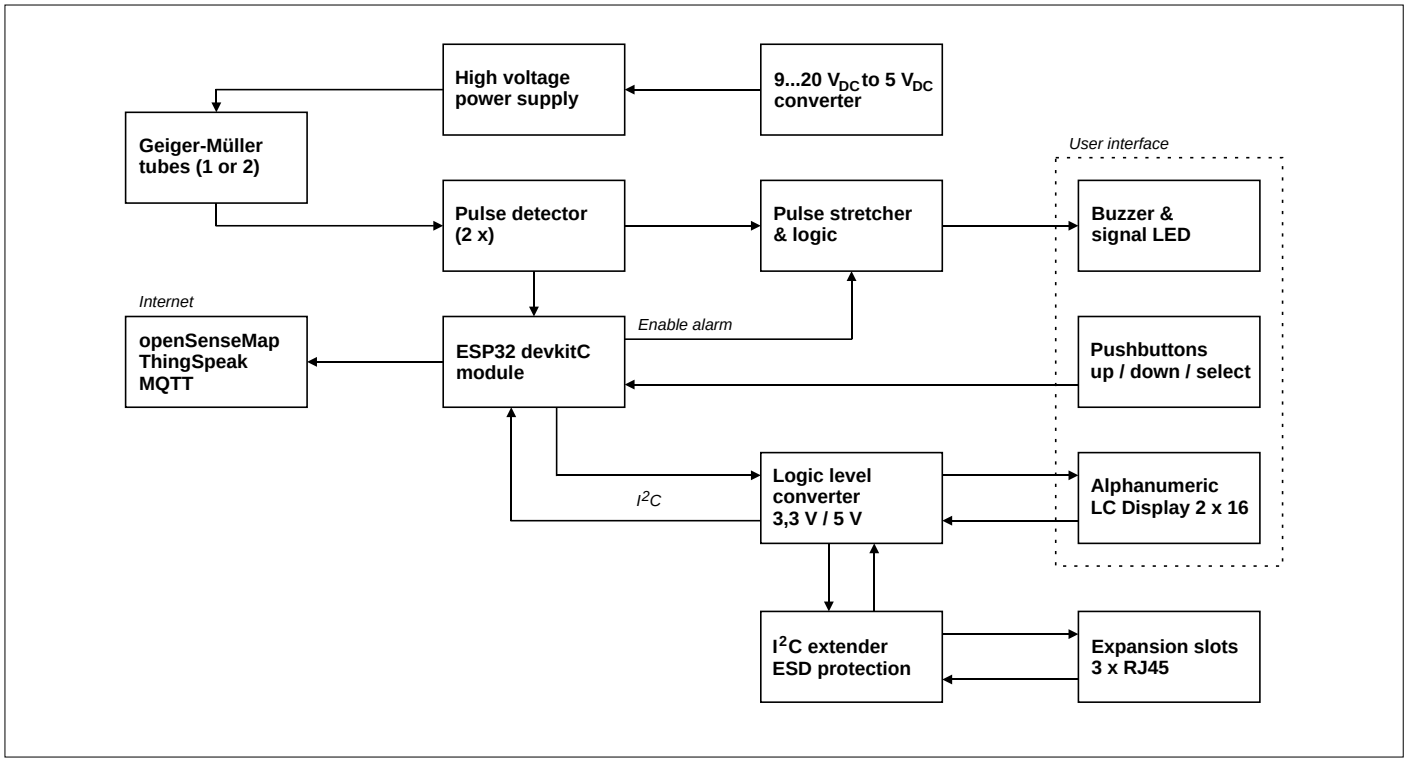


Figure 3: The block diagram of the Environmental Monitoring System shows that it is more than "yet another" Geiger-Müller counter.

as SBM-20 tubes. Another option is the affordable LND712 but as it is not very sensitive to gamma radiation produced by cesium-137, it is not the best choice to detect radiation produced by nuclear disasters like Chernobyl. We have also tried LND71217 and LND78017 tubes. The first has a sensitivity like the SBM-20, the latter is much more sensitive.

The Geiger-Müller tube

German physicist Johannes Wilhelm "Hans" Geiger invented its operating principle in 1908 while German physicist Walther Müller developed the idea further into a practical tube in 1928.

G-M tubes usually have the form of a metal tube acting as the cathode and a central anode wire (**Figure 2**). Some tubes have a mica window for the detection of alpha particles and low-energy beta radiation that cannot penetrate the metal wall of the tube. Tubes made of glass instead of metal and so-called pancake tubes are also quite common [4].

The tube is filled with a 'Penning' mixture consisting of an inert filler gas such as neon or argon and a quenching gas, usually chlorine or bromine. The gas mixture has a pressure of a few tenths of the atmospheric pressure. A high voltage — typically between 300 V and 1200 V — is applied between the anode and cathode. An alpha or beta particle entering the tube will ionize the gas. The same happens indirectly by the photoelectric effect when a gamma photon hits the tube and knocks an electron from the inner wall of the tube into the filler gas.

The Townsend avalanche

When the filler gas is ionized, pairs of positively charged ions and free electrons are created. The electric field inside the tube will accelerate the positive ions towards the cathode and the negative electrons towards the anode. If the free electrons gain enough momentum, they will ionize other gas molecules on their way, creating additional free electrons which in turn will ionize even more gas molecules, and so on. The result of this so-called Townsend avalanche is an easily detectable electrical pulse.

Eventually the avalanches will stop when the strength of the electric field drops due to the accumulation of positive ions around the anode.

The quenching gas counteracts prolonged avalanches and avoids spurious secondary discharges by preventing UV photons from being emitted when the positive ions reach the cathode. When the avalanche discharge stops, the tube will be temporarily insensitive to a new ionizing event, the so-called dead time.

Advantages and inconveniences of G-M tubes

G-M tubes are relatively inexpensive; they can detect all types of radiation, and are durable and portable. Compared to scintillation and semiconductor detectors, the output pulse is always the same regardless of the energy level or the type of the detected radiation. G-M tubes have a very low efficiency (ε) and only detect a small fraction of the ionizing radiation that hits them. They are generally not sensitive enough to reliably detect e.g. radioactive contamination of food products.

How to use a G-M tube in a circuit

Now that we understand how G-M tubes work, we can design a circuit for using them. Basically, all we need is a high-voltage (HV) supply to bias the G-M tube and an amplifier to make the output pulses audible and visible.

The block diagram of our design is depicted in **Figure 3**. It is a bit more complicated than just described due to the bells and whistles we added. The basic circuit is formed by the HV supply, the G-M tube, a pulse detector and

stretcher, and a buzzer and a LED. This design, however, can listen to two G-M tubes instead of just one (useful for doing things like comparisons), which is why there are two pulse detectors. Furthermore, we added a microcontroller unit (MCU) to it for further processing of the data. The MCU displays measured values on an LCD and it transmits them over a wireless connection to the Internet. Last but not least, we added three buffered I²C expansion slots for the connection of sensors or other devices.

Translating the block diagram into the schematic of **Figure** 4 is relatively easy.

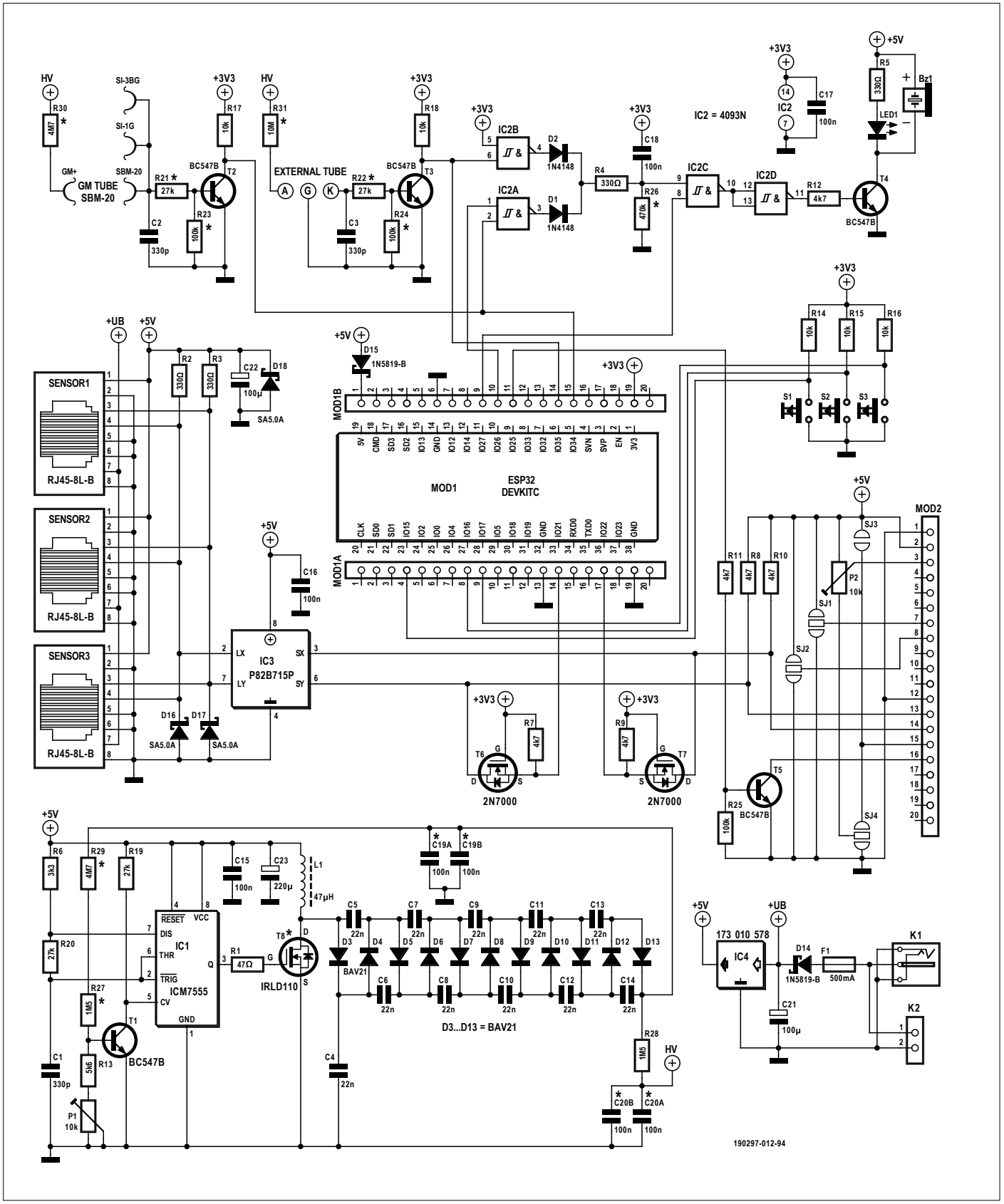


Figure 4: The voltage multiplier constructed with its 11-stage diode ladder is the main attention of this circuit. The ESP32 module is a worthy second.



Figure 5: The high-voltage supply powers the Geiger-Müller tube.

First, a high-voltage supply

The high-voltage (HV) supply is built around a 7555 timer IC. It is a step-up converter followed by an HV cascade that multiplies the output voltage by a factor of six (**Figure 5**). To get a voltage that's as clean as possible, the output of the multiplier is filtered by RC-filter R28/C20 (A or B). As T8 (a type IRLD110 MOSFET) can switch voltages up to 100 V, the theoretical maximum output voltage is 600 V. In real life, the output voltage can be adjusted from 280 V to 550 V, enabling the use of many types of G-M tubes. If a type IRFD220 is used for T8 and C19B, and C20B are mounted instead of C19A and C20A, an even higher tube voltage is possible. In that case, the value of the feedback resistors R27 and R29 must be increased to 5.6 MΩ or so each.

The HV supply is powered from the stabilized 5-volt rail and is therefore independent of the circuit's 9 to 20 VDC input supply voltage. At an output voltage of 550 V resistors R27 and R29 dissipate around 50 mW and get a little warm. Replacing them by low-temperature-coefficient types improves the HV supply's stability even more.

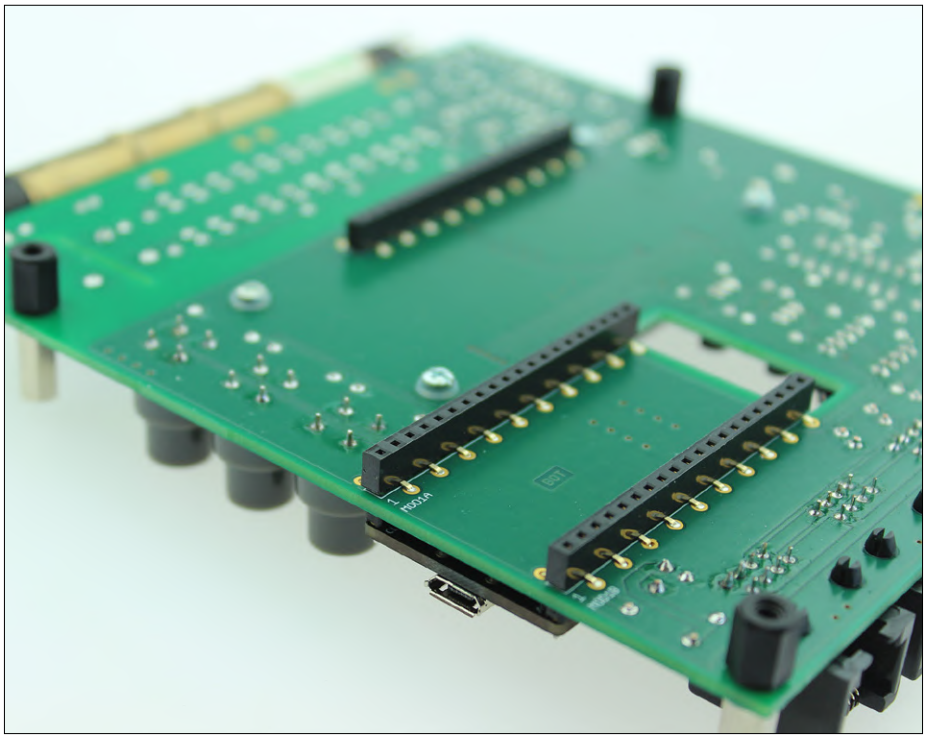


Figure 6: Low-profile sockets avoid a too tall board stack.

Second, we need a pulse detector

The pulse detector circuits operate at the "low" side of the G-M tube. They consist of an NPN transistor, a capacitor and a few resistors. Anode resistors R30 and R31 and the components in the detector circuits should be chosen according to the G-M tube to be used. As a rule of thumb, the sum of the values of the resistors connected to the base of the transistor should be approximately 1/45th (0.022) of the value of the anode resistor.

The collectors of the transistors are connected to the MCU (and to the 3.3-volt rail via a 10 kΩ pull-up resistor).

Add a pulse stretcher

The pulses at the collectors are too short (a few microseconds) to be noticeable when driving LED1 and buzzer BZ1 directly. They are therefore 'stretched' by the circuit R4, R26, C18 and IC2C. At high pulse rates, the stretcher may be continuously re-triggered before the end of the current pulse. A sustained beep is the result. Play with the component values of the pulse stretcher to mitigate this effect if desired.

Pulse stretching could have been implemented in software, but at high pulse rates (several hundred pulses per second or higher) this would consume precious processing power that we prefered to reserve for more important tasks.

MCU plus LCD with I²C interface

The ESP32 is a popular MCU with wireless networking (Wi-Fi) capabilities. Mounted on a module it can be bought in many shapes. We chose the ESP32 devkitC module. We added to it a user interface consisting of three pushbuttons ('Up', 'Down' and 'Select') and a 2 × 16-character alphanumeric LCD. This display has

an I²C interface, which is quite rare. As it operates from 5 volts, level converters T6 & T7 are needed. Standard, parallel LCD modules with a small I²C-to-parallel convertor PCB mounted at the bottom side are more common but would protrude too far above the PCB.
Even though the ESP32 module has two 19-way pinheaders and the LCD only has 16 pins, we opted for 20-way sockets as these are readily available. The extra pins are simply left unconnected.

Extension connectors

Three RJ45 jacks are available for connecting external sensors to the system using standard Ethernet patch cables. The jacks expose the input supply voltage, the 5-volt rail and a buffered I²C bus (via IC3, a P82B715 bus extender). TVS diodes protect the 5-volt rail and the I²C signals against high-voltage surges.

Software

An MCU without software is like a car without a motor, and so we spent quite a few hours developing 's/w'. As a result, the firmware features many commands for controlling the system (see **inset**, documented in the file 'cmd_proc.ino'). Commands can be sent to the Environmental Monitor over Wi-Fi or through the serial port (via the USB connector of the ESP32 module). The serial port speed is set to 115,200 bits/s. When using the serial monitor of the Arduino IDE, deselect the option 'No line ending'. Send Wi-Fi commands to port 5010 at the Environmental Monitor's IP address. You can use PuTTY or a TCP terminal app on your smartphone for this purpose.
The software was written in the Arduino IDE 1.8.9 with the Arduino core for the ESP32 installed.

Pulse counting

The pulses from the G-M tubes are counted during an adjustable interval and then converted to an average counts-per-minute (CPM) value for that interval. A dose rate in microsieverts per hour (µSv/h) is calculated from this value and published on an IoT platform. The value on the LCD is updated every 15 seconds. As it is based on the number of pulses detected during the last minute, it is responsive enough for experimenting with a radioactive source. The values shown on the LCD may differ somewhat from the values sent to the Internet due to less filtering.

From pulse count to microsievert per hour

First, correct the count rate R (expressed in CPM) for the dead time of the tube:

$$R_{corrected} = R_{observed} / (1 - (R_{observed} \times t_{dead})) \text{ [count/min.]}$$

Note that the dead time of a tube is usually specified in microseconds (µs) and in the above formula, must be converted to minutes (m) when the count rate R is in counts per minute. For the SBM-20 that has a dead time of 190 µs its effect is insignificant for CPM values below 3,000 CPM (i.e. under 1% error).

Conversion factor

The second (and last) step of the calculation is easy when you know the 'conversion factor' of your G-M tube. Unfortunately, a radioactive source with a precisely known activity is required to determine this factor. Such sources are not only very expensive, they usually have to be licensed by a nuclear regulation authority... As an alternative, you can trawl the Internet for official radiation monitors in your vicinity and fine-tune the conversion factor of your system to get measurements in line with the official values. For the SBM-20 tube used in our prototype, we obtained good results with a conversion factor of 0.003931.

Configuring the Environmental Monitoring System

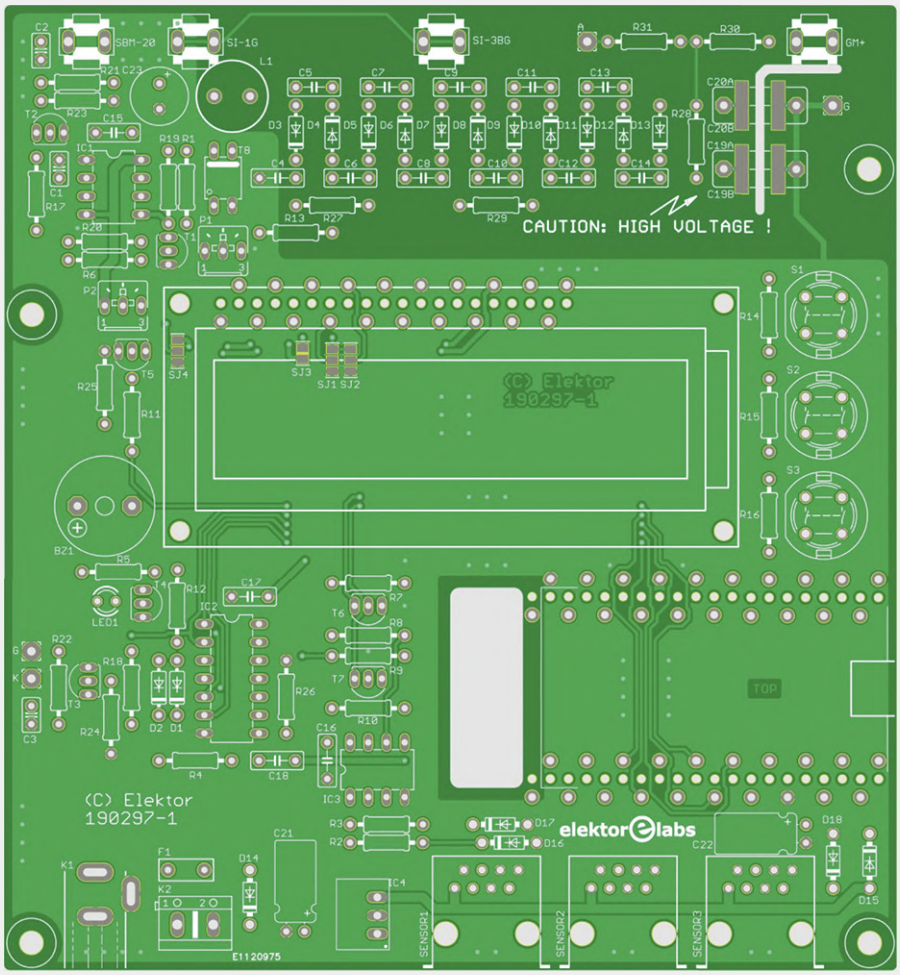
Commands for configuring the G-M tubes.			
gm N			**G-M tube, N=1...2**
	set		Settings
		attached|att YN	The tube is physically attached y/n
		accperiod|ap N	Period for reporting accumulated value (minutes)
		deadtime|dt N	Dead time (µs)
		cnvfactor|cf N	Conversion factor
		threshold|th N	Threshold value for alarm (nanoSv/h)

Commands for configuring the buzzer and LED.			
buzzer			**Buzzer & LED**
	alarm|al YN		Turn on or off the alarm sound
	set		Settings
		enabled|ena YN	Enable or disable the buzzer

If you want to publish the measurements on an IoT platform you can choose openSenseMap and/or ThingSpeak (accounts are required). For ThingSpeak you'll need a write API key and a Field name for each G-M tube. For openSenseMap you'll need a SenseBox ID and a Sensor ID.

thingspeak|thsp				ThingSpeak client
	set			**Settings**
		gm N		G-M tube, N=1..2
			writekey|wk "..."	Write API key
			fieldname|fn "..."	Field name
opensensemap|osmap				**OpenSenseMap client**
	set			**Settings**
		gm N		G-M tube, N=1..2
			senseboxid|sbid "..."	SenseBox ID
			sensorid|sid "..."	Sensor ID

COMPONENT LIST



Resistors

R1 = 47Ω
R2-R5 = 330Ω
R6 = 3.3kΩ
R7-R12 = 4.7kΩ
R13 = 5.6kΩ 1%
R14-R18 = 10kΩ
R19-R22 = 27kΩ
R23,R24,25 = 100kΩ
R26 = 470kΩ*
R27,R28 = 1.5MΩ 1.6kV
(e.g. Vishay HVR2500001504JA100)*
R29,R30 = 4.7MΩ 1.6kV
(e.g. Vishay VR25000004704FA500)*
R31 = 10MΩ 1.6kV
(e.g. Vishay VR25000001005FA500)*
P1,P2 = trimmer 10kΩ
(e.g. Vishay T73YU103KT20)

Capacitors

C1,C2,C3 = 330pF NP0/COG
C4-C14 = 22nF 250V film capacitor
(e.g. Epcos B32529C3223J000)
C15-C18 = 100nF X7R
C19A,C20A = 100nF 630V film capacitor
(e.g. Epcos B32671P6104K000)*
C19B,C20B = 100nF X7R 1.5kV
(e.g. Kemet C2225C104KFRACTU)*
C21,C22 = 100µF 25V
C23 = 220µF 16V, low ESR

Inductor

L1 = 47µH, radial
(e.g. Würth Elektronik 744 772 047 0)

Semiconductors

D1,D2 = 1N4148
D3-D13 = BAV21
D14, D15 = 1N5819-B
D16,D17,D18 = SA5.0A
LED1 = LED 3mm red
T1-T5 = BC547B
T6,T7 = 2N7000
T8 = IRLD110PBF*
IC1 = ICM7555
IC2 = CD4093N
IC3 = P82B715P
IC4 = Step-down regulator Vin 8-28V, Vout 5V, 1A, Würth Elektronik 173 010 578

Miscellaneous

BZ1 = Sound transducer
(e.g. Loudity LD-BZEG-1205/3)
F1 = polyfuse 500mA
(e.g. Littlefuse 60R050XPR)
K1: DC socket 2.1/5.5mm (e.g. Ninigi PC-GK2.1)
K2: 2-way PCB screw terminal block, 5mm pitch
MOD1A,MOD1B,MOD2 = 20-way pin socket, 0.1" pitch, bottom entry
(e.g. Würth Elektronik 613 020 157 21)
S1,S2,S3 = pushbutton
(e.g. C&K Components D6R10LFS)

When the conversion factor *k* is known, the dose rate *D* is:

$$D = k \times R_{corrected}\ [\mu Sv/h]$$

Assembling the environmental monitoring system

As usual, start with mounting the smallest components like the BAV21 diodes (D3-D13) and work your way up to the tallest parts. Mount all the parts except the LCD and the ESP32 module. Insert IC1 in its socket but do not insert IC2 and IC3 yet. Mount either capacitors C19A & C20A or C19B & C20B; do not mount all four. TVS diodes D16 and D17 have a tight fit on the PCB so bend the leads of D17 with care and make sure D16 is not in the way of RJ45 jack SENSOR1.

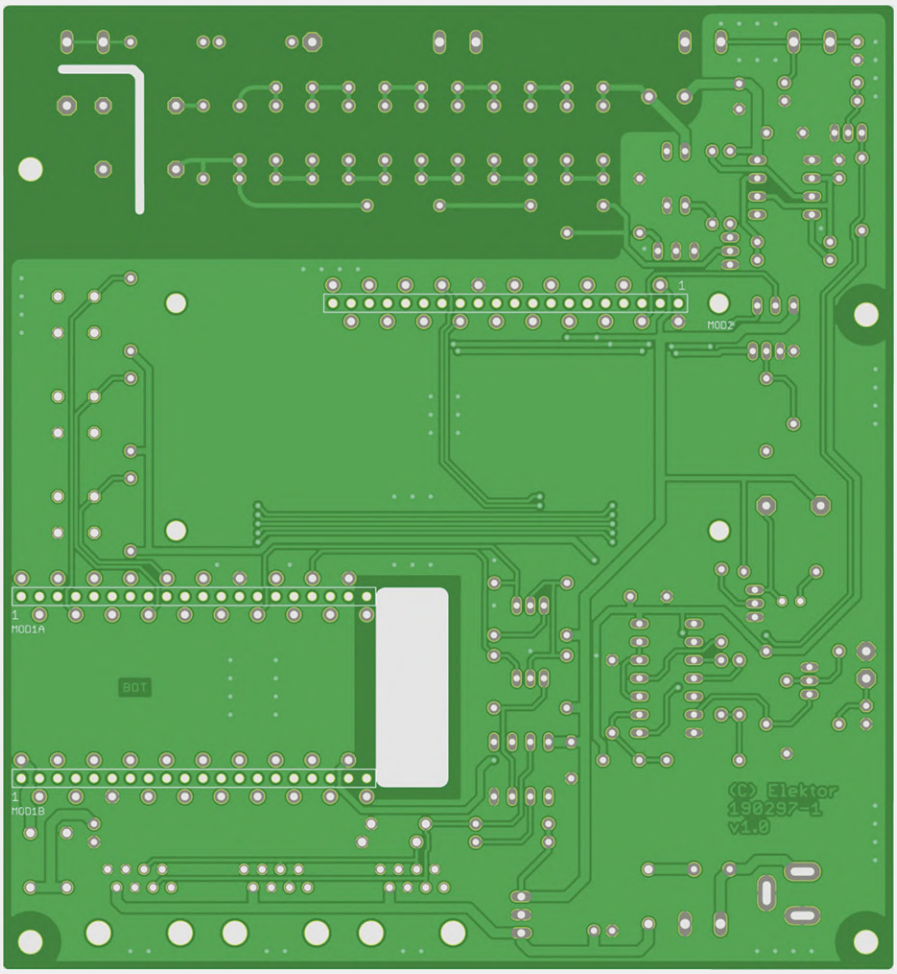
To reduce the total height of the assembled PCB, the LCD and ESP32 module are mounted using bottom-entry sockets (MOD1A, MOD1B & MOD2) at the solder side of the PCB (**Figure 6**). You can trim the leads of the MOD2 header a bit, so

@ WWW.ELEKTOR.COM

→ Environmental Monitoring System, Bare PCB
www.elektor.com/190297-1

→ Environmental Monitoring System, Kit of Parts
www.elektor.com/190297-71

→ Environmental Monitoring System, Laser-cut enclosure
www.elektor.com/190297-72



SENSOR1-SENSOR3 = RJ45 jack
(e.g. Molex 85503-5001)
LCD 2x16 alphanumeric module, I²C,
Raystar Optronics RC1602B5-LLH-JWV
ESP32 devkitC with header pins
For G-M tube: 2× 5mm fuse holder clip

(*) see article text.

shocking experience is guaranteed!
If the supplies are correct, turn off power and install IC2, IC3, the LCD, and the ESP32 module. You can now also connect the G-M tube (check the polarity). Turn power on again and connect the system with a USB cable to a computer that has the Arduino IDE installed (we used version 1.8.9). Compile the sketch and upload it to the ESP32 module. Alternatively, you can upload the binary files directly to the ESP32.
Adjust contrast trimmer P2 to make the text on the LCD module legible.
Use pushbuttons 'Up' and 'Down' to select a sensor (currently only G-M tube 1 and G-M tube 2), press 'Select' to alternate the displayed value between µSv/h and CPM. You can also turn the buzzer and LED on or off and reset a radiation alarm, and display the date and time from an NTP server and select a time zone. Finally, you can connect your Environmental Monitor System to an IoT platform.You should now also be able to connect to it via the serial port.

Two Tubes

Up to two G-M tubes can be connected if both tubes can operate at the same voltage. If you intend to connect a tube via a cable, keep it as short as possible. Use twin-shielded cable and connect the anode resistor directly to the G-M tube. As the high voltage on the cable can now bite when touched accidentally, you can also opt to use two resistors in series, one on the PCB and one connected to the G-M tube. ⏮

190297-01

they won't touch anything on the underside of the LCD module.
Set the I²C address of the LCD with a drop of solder on pads 1 & 2 of jumpers SJ1 and SJ2.
If, like the one in the parts list, your LCD provides a negative contrast voltage (V_{EE}) on pin 15, leave solder jumper SJ3 open and short pads 1 & 2 on SJ4.

Testing

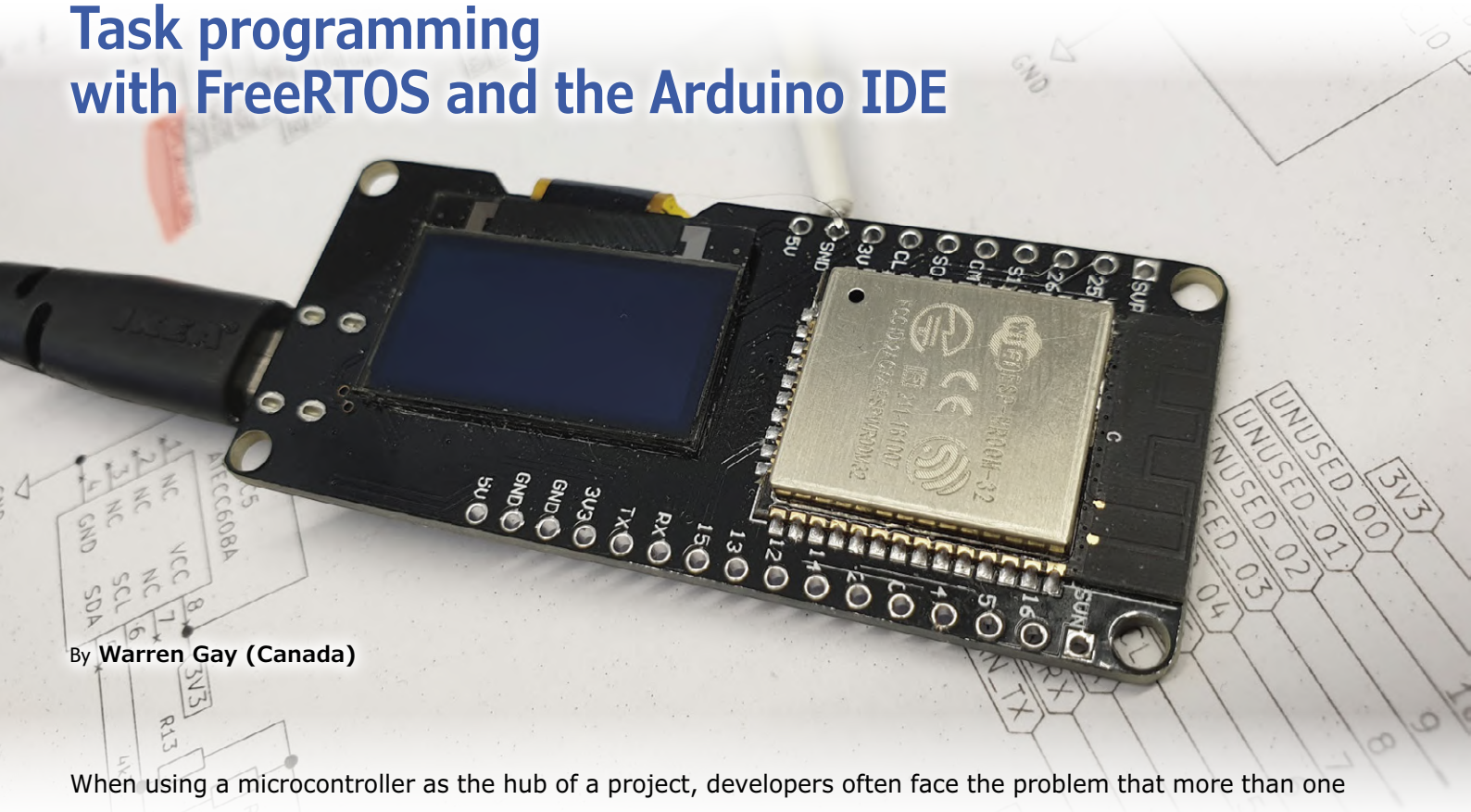
Set trimmer P1 midway and connect a 9 to 20 V_{DC} power supply to K1 or K2. Make sure to use a proper power supply as the circuit can draw quite some current (about 500 mA when powered from 9 VDC) when starting. Check the 5 V supply. Measure the voltage across C19 and turn P1 to adjust it to a value suitable for your G-M tube (400 V for an SBM-20). Be very careful when doing this; touching the high voltage is not lethal, but a

Web Links

[1] Radioactivity is in the Air for You and Me - bionerd 23 @ EHSM - v4: www.youtube.com/watch?v=6Z5mRpFdT5I
[2] Cosmic-ray air showers: www.mpi-hd.mpg.de/hfm/CosmicRay/Showers.html
[3] Radioactive household items: www.youtube.com/watch?v=XTIvoTiTTSU
[4] Geiger-Müller (G-M) Tubes: www.orau.org/ptp/collection/GMs/GMs.htm

Practical ESP32 Multitasking

Task programming with FreeRTOS and the Arduino IDE



By **Warren Gay (Canada)**

When using a microcontroller as the hub of a project, developers often face the problem that more than one task needs to be performed at a time: Scanning sensor values, controlling actuators, visualizing device states and/or waiting for a human user to enter. Fortunately, this problem can be solved in a very elegant way with task programming based on lightweight embedded operating systems. FreeRTOS is an open source and widely used operating system, available for many microcontroller platforms. The very popular tandem ESP32 and Arduino IDE makes it especially easy to use FreeRTOS for task programming, as it is already integrated into the core libraries. In fact, you may have always used FreeRTOS without knowing it!

The Espressif ESP32 platform is an exciting microcontroller for maker projects. Its hardware capabilities, extensive software and affordability place it in the "must have" category for many. Espressif has provided both Arduino compatible and native development environments. Their native mode is officially known as ESP-IDF (Espressif IoT Development Framework). For this article, the familiar Arduino environment will be used. Most of the API (Application Programming Interface) is available to both [1]. Some of our readers may know that Espressif uses the popular open source embedded operation system FreeRTOS, to implement the ESP32 library functions for WiFi, Bluetooth and many more features of the microcontroller. In this article, we'll see that FreeRTOS [2] and task programming is also used for the simple Arduino `setup()` and `loop()` functions. First, we will describe a simple ESP32 demo project [3]. Then we will have a look under the hood; later we will establish our own tasks for the demo.

Application

For this installment in the series we'll develop an application that reads the analogue to digital converter (ADC) from a potentiometer and display it on an OLED in bar graph form. In addition we'll illuminate an LED using pulsewidth modulation (PWM), varying it from dim to bright.

This article will use the Lolin ESP32 board with the builtin OLED (**Figure 1**) [4]. If you have another ESP32 board but lack the SSD1306 compatible OLED, you can disable the OLED display in the program and rely on the brightness of the LED instead. The LED and the potentiometer will be wired up according to the schematic in **Figure 2**.

As the application runs, turning the potentiometer will vary the input voltage seen on GPIO36. This voltage will be measured as a 12-bit value, returning a value from 0 to 4095. Turning the potentiometer (or "pot") fully clockwise will return a value of 4095 which will cause full brightness of the LED. If the opposite

happens, then reverse the outer lead connections on the pot. The centre connection is the pot's wiper arm and will vary in voltage from ground potential up to the full +3.3 V. Be careful not to wire the pot to the +5 V supply, as that exceeds the GPIO input maximum voltage.

Program configuration

Let's examine the software now. The program has C macros defined to make it easy for you to reconfigure it. These are shown in **Listing 1**. Set macro `CFG_OLED` to zero, if you are not using a display (this causes the address and I²C macros to be ignored).
Macro `CFG_ADC_GPIO` value has been chosen to use ADC1 on channel 0 (GPIO36). Finally, the LED has been configured by `CFG_LED_GPIO` to use GPIO13.
The full source code is available from github in the *freertos-tasks1* subdirectory [3].

Initialization

Ignoring tasks for the moment, the device `setup()` for our application is as shown in **Listing 2**. This would be a typical initialization in the Arduino framework.
The display is only initialized when it is compiled in (configured). After that, using the Arduino routine `analogReadResolution()`, the ADC is configured to read 12-bit values. The function `analogSetAttenuation()` is called so that it will read a value from 0 to +3.3 V. The ADC includes an internal amplifier, so it is important to configure it to use the correct range.
After that, the GPIO for the LED (`gpio_led`) is configured as an output, and configured for PWM by calls to `ledcAttachPin()` and `ledcSetup()`.

Listing 1. Configuration Options.

```
// Set to zero if NOT using SSD1306
#define CFG_OLED            1

// I2C address of SSD1306 display
#define CFG_OLED_ADDRESS    0x3C

// GPIO for display I2C SDA
#define CFG_OLED_SDA        5

// GPIO for display I2C SCL
#define CFG_OLED_SCL        4

// Pixel width of display
#define CFG_OLED_WIDTH      128

// Pixel height of display
#define CFG_OLED_HEIGHT     64

// GPIO for ADC input
#define CFG_ADC_GPIO        36

// GPIO for PWM LED
#define CFG_LED_GPIO        13
```

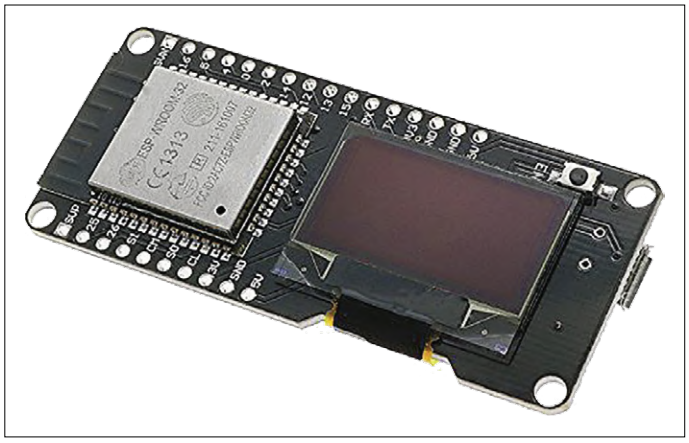


Figure 1: The Lolin ESP32 with OLED Display.

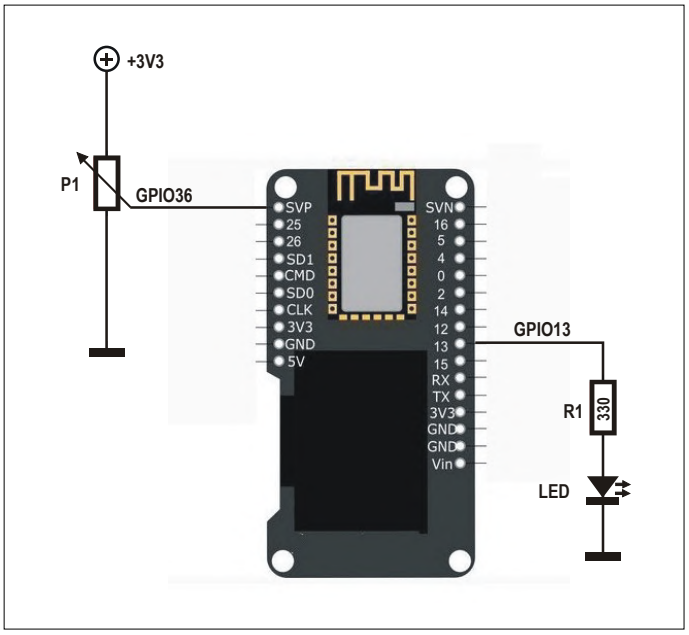


Figure 2: Lolin ESP32 module equipped with potentiometer and LED.

Listing 2. The setup() Function.

```
void setup() {

#if CFG_OLED
  display.init();
  display.clear();
  display.setColor(WHITE);
  display.display();
#endif

  analogReadResolution(12);
  analogSetAttenuation(ADC_11db);

  pinMode(gpio_led,OUTPUT);
  ledcAttachPin(gpio_led,0);
  ledcSetup(0,5000,8);
}
```

Listing 3. The Bar Graph Display Function.

```
#include "SSD1306.h"
...
void barGraph(unsigned v) {
  char buf[20];
  unsigned width, w;

  snprintf(buf,sizeof buf,"ADC %u",v);
  width = disp_width-2;
  w = v * width / 4095;
  display.fillRect(1,38,w,disp_height-2);
  display.setColor(BLACK);
  display.
fillRect(w,38,disp_width-2,disp_height-2);
  display.fillRect(1,1,disp_width-2,37);
  display.setColor(WHITE);
  display.drawLine(0,38,disp_width-2,38);
  display.
drawRect(0,0,disp_width-1,disp_height-1);
  display.setTextAlignment(TEXT_ALIGN_CENTER);
  display.setFont(ArialMT_Plain_24);
  display.drawString(64,5,buf);
  display.display();
}
```

Bar graph display

Using the SSD1306 OLED device, the application will draw a bar graph on it and display the ADC value (see **Listing 3**). The `barGraph()` function accepts an input value `v` which is the ADC value, and formats that into a short message using the array buf, using `snprintf()`. The some rectangles are drawn in black and in others in white to represent the bar graph. The `display.drawString()` method call places the formatted text in the display also. At the end, the `display.display()` method is called to put the in-memory image of the display into the OLED controller for the device.

The loop() function

If we were writing a normal Arduino program, we would code a loop like the one in **Listing 4**.

Listing 4. Typical Arduino Loop Code.

```
void loop() {
  uint adc = analogRead(ADC_CH);

#if CFG_OLED
  barGraph(adc);
#endif

  printf("ADC %u\n",adc);
  ledcWrite(0,adc*255/4095);
  delay(50);
}
```

Table 1. Tasks running at Arduino Startup.

Task Name	Task #	Priority	Stack	CPU
loopTask	12	1	5188	1
Tmr Svc	8	1	1468	0
IDLE1	7	0	592	1
IDLE0	6	0	396	0
ipc1	3	24	480	1
ipc0	2	24	604	0
esp_timer	1	22	4180	0

In this code, the steps are simple:

- Read the 12-bit ADC value (gets a value from 0 to 4095).
- Display it on the bar graph (when configured).
- Print “ADC” and the value to the serial monitor.
- Set the LED brightness by changing the PWM parameter.
- And delay for 50 ticks.

This is an intentionally simple program. But if this application was complicated, you would want to break it up into smaller “tasks”.

What are tasks?

When you have a dual core ESP32 CPU, it is possible to have two programs running **simultaneously**. This is exciting in a micro-controller! Each CPU operates with its own program counter and other registers to carry out instructions. This represents two **threads** of control. A single core CPU on the other hand, would only perform one thread of control at any given instant. To allow more than two programs to execute **concurrently**, a scheduling trick is used to suspend programs while resuming others. This is the main job of FreeRTOS. This scheduling is done so quickly that it has the **appearance** of running several programs at once. But at any given instant in time, the dual core CPU has no more than two programs executing simultaneously. Think of this loosely used term “program” as a task. To manage running several concurrent tasks, the FreeRTOS scheduler saves the registers of the current task and restores the registers of the next task to be run. In this way, multiple tasks can run independently. In addition to the program counter register for each task, its **stack pointer** must also be saved and restored. This is necessary because C/C++ programs have variables and return function addresses saved on the stack. Each task must have its own private stack.
On the ESP32, all tasks run within the same memory space (unlike the Raspberry Pi, for example). For this reason, tasks must be programmed carefully so that they do not corrupt the memory used by other tasks. Additional considerations apply when you have a multi-CPU core operating, but let’s save that discussion for another time.

Arduino tasks

When your ESP32 Arduino program begins, is it in a task? Absolutely! In addition to your own task at startup, there are other FreeRTOS tasks executing. Some of these tasks provide services such as timers, TCP/IP, Bluetooth etc. Additional tasks may be started as you request services from from the ESP32. **Table 1** shows an example of FreeRTOS tasks running when

Arduino functions `setup()` and `loop()` are invoked. The task named *loopTask* is your main Arduino task.
The column labeled *Stack* represents FreeRTOS unused stack bytes. How to plan for stack sizes will be covered at a later time. The chart is sorted in reverse chronological order, by task number. Your main task (*loopTask*) is the last task created. Missing task numbers suggest that some tasks were created and ended when their job was completed. The priority column illustrates the task priorities assigned, with zero representing lowest priority. For now, just know that priorities operate a little differently in FreeRTOS than they do in Linux for example. Finally, the tasks are divided between CPU 0 and CPU 1. Espressif places their support tasks in CPU 0, while application tasks use CPU 1. This keeps services for TCP/IP and Bluetooth etc. running smoothly without any special consideration from your application. Despite this convention, it is still possible for you to create additional tasks in either CPU.

Arduino startup

It's good to know how ESP32 Arduino programs initialize before `setup()` is called. Consider the simplified program snippet in **Listing 5** (the watchdog timer elements have been left out). From this example, we can note some interesting things:

- Note that this is a C++ startup (hence the extern "C" declaration of `app_main()`).
- Some Arduino initialization is performed by `initArduino()`.
- The *loopTask* is created and run by the call to `xTaskCreatePinnedToCore()`.

The loopTask is created by calling `xTaskCreatePinnedToCore()`, with a number of arguments. The first argument is the address of the function to be executed for the task (*loopTask*). When the task is created, the function `loopTask()` then runs invoking `setup()` first and then `loop()` from a forever "for" loop.
The second argument is a string describing the name of the task as a C string (`"loopTask"`). The stack size argument three is specified as 8192 *bytes*. Note that this differs from the stock FreeRTOS on other platforms, which use 4-byte words instead. When the task is created, this is allocated from the heap and assigned to the task before the function is called. This is a fair bit of SRAM that would be wasted if the main task was not used.
All FreeRTOS tasks accept one void pointer as an argument. In this case the value is not used and is supplied as NULL. The fifth argument is the FreeRTOS priority to assign to the task, which is 1 in this example. This requires more discussion, so use 1 until FreeRTOS priorities can be explained.
After the priority argument is a pointer to a variable that will receive the handle to the task. In this case, it is not used and could have been supplied as NULL. The last argument specifies which CPU core for it to run on. It can be 0 or 1, for the dual ESP32. CPU 1 is used to start the main task. For single core devices, this can only be zero.

Creating a task

Let's pretend our application is complicated and that we need to break the code into two tasks. We already have the `loopTask()`, which calls `loop()` over and over. To take advantage of the stack space allocated to it, we would normally code our most stack intensive part of the program there.

Listing 5. Simplified ESP32 Arduino Startup.

```
void loopTask(void *pvParameters) {
  setup();
  for (;;) {
    loop();
  }
}

extern "C" void app_main() {
  initArduino();
  xTaskCreatePinnedToCore(
    loopTask,       // function to run
    "loopTask",     // Name of the task
    8192,           // Stack size (bytes!)
    NULL,           // No parameters
    1,              // Priority
    &loopTaskHandle, // Task Handle
    1);             // ARDUINO_RUNNING_CORE
}
```

For our task demonstration, let's perform the following in the `loop()` function:

- Read the 12-bit value from the ADC.
- Print the value to the serial monitor.
- Send the ADC value to the second task.
- Delay 50 ticks and return (from `loop()`).

This leaves the second task to do:

- Get the ADC value from the `loop()` function (in the *loopTask*).
- Display the ADC value on the bar graph (when configured).
- Modify the PWM control for LED brightness.

The only sensible place to create the second task is in the `setup()` task because this creation only needs to be performed once. So let's add that to our `setup()` task:

```
void setup() {

#if CFG_OLED
  display.init();
  display.clear();
  display.setColor(WHITE);
  display.display();
#endif

  analogReadResolution(12);
  analogSetAttenuation(ADC_11db);

  pinMode (gpio_led,OUTPUT);
  ledcAttachPin(gpio_led,0);
  ledcSetup(0,5000,8);

...
```

```
  xTaskCreatePinnedToCore(
    dispTask,   // Display task
    "dispTask", // Task name
    2048,       // Stack size (bytes)
    NULL,       // No parameters
    1,          // Priority
    NULL,       // No handle returned
    1);         // CPU 1
}
```

This call will start a new task named *dispTask*, with a stack of 2048 bytes. We have assigned it to CPU 1, with a priority of 1. It will be executing function `dispTask()` (yet to be defined). Since the task will run on the same CPU and at the same priority as our `loopTask()`, the CPU will be shared with the `loopTask()` and any other priority 1 tasks on that CPU.

Queues

There is still one missing ingredient — how do we send the ADC value from one task to another? You might try to design something in memory, which might work for tasks on the same CPU. But things get more complicated when passing information from one CPU to another. The FreeRTOS queue is the solution to our problem.

The queue allows one task to push an item into it in an "atomic" fashion. It likewise allows the receiver to fetch that item in an atomic way. By atomic, I mean that the item cannot be partly pushed or partly received. With a dual core CPU with two instructions executing simultaneously, there are timing issues and race conditions. The queue mechanism takes special measures to make this happen atomically.

To create a FreeRTOS queue, see the listing below:

```
static QueueHandle_t qh = 0;
...
  qh = xQueueCreate(8,sizeof(uint));
```

The first argument is the depth of the queue (the maximum number of queued entries that it can hold). The second argument specifies the size of each queued item. In this case it is the size of a `uint` value in bytes. The return value is the queue's handle. This step should be performed immediately before the creation of the *dispTask* in `setup()`, so that it is immediately available for it.

Now let's turn our attention to feeding that queue:

```
void loop() {
  uint adc = analogRead(ADC_CH);
  printf("ADC %u\n",adc);
  xQueueSendToBack(qh,&adc,portMAX_DELAY);
  delay(50);
```

The bar graph and LED updates will be done in our new function `dispTask()`, yet to be described. The `loopTask()` however, reads the ADC value into adc and then prints it to the serial monitor. After that is done, it is pushed on to the *back* of the queue using the function `xQueueSendToBack()`. We specify the queue handle in the first argument. The value to be queued is provided using a pointer in the second argument. The last argument indicates how long to wait if the queue is full. By specifying the macro `portMAX_DELAY`, we indicate that we want the execution to block until there is room in the queue.

The display task

Now lets examine the display task in the following code:

```
void dispTask(void *arg) {
  uint adc;

  for (;;) {
    xQueueReceive(qh,&adc,portMAX_DELAY);
    barGraph(adc);
    ledcWrite(0,adc*255/4095);
  }
}
```

This function, like all task functions receives a pointer argument when it starts. Here it is unused and is NULL because of the way that the task was created.

The main body of the task is a for loop receiving from the queue. Here `xQueueReceive()` returns the queued data value, but otherwise waits forever if the queue is empty (the third argument value is `portMAX_DELAY`). Once a value is received from the queue, the bar graph display is updated and the LED PWM value is changed.

Note that there is no need to call `delay()` in this display task. The reason is that the task will automatically stall waiting in function `xQueueReceive()` when the queue is empty. The pace of the execution will be set by the sending task.

Running the demo

When wired up and flashed, you should see the bar graph display and the LED light (see **Figure 3**). If the LED is dark, turning the pot clockwise should brighten it. If you compiled the program without a display, then startup the serial monitor

Web Links

[1] ESP32 API reference: https://docs.espressif.com/projects/esp-idf/en/latest/api-reference/

[2] FreeRTOS homepage: www.freertos.org

[3] Project source code: https://github.com/ve3wwg/esp32_freertos/blob/master/freertos-tasks1/freertos-tasks1.ino

[4] ESP32 Lolin Board with OLED: www.elektor.com/lolin-esp32-oled-module-with-wifi

and look for lines of output of the form “ADC 3420” etc. Turning the pot counter clockwise should result in the dimming of the PWD LED and clockwise should brighten it. Likewise, counter clockwise should display reduced ADC values and clockwise increased values.

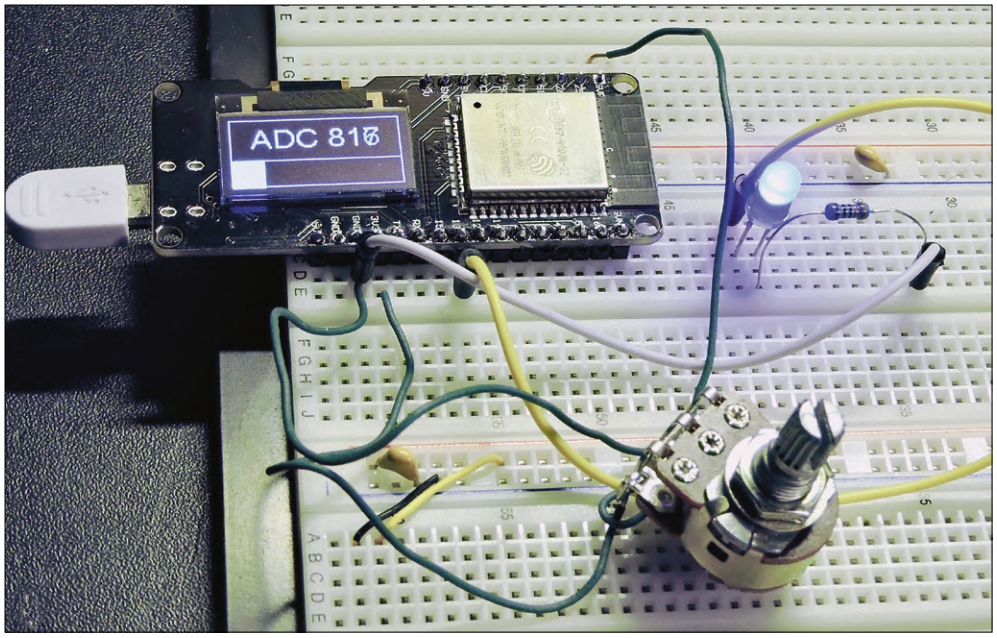


Figure 3: Running the demo.

Summary

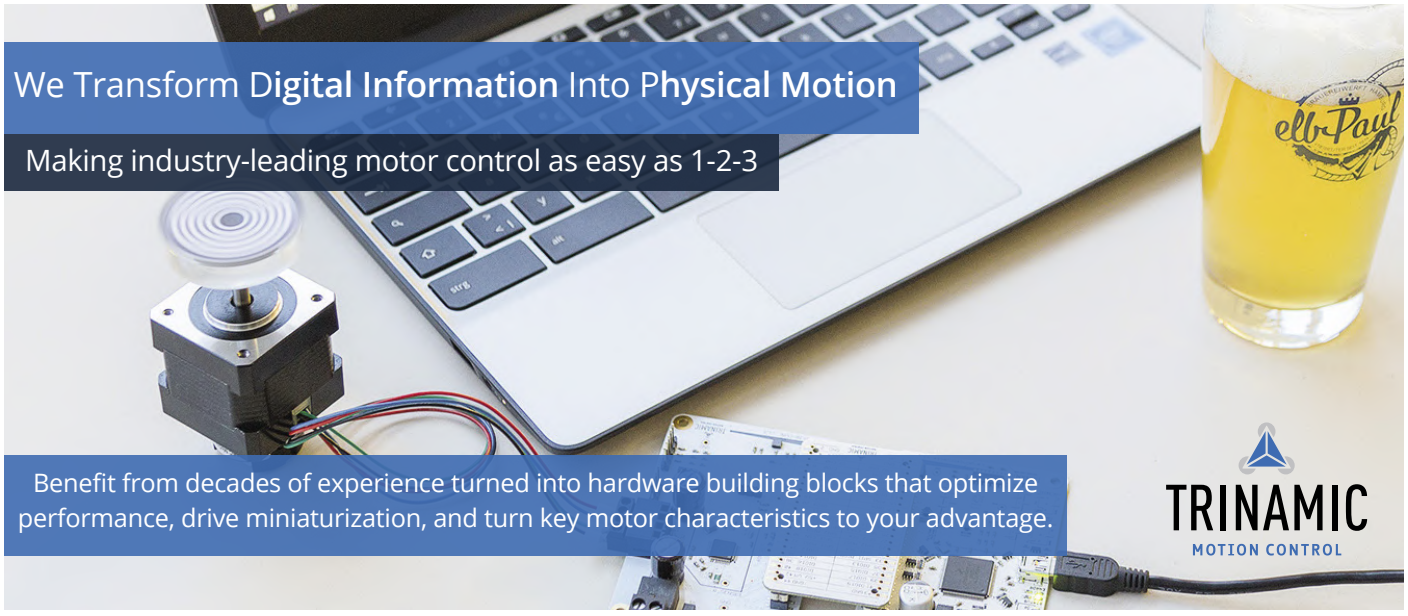
We’ve covered a lot of ground in this article. You’ve seen the ESP32 startup procedure from `app_main()`, to `setup()` and `loop()`. The creation of the main task named `loopTask` was covered. In the demo we created our own additional task for the purpose of controlling the OLED and PWM LED, and communicated data to it using a queue. This code runs nicely independent from the main task.

Additionally, we’ve made use of the main task knowing that a fair amount of stack space is allocated to it (from the heap). In a future instalment, we’ll discuss how to determine stack usage for new tasks that you want to create. This example was trivial in terms of complexity. But recognize that tasks make complicated applications simpler by subdividing the problem. Consider how a MIDI controller might be divided into sending, receiving and control tasks. The receiving task is then free to receive incoming serial data and decode them into events for the control task to execute. Some control events may in turn result in sending serial data from the sending task. This is a neat subdivision of labour. Not only does this make the maintenance of the code a joy but it improves upon program correctness. ⏮

190182-01

@ WWW.ELEKTOR.COM
➞ Lolin ESP32 OLED Display Module
www.elektor.com/lolin-esp32-oled-module-with-wifi

Advertisement



Interactive

Corrections & Updates || Questions & Answers

By **Clemens Valens**

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

9-Channel Relay Switcher Board Q&A

After publishing this project in the May & June 2019 edition a few questions came in about this versatile project. Here they are, along with with hopefully satisfying answers.

Q: The Component List states R3, R4 and R6 as 0 Ω devices, but the schematics shows them having a value of 1 kΩ, 1 kΩ and 10 Ω respectively. Which is correct?

A: The parts list and schematic are both correct in a way. These resistors are intended to provide current limiting for the microcontroller ports to which they are connected in case a less careful user plugs something on the extension connectors without first removing the supply voltage. The exact value of these resistors is unimportant as long as it is not too high. In systems where the USB-to-serial module is always present, replacing these resistors by a piece of wire is fine too.

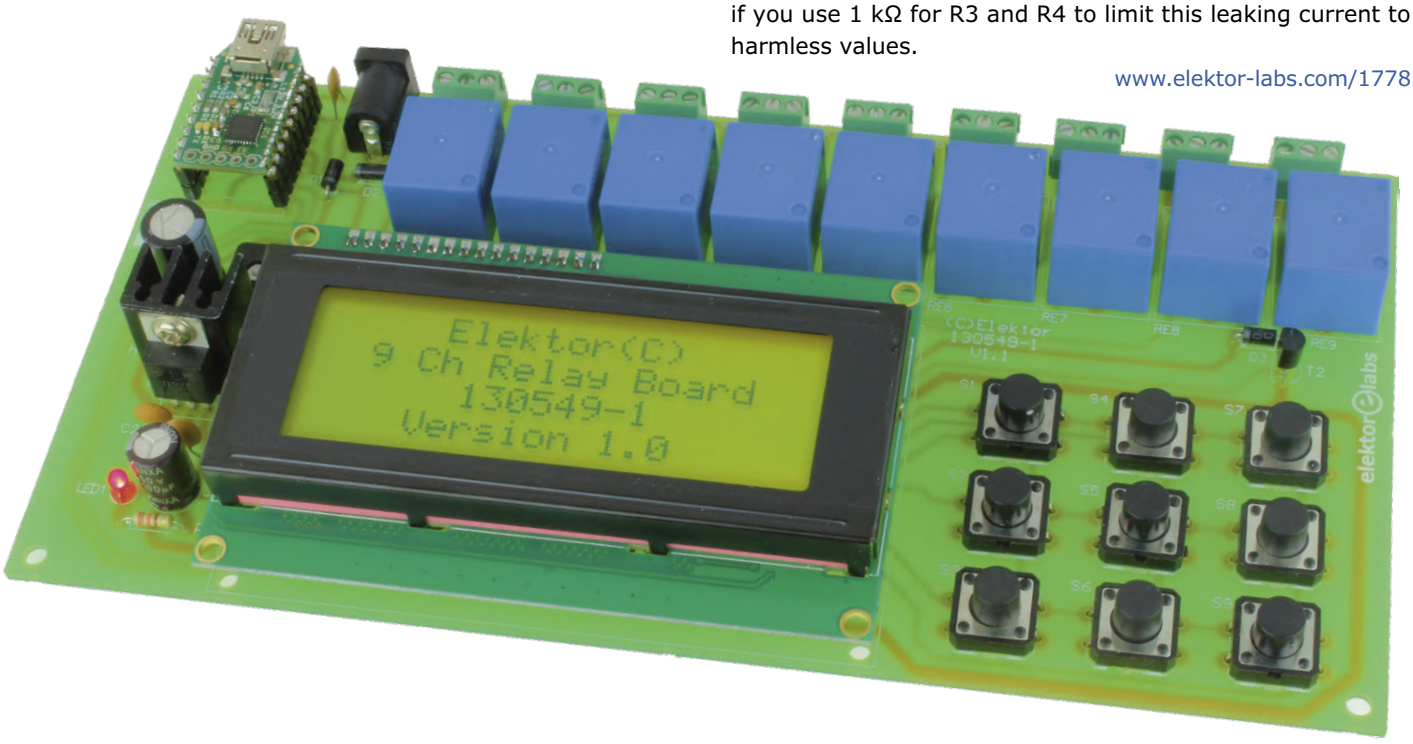
Q: When programming the PIC microcontroller, I received the error "*At address 300006 Expected Value 81 Received Value 85*". Replacing the MCU or programming it on a breadboard gives the same result. I use a PICkit 3 clone and the MPLAB IPE software which has worked fine in other projects. Strangely, although the final message is that programming failed, the board appears to work fine. What does this error mean?

A: It means the programmer cannot clear bit 2 at MCU address 300006. It is completely unrelated to the 9-Channel Relay Switcher Board. The offending bit is the LVP bit of the PIC's CONFIG4L register. Searching the Internet reveals several posts from people having the exact same problem. Suggested solutions include high-voltage programming, a bulk erase or using another programmer.

Q: When the board is disconnected from the 12 V supply but connected to a powered USB port, LED1 flashes at reduced brightness. An oscilloscope shows a roughly triangular waveform on the 5-volt supply line, rising from 1.4 V to 1.9 V at a frequency of around 8.5 Hz. Using a separate USB-to-serial converter, or bypassing the USB-to-serial module give the same result. D2 checks OK and after double-checking everything else I can find no errors in the construction. What is going on?

A: The USB-to-serial module is not supposed to power the circuit because it cannot power the relays. D2 is present to enforce this. Under normal circumstances it is IC1 that powers the USB-to-serial module, not the other way around. What you observe is probably current leaking through the RXD/TXD lines from the USB-to-serial module to the MCU's ports. Those ports have internal protection diodes that connect to the power supply pin of the MCU. This pin now charges C4 which makes LED1 light up when the voltage is high enough. This in turn C4 discharges slightly, causing LED1 to shine less brightly and allowing C4 to be charged again, and so on. The result is an oscillation like the one you see. There is no problem, especially if you use 1 kΩ for R3 and R4 to limit this leaking current to harmless values.

www.elektor-labs.com/1778



Static IP address for the ESP32-based NTP Server

In the July & August edition of 2019, we published a small Network Time Protocol (NTP) server to provide time (real or customized) on a network. It used DHCP to obtain an IP address, which is practical in many cases but not always, like when there is no DHCP server available on the network. For this and other reasons, we decided to add the option to set a static IP address for the NTP server. Doing so is simple: just tick the box 'Use static IPv4' on the 'IPv4 Settings' page and fill in the desired IP address with subnet mask.

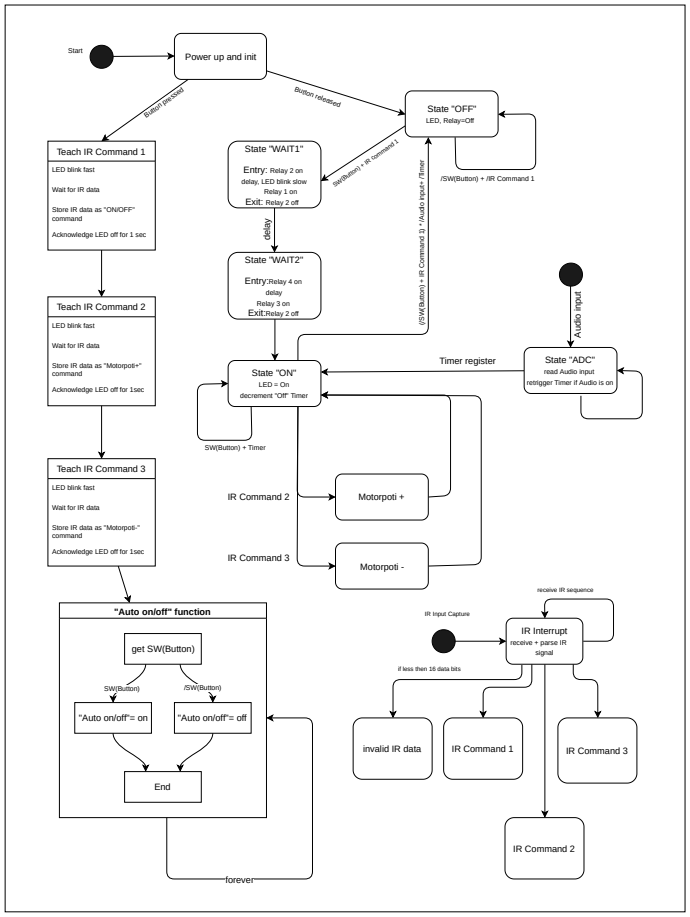
Soft Start for Amplifiers

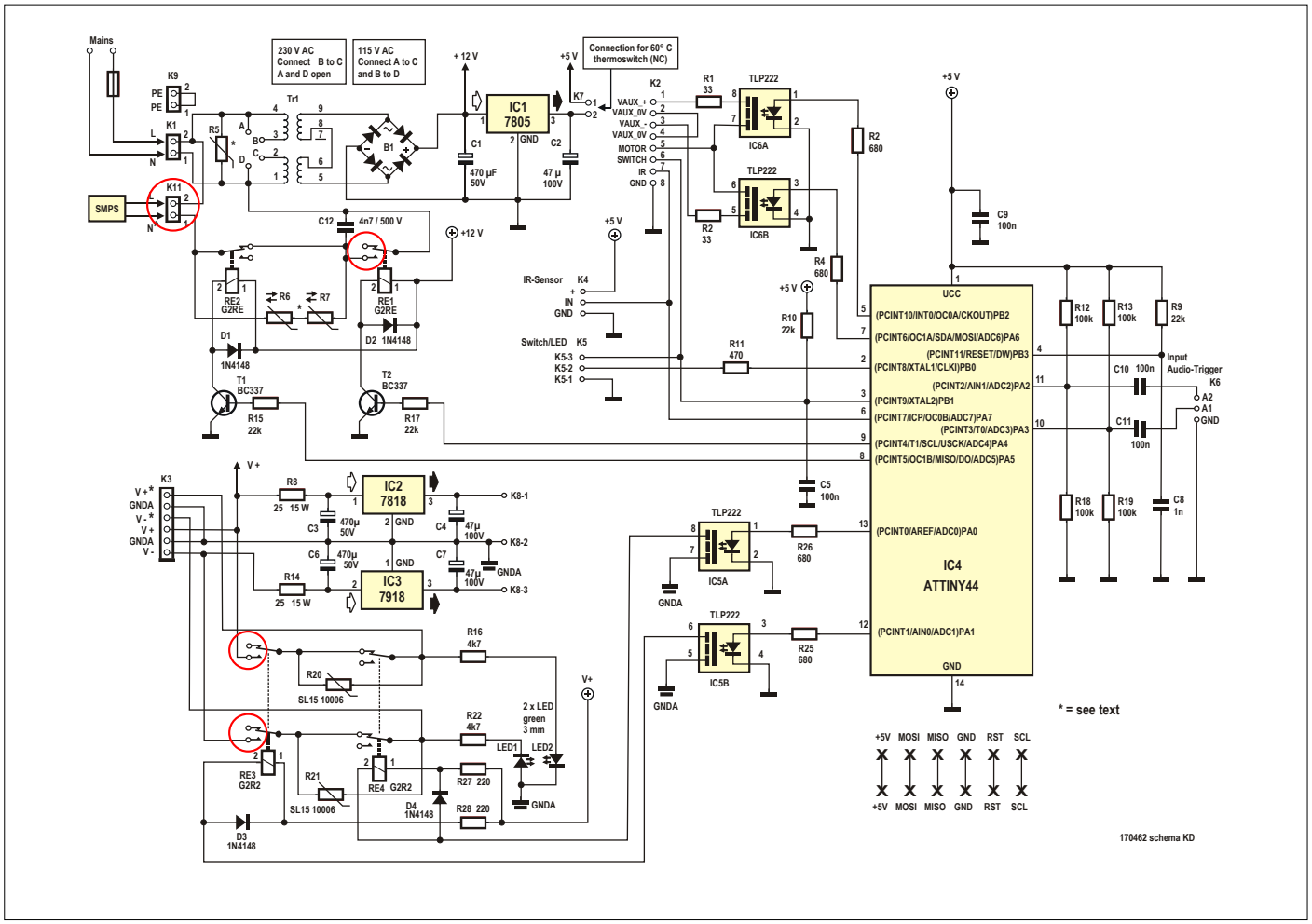
After publishing this project in the October & November 2019 edition, some mistakes were discovered in two drawings. First, the schematic in Figure 4 shows the normally-open (NO) and normally-closed (NC) contacts of relays RE1 and RE3 swapped. The labels and wiring of terminal K11 were also erroneous; the corrected schematic diagram is shown here.

The second problem is state diagram Figure 6, for which an old version was printed. The correct version is shown here.

In IR learning mode, the LED blinks while the software waits for an IR signal. Reception will be acknowledged by the LED switching off for one second.

In the formulas for calculating R_{NTC} and total energy, the Ω sign was lost in print. The complete calculations can be found in the project documentation at www.elektor-labs.com/1343





Automotive Headlight Tuning

Legal, illegal — it matters!

By **Dr Thomas Scherer** (Germany)



Better lights are on the wish list of many car drivers whose vehicles are not equipped with xenon or LED lamps by the manufacturer. After all, halogen lamps are yellowish by comparison, less bright, and in some cases do not last long. Something can be done about this — and quite legally.

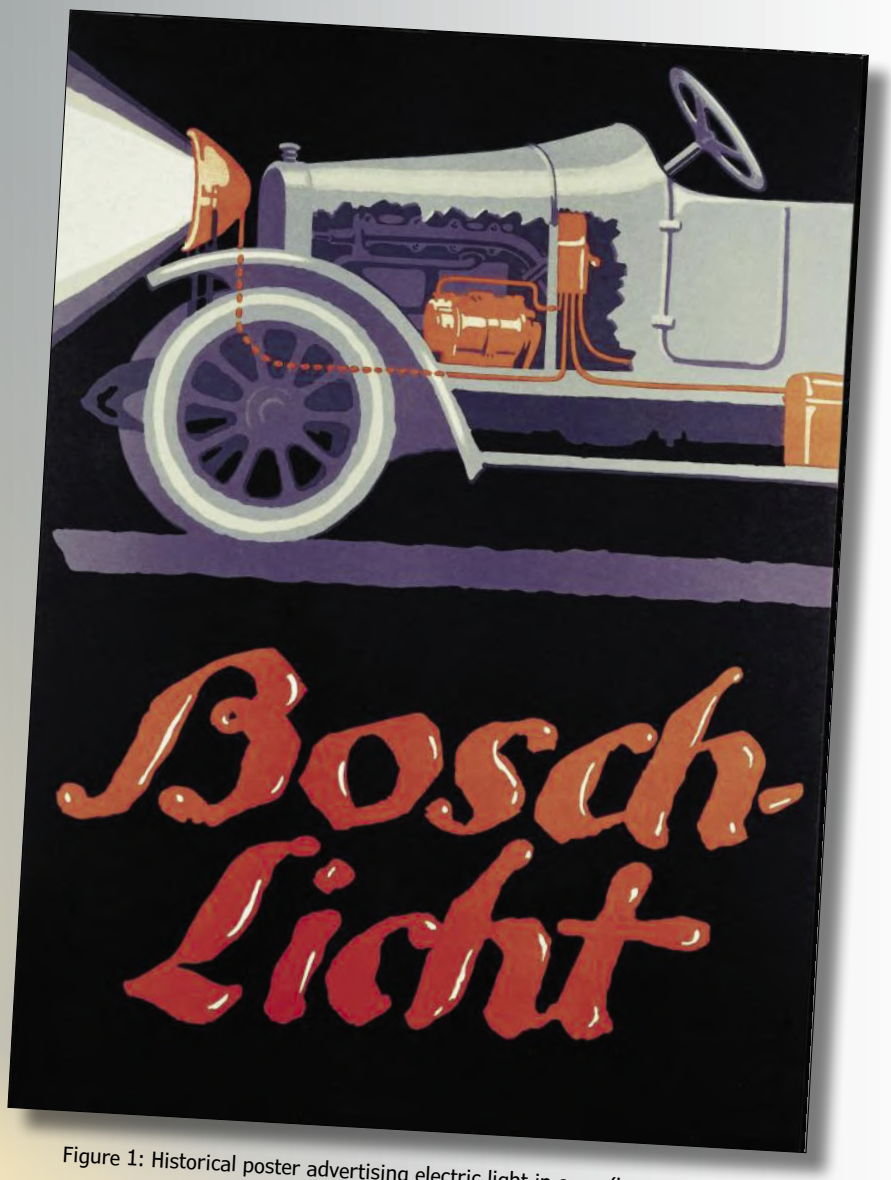


Figure 1: Historical poster advertising electric light in cars. (image: Bosch)

Better lights are on the wish list of many car drivers whose vehicles are not equipped with xenon or LED lamps by the manufacturer. After all, halogen lamps are yellowish by comparison, less bright, and in some cases do not last long. Something can be done about this — and quite legally.

The first cars produced at the end of the 19th century had four wheels, an engine, a steering wheel and brakes. The lighting did not differ from competitors with organic horsepower: just as in carriages, there were lamps with candles or carbide. These dim lights were replaced by electric light for the first time in a Cadillac back in 1911. As early as 1913, Bosch perfected mobile power generation by using a battery with a generator equipped with a controller (**Figure 1**). Contemporaries were immediately blinded by the brighter light — modern is not always popular because of its association with 'innovation'!

So the Cadillacs got an anti-glare option from 1917. This was also considered 'major' on this side of the Great Pond and in 1921 in the German Reich, in typical German "joy of regulation", the permanent dimming of the lights was prescribed by law.

However, this was not the last word on the matter: on German roads cars were now visible, but the drivers themselves could no longer see well at night, which is why night-time accidents accumulated. So, more work and experiment was due. Separate headlamps for full beam and dipped headlights or even



Figure 2: Asian xenon retrofit kit. (image: TXVSO)

more complex systems with movable optics became up to date. In the USA, the two-filament lamp was invented in 1924, permitting both ranges with one headlight. This technology was perfected by Bosch in 1925 with their 'Bilux' lamps.

And so the situation remained unchanged for decades, even if 6-V wiring systems went out of fashion after WWII in favour of 12-V technology, which halved the necessary currents and the resulting voltage drops. Car headlights did not really improve until the introduction of brighter halogen bulbs at the end of the 1960s. But it was not until 1971 that the Mercedes Benz 350SL, a production car, was fitted with the H4-style halogen two-filament bulb.

So much for the prehistory and early history of car headlights. Today all modern cars have at least halogen light. Better things like xenon lamps or even LED headlights are currently only available for small cars and mid-range cars at full four-digit surcharges.



Figure 3: LED retrofit solution with active cooling. (image: Banggood)

Better light

In the 1990s, limousines of the upper and luxury class increasingly boasted xenon light, but this created two reactions: in addition to the 'wannahave' effect, other road users who were accustomed to yellowish halogen light often felt dazzled by the new luxury light. The fact is that xenon lamps are not only brighter but also 'blueish'. While the best 55-W halogen lamps reach a maximum of 1500 lm, the gas discharge light from xenon lamps at only 35 W can impress with up to 3,200 lm. Added to this is the colour temperature of 4200 K compared with around 3000 K for halogen. This is definitely brighter, and anyone who has ever driven a car with xenon headlights knows that this has a particularly positive effect on visibility with low beam.

And what about LEDs? Until 2008, the technically better and cheaper xenon light LEDs in the EU were approved for the taillights only. But then came the first cars with genuine LED headlights. In terms of data, they compete well with xenon lamps and are much more reliable in principle. The car industry is used to the surcharges for better light, but even today cashes in if there is a choice between halogen and LED headlamps. These costs, like the 'relative novelty', contribute to the fact that cars with halogen lamps still form the majority on the roads, even if this is changing steadily but slowly. Little wonder, then, that with halogen unpopular but technically superior, car owners experience a strong hint of room for tweaking.

Ground rules

The hint is just right. It's really obvious that you simply replace the dull halogen bulbs of the H1...H19 classes with xenon or LED retrofits that fit mechanically. After all, you don't even have to solder! The resourceful Far Eastern industry offers suitable solutions for almost all purposes. **Figure 2** shows the example of a xenon kit suitable for H8/H9/H11 sockets, which you can put together with preregulator electronics for less than €30. Kits are available not only at the usual shops like Alibaba or Banggood directly from China, but also conveniently at Amazon and other shops depending on your country. And from the same sources, replacement lights with LED technology are also available at prices that are still far below €100 for better solutions and in extreme cases even undercut 'good' halogen lamps.

But is that allowed? Unfortunately not!

While the rules are surprisingly loose in the USA and, as to be expected, in many Asian countries, we stupid Europeans are protected from too much freedom by helicopter policy. In the whole EU, it is strictly forbidden to equip headlight casings intended for halogen lamps and 'accepted' by the regulatory authorities with different light sources. Full stop.

At least in Germany, if you do, you lose the registration for your modified car and insurance companies can refuse insurance cover in the event of an accident. A pretty high risk, don't you think?

The legal situation has nothing to do with arguments, because there are good reasons for replacing halogen lamps with xenon or LED lamps. The (Asian) industry even manufactures more or less suitable LED retrofit lamps, i.e. LED lamps suitable for direct replacement of incandescent lamps. The ADAC (German Automobile Association) has investigated the use of such LED retrofits [1] and came to quite positive results. But that doesn't help — it's prohibited. As long as the legislator does

not try to create and adopt clear rules and testing standards for LED retrofit lamps, even large and reputable companies such as Osram or Philips will not care about this interesting market. This situation not only protects the profits of the car companies but also leads to an illegal scene of light upgrades and tuners, who are thus dependent on dubious products from Asian origin. Looking at the offers in the various online stores, you quickly realize that HTML is patient and that information regarding performance, luminous flux and colour temperature as well as service life are not very trustworthy.

Using these retrofits 'anyway' is not a good idea from a technical point of view. LEDs are larger than the filaments of halogen lamps at similar power. The illumination properties can therefore vary, especially for headlamps with parabolic reflectors. In the case of so-called 'projection headlights', this is not quite as tragic in principle, since an illuminated surface enclosed is projected onto the road through a lens. There are also thermal and electrical problems, as LED retrofits often lead to very limited space compared to a headlamp fitted with LEDs directly by the manufacturer. It does not only need preregulator electronics (small problem), but somehow the warmth must be dissipated off the LED chips, because LEDs and warmth do not get along well. With the appropriate performance, a passive aluminium carrier is not enough here, but it must be actively cooled by fan, as can be seen in the example in **Figure 3**. This block with radiator and fan does not only protrude at the rear and does not fit into every headlight due to the cramped construction of modern cars. Worse still: If these mini fans get dirty and blocked, the LEDs burn out quickly and under certain circumstances both the light sources and the headlights themselves break down due to the precipitation of smoke and fumes.

Legal and illegal alternatives

The wishes of the car DIYers' scene were quickly recognized by the industry. Since xenon light has become more common in road traffic, it is increasingly possible to buy halogen lamps that promise bluer light. This is achieved by simply colouring the glass body blue. Looking at **Figure 4**, not only does it help much, but reduces the luminous efficacy. The light spectrum of halogen lamps is very red. The blue coating only attenuates longer-wavelength light components but fails to offer additional blue light. The result actually looks a bit bluer, yet is not brighter, but rather darker. If you buy something like this, your physics may be a bit rusty.

And before I forget: There are also 100-W versions of common halogen bulbs. However, since they are far above the permissible power (between 55 W and 65 W), this is not only illegal, but also risky. The plastic headlight casings are not built for such power and can be damaged by the higher heat dissipation.

It's also feasible to replace the complete halogen headlamps with genuine xenon or LED devices. Unfortunately, these are hard to find on the accessory market, where one gets inexpensive imitation spare parts from third manufacturers. But the original parts are usually expensive. Beside the price, there is also another hurdle: the legislator prescribes the presence of a headlight cleaning system for luminous fluxes greater than 2,000 lm. And that's not all: automatic headlamp levelling is also mandatory. Retrofitting both is not only very time consuming, but also pointless economically. There is one exception: the VW Group has recently installed xenon lamps with just 25 W and a luminous flux of no more than 2,000 lm, which saves on the headlamp cleaning system. However, headlamp levelling is still integrated. If you're interested: Osram builds these lamps under the name D8.

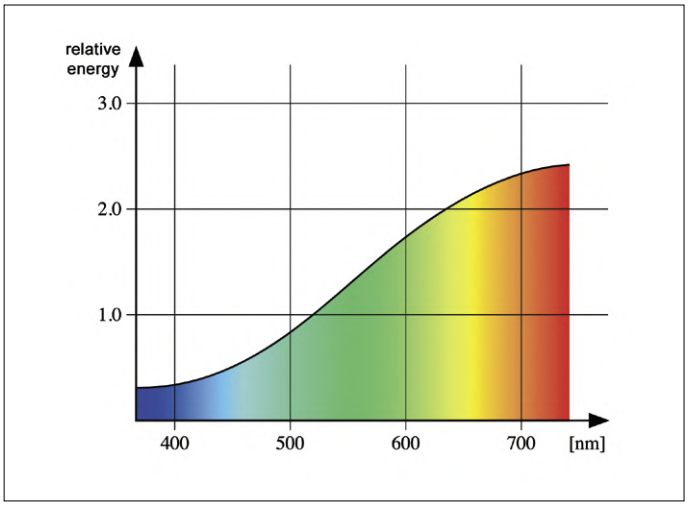


Figure 4: Shape of the light spectrum of halogen lamps. There is not much blue. (image: Jean-Jacques MILAN, GNU CCL 1.2, CCASA 3.0 [5]).

If you want to stay compliant but do want brighter lights on your motor, the only option is to buy special halogen lamps, which are sold by OEMs for absolutely little, but relatively much money. Philips also offers the types "X-tremeVison" with 130% and "RacingVision" with 150% of the luminous efficacy of normal halogen lamps for just under €20 a pair in addition to absurdly blue lacquered lamps. Supposedly they have a slightly increased colour temperature of 3500 K, which is achieved by pale blue lacquer, and interestingly both have a luminous flux of 1500 lm. But brochure information [2][3] should not be taken quite literally, because firstly these lamps are advertised with 130% and 150% "more" light, but this is only 30% and 50% "more" although the weaker lamp is listed as 1550 lm and the stronger one, as 1500 lm. The fact that the former lasts a maximum of 350 hours and the latter only 200 hours (in contrast to the 1000 hours of a standard lamp) is credible and not paradoxical.

Osram, the other major lamp manufacturer, is also unwavering and offers 'Night Breaker' halogen lamps [4] (**Figure 5**) for just over €20, which supposedly are "up to 150 % brighter than the statutory minimum requirements". Whatever that refers to: the luminous flux here is an astonishingly moderate 1350 lm and the lamp should last for up to 250 hours. I bought this one.

Physics of lamp tuning

Having read the reviews on Amazon & Co. you will see a lot of frustration among users of these high-performance halogen lamps. The bulk of the disappointment is due to "durability". Depending on the on-board voltage, the more expensive halogen lamps are gone in no time. No wonder, because the manufacturers build them for a voltage of about 13.2 V at the terminals. The alternators in most cars are matched to the final charging voltage of lead batteries and deliver about 13.8 to 14.2 V. Depending on the level of this voltage and the voltage drop on the wiring and relay contacts, the lamps may operate well below or above 13.2 V. The diagram in **Figure 6**



Figure 5: OSRAM's Night Breakers promise 150% "more light".

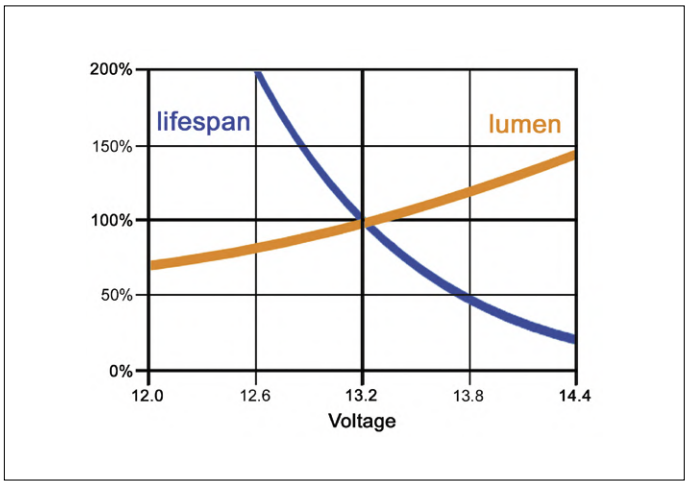


Figure 6: The effect of the operating voltage on the service life and luminous efficacy of a halogen lamp.

clearly shows what this means: a mere 5% up on the voltage halves the lamp's service life! On the other hand, the lamp lights around 20% brighter.

Brighter… for real? No, because this 20% more luminous flux is at the limit of perception, as the impression of brightness unfortunately only follows the luminous flux logarithmically. For twice the perceived brightness, more than four times the luminous flux would be required under these visual conditions. The second disappointment is therefore the benefit of these lamps, because a 50% increase in brightness is unfortunately only perceived as "slightly brighter". The observable shift in colour temperature towards blue is more positive. A 500-K difference with the normal halogen light is perceivable.



Figure 7: DC/DC converter: 9–14 V in and 13.8 V out, weatherproof, and encapsulated in a sturdy aluminium casing.

Also apparent is the fact that these great high-performance halogen lamps were designed in such a way that the filament becomes hotter and therefore brighter and bluer but lasts shorter. In principle, it is simply a lamp operated at a slight overvoltage — nothing more. And there's a surcharge for that, because after all, the lurid marketing has to be paid for. The question remains as to the meaning of the whole thing. In my opinion, such an option is doable. The surcharge keeps itself within limits and all other tweaking and tuning measures are either prohibited or really expensive.

Steady light

As already mentioned, many users of these halogen lamps complain about frequent bulb changes. However, one set of these lamps lasted three years and a good 30,000 miles in my car. But I didn't find them much brighter than normal halogen bulbs, which didn't surprise me because of the psychophysics of human view. Nevertheless I got suspicious, because there was something wrong with my car, wasn't there?

Solution to the riddle: I drive a Prius III, and it doesn't have an alternator at all. The 12-V lead-acid battery required for lighting, power steering and on-board computer is charged very gently from the high-voltage battery by electronics. The lead-acid battery is loaded with under 20 A (measured) and kept at just under 13 V. The battery is then charged with a maximum of 20 A. It also sits in the back of the trunk. Only 11.8 V reaches the front lamps (again, measured). According to **Figure 5**, I can look forward to a relative service life of around 400%. Solved. But I pay for the long service life with 30% less lumens. And since the Prius has a projection headlight for dipped beam, I should not wonder about the really poor front lights.

I could have done it differently, but I would have had to choose an upgrade model with LED lights, which would have meant 0.1 l per 100 km more fuel consumption in addition to the surcharge. When I bought it, I thought to myself, "If it's environmentally friendly, then it's the right thing to do." And now I am stuck with this mess.

Is there anything we can do? How about a powerful DC/DC converter stepping up 12 V to about 13.2 V? As soon as this

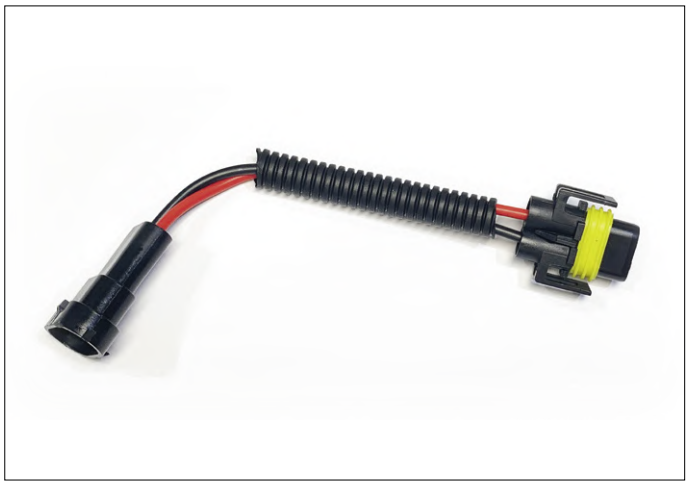


Figure 8: Extension cable for halogen lamps. These were used to equip the converters with matching sockets and plugs for looping in.

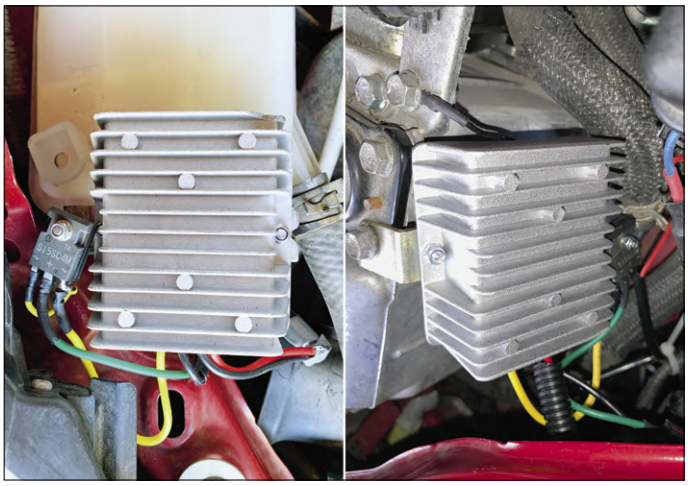


Figure 9: Left and right converter with power Schottky diode connected in series with the output.

occurred to me, I opened eBay. And indeed, there were suitable boost converters with an output capacity of 8 A — a little headroom is not bad. The whole thing for only €12 a piece in a nice aluminium case. Unfortunately, it didn't offer adjustable 13.8 V at the output. But I would find a workaround for that. So, I placed the order and three weeks later I received the thing pictured in **Figure 7**. Tests showed that this converter is suitable, delivering a stable 13.8 V at the output even with variable loads and against input voltages between 9 V and 14 V. I also ordered two H11 extension cables (**Figure 8**) from another source for €2 each, which I wanted to split in the middle and solder to the converters. The operation at 13.8 V would have halved the lamp's life span. So I decided to 'burn' some voltage using a Schottky diode. A 15-A type from an old PC power supply in series with the output theoretically causes a voltage drop of 0.4 V at the given load. My measurements confirmed this: about 13.4 V got applied to the halogen lamp. I could live with that.

So I installed the two modified DC/DC converters with the help of metal brackets near the headlights. As **Figure 9** shows, the space on the left side was a little bit cramped. And if you look at **Figure 10** you can see the slightly bluer light of the outer low beam compared to the high beam. Above it, by the way, is the really yellowish 5-W parking light, quite conventional and not even in halogen. After one month of operation and over 600 night-time miles, I have no complaints. Is it effective? That's purely subjective but at least it works.

By the way, such a mod can also be interesting for non-Prius drivers, because the light is switched on softer and operated without voltage fluctuations. This benefits the service life of the lamps. And since approved light sources and unmodified headlights are used, this improvement is legal in Germany to the best of our knowledge. Otherwise you can't change the exterior lighting of a car. But you can let off steam with the interior lighting of your car: retrofit LED lamps are permitted here. At least in my car, you won't find a single light bulb inside. ⏮

191016-03



Figure 10: Tadaaa! My Prius with DIY-tuned low beam. You can see the slightly less yellowish colour of the low beam, which is now run at 13.4 V (high beam at only 11.8 V).

Web Links

[1] ADAC Test (in German): www.adac.de/infotestrat/technik-und-zubehoer/licht/lichttechniken/

[2] X-treme Vision: www.philips.co.uk/c-p/37166628/x-tremevision-car-headlight-bulb

[3] Racing Vision: www.philips.de/c-p/12972RVS2/racingvision-fahrzeugscheinwerferlampe

[4] Night Breaker: www.osram.com/am/campaigns/nightbreaker.jsp

[5] CCASA 3.0: https://creativecommons.org/licenses/by-sa/3.0/deed.en

REVIEW

Toolcraft Digital Soldering Station

By **Luc Lemmens** (Elektor Labs)



If I had to choose a soldering station, I'd have a strong preference for the A-brands. It's not a question of snobbery: in general, I only want to buy tools that I can enjoy using for many years. Quality and reliability are important factors, but the long-term availability of accessories or spare parts also plays an important role for me.

With all due respect, I must admit that I do not consider Voltcraft — Conrad's house brand — to be one of the top brands. Yet their ST-100D soldering station looks interesting enough to put through its paces and investigate just what you get when you go for mid-range in terms of price.

First blush

You only get one chance to make a first impression, and I can only say that the ST-100D scores well in that respect: a nicely finished, robust metal case for the soldering station and neat standard for the soldering iron. I didn't delve deeply into the history of this set, but on the web I read that there is at least one predecessor with the same type designation, which suggests that this model was worth its salt over the years and deserved a makeover.

I have no idea if the exterior had a (successful) facelift only, or if more got modified, but for a review of the current version that's not very interesting either. It is important though to know that it is apparently not a passing fad and that gives us enough confidence in the availability in the future of a replacement soldering iron and spare tips.

The soldering station

There is not much to be said about the heavy metal casing. It's very firm and doesn't move when you operate the buttons — just the way you want it. A clear LC-display indicates

the iron's set (desired) and actual temperature. A bar graph on the left-hand side of the display gives an indication of the station's power output according to the instructions for use, but it looks more like it indicates the extent to which the actual temperature deviates from the desired value.
To set the temperature, we find a rotary control on the front, which works better than the Up and Down keys typically found on soldering stations nowadays. There are also keys for three preferred temperatures that can be easily adjusted, which is quite handy too.
For sure, there's an On/Off switch on the front, the connection for the soldering iron and a 4-mm socket for equipotential bonding. The station lacks a timer that automatically regulates the temperature if the iron is not used for a longer period. If you want to enjoy the use of soldering irons for a long time and wish to conserve energy, you will have to think about shutting down the soldering station in time.
The only 'hidden' function in this unit is the temperature calibration, which, according to the manual, should be performed when another soldering iron is connected. Not everyone will have a thermometer handy that can measure such high temperatures (up to 450 °C). However, it's essential for this adjustment. The soldering station has been matched to the supplied soldering iron at the factory. Furthermore, the temperature may differ slightly when a different soldering iron is mounted. In that case it's a matter of setting an offset for the display, but of course you also need a suitable thermometer.



Figure 1: The front is clearly laid out; the temperature dial works very well.

The solder stand

To be honest, I find this one a little disappointing. I really like it when a stand can be placed securely on the table and doesn't move when I insert or remove the iron; in that respect, this set doesn't score particularly well. A little more mass would have made a better impression on me.

A sponge for wet cleaning and a dot of 'brass wool' with matching cover plate for dry removal of solder and resin residues are included. However, the solder stand only offers space for either one at a time. I prefer to use the wet sponge for cleaning the tip during soldering jobs; I only use the dry metal for more thorough cleaning, but changing it is a bit more cumbersome with this stand.

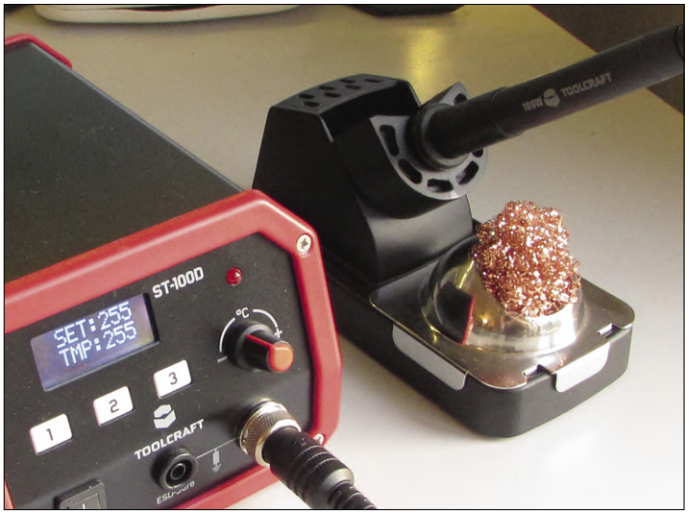


Figure 2: The stand could have been slightly heavier; also, it also only offers space for either a wet or a dry 'sponge'.

The solder tips

The manual doesn't mention any suitable tips for the soldering iron. But if you look up the ST-100D in Conrad Electronics' web shop, at the bottom of the page under 'Recommended accessories' you will find a series of pens with diameters ranging from 1.2 mm to 3.2 mm, chisel and pencil models as well as chamfered tip models. Interestingly, one of the pictures shows the logo and name of the company Hakko; their 900M-T series of soldering tips (a type designation that's also visible in the pictures) is much more extensive and available "all over the web" at great prices. As far as tips are concerned, this soldering iron is an excellent choice, although the cheaper ones — most likely copies of the original Hakko tips — are guaranteed "dubious-quality".
Hakko only supplies this series of tips as spare parts, but no longer uses them for more recent soldering irons. From inquiries, Hakko now uses the improved T18 series, which can certainly be used as a replacement for the 900M series.





Figure 3: A fine (1.2-mm) pencil–shaped tip is supplied as standard.

Our Toolcraft ST-100D came with a pencil shaped tip with a 1.2-mm diameter tip, which is very nice for all finer (SMD) work and rework. I would have preferred a slightly larger chisel tip of about 2.4 mm, which is more universally applicable and therefore also suitable for heavier soldering jobs. Fortunately, the cost of the pens is not so bad, and it never hurts to have different sizes or models available anyway, so I would immediately order a few.

So how does it solder?

In the end, the most important question when talking about tools is: do you enjoy working with them? The cable between iron and station may be a bit thick, but long enough and certainly not annoyingly stiff or heavy during soldering. Even at the highest temperature the handle doesn't run annoyingly hot — a pleasant observation. But how a soldering iron works for you is and remains a matter of taste: I found this Toolcraft station good and pleasant to handle myself, while a colleague found the tip too long to be able to solder precisely (meaning the hand is too far from the solder tip).

The soldering station keeps the iron at the right temperature. The supplied tip is not really suitable for larger solder joints, but with a little effort (keeping the iron flat) you can easily heat larger surfaces to the right temperature, even with a tip that thin.

All in all, for the price of the Toolcraft ST-100D I think you get a good soldering station, even though I see — as mentioned before — a few points where this set could be improved. Well... if the perfect soldering station would exist, there wouldn't be that many brands and models in the market. ⏮

190385-03

@ WWW.ELEKTOR.COM

→ Toolcraft ST-100D Digital Soldering Station (100 W)
www.elektor.com/18993



Arduino Pro IDE

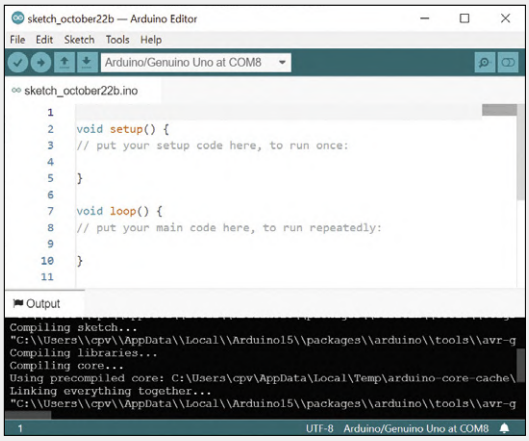
First Impressions

By **Clemens Valens**

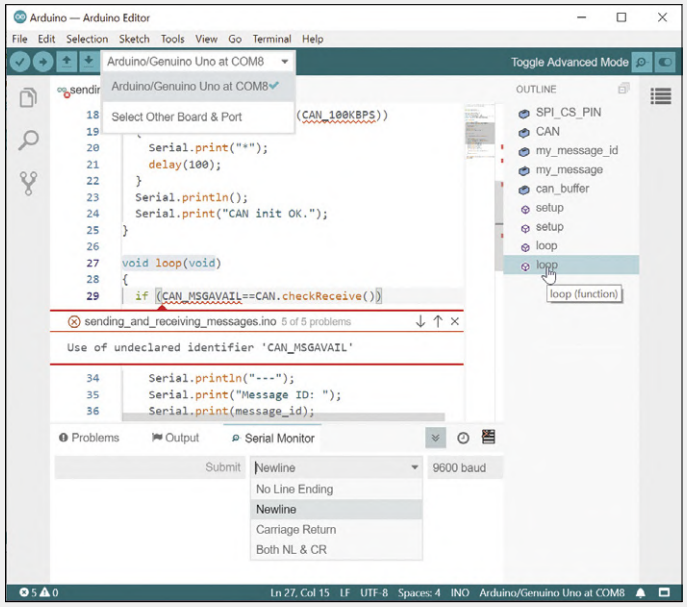
Released last October at the 2019 Maker Faire Rome, the Arduino Pro IDE is based on Eclipse Theia. The classic Arduino command line interface (CLI) still provides all the main Arduino features, but now runs in deamon mode.
Available for Windows, Mac OS X and Linux64 at the time of writing, the Arduino Pro IDE was at version 0.0.1-alpha-preview which suggests that most users had better wait a few releases before adopting the new IDE. Those who prefer to build the IDE themselves from the source code will have to wait too.

Main features

- Dual Mode: classic and advanced
- New Board Manager
- New Library Manager
- Board list
- Basic Auto Completion
- Git integration
- Nicer Serial Monitor



In Classic mode the Arduino Pro IDE indeed looks a lot like the classic IDE.



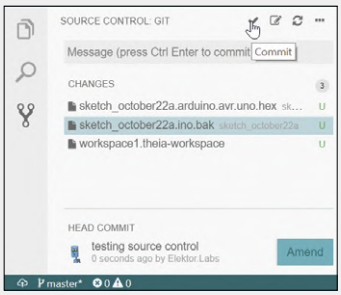
The new IDE showing off some of its features like board & port selection, sketch outline, Serial Monitor configuration and error messaging.



In case of a compilation error, the editor proposes solutions.



Autocompletion needs some getting used to before it will help you save time and avoid typing mistakes.



Source control in the shape of GIT is included, but how the heck do you use it?

191146-01

Web Link

[1] Video: First impressions of the Arduino Pro IDE: https://youtu.be/AfvCBJ8By-0

Two Thermal Imaging Cameras Compared

Seek ShotPro and Seek Compact

By **Harry Baggen**
(Elektor Netherlands)



Infrared cameras are very convenient for tracking insulation leaks in rooms, blockages in drains and (aspiring) hotspots in electrical installations. But such a camera can also offer the electronics engineer a solution to tracking down heat problems in circuits. In this article we compare two IR cameras from the American manufacturer Seek, which have sensors with a much higher resolution than those in other devices in the same price category.

In this article we compare two IR cameras from the American manufacturer Seek, which have sensors with a much higher resolution than those in other devices in the same price category. We probably don't need to explain to most electronics engineers what such a thermal imaging camera does. In contrast to a normal camera, this one is not sensitive to the visible light spectrum, but is only sensitive to infrared radiation. And this radiation is a measure of the temperature of the objects that are observed. These temperatures are usually represented on the screen using different colours.

Large and small

Here we look at two products from Seek that are in very different price segments. On one side we have the semi-professional Seek ShotPro with a price tag of 850 euros and on the other side there is the Seek Compact, a plug-in module for smartphones for some 300 euros, which is available as an Android and an iPhone version. Nevertheless these two have one thing in common: They provide a very detailed image that shows an accurate distribution of the heat. To compare: an IR camera in the price category up to 1000 euros has a sensor with at most 160 x 120 pixels, while the Seek ShotPro has 320 x 240 pixels. Even the relatively cheap Seek Compact already has a sensor with 206 x 156 pixels.

Seek ShotPro

The ShotPro is an standalone IR camera in the format of a smartphone. The entire housing is covered with a tough rubber covering and the thing looks like it could handle the odd bump. On the front is the lens with a lens cover, on the other side is a 3.5" touch screen. The camera can make a wireless connection to a PC using WiFi.
When touching the image on the screen, a bar appears at the top and bottom for the various settings. The top bar leads to the preference settings including the choice of temperature unit, setting the clock and the emissivity. The bottom bar gives access to the image settings. Here different colour patterns for the display of the image are available, which spot-tem-



Figure 1: Seek ShotPro (image: Seek Thermal).



Figure 2: Seek Compact (image: Seek Thermal).

peratures are shown in the image and there is access to the stored screen shots.
A very important little icon is the 'eye', that offers the option of combining the thermal image with an image from the normal camera that's also built in. This results in a mixed image where it is much easier to see what you are measuring. You can adjust the amount that each image is displayed. There is also the option of shifting the IR image with respect to the visible one, so that they align with each other as accurately as possible. This feature can also be found on professional IR cameras.

Seek Compact

This is the smallest camera in the Seek family and it isn't quite 4.5 x 2 x 2 cm in size; nevertheless it contains a sensor with 206 x 156 pixels. The Seek Compact is implemented as a plug-in module for a smartphone and exists in two versions: with a micro-USB connector (Android phones) and with a Lightning connector (iPhones/iPads). The little camera is supplied with a sturdy storage case so that you can carry it around safely.
After installing the accompanying app, the Seek Compact is ready for use. The features of the app look very much like those of the ShotPro. There is a range of colour palettes to choose from, there are various options for displaying spot temperatures in the image, of course there is also a screenshot function and there is a settings menu. Compared to the ShotPro there is here no option for setting the emissivity, but the average user will not likely miss that much.
Just as with the ShotPro, it is here also possible to combine the thermal image with the image from the smartphone camera. The result is however somewhat different, one part of the display will show the thermal image and the other part the image from the normal camera. The images are therefore not projected over each other, as is the case with the ShotPro. While there are various settings for adjusting the size and position of the images, I didn't find the result quite as convincing because of the differences in the view angles and positions of the cameras.

Side by side

I tested these two camera side by side and measured a large number of objects.
In practice, the Seek ShotPro turns out to be a very convenient and robust device, with a very good image quality. The resolution is excellent and especially with the overlaid visible/IR image you will have a great deal of information. The battery life, according to the manufacturer, is four hours. That is generous. Of course, there is quite a substantial price tag attached to this device, but considering the quality and performance it is certainly worth its salt.
With the Seek Compact the contours in the image are a not as easy to see. I have experimented quite a bit with the different palettes to get things to show up somewhat clearer. While the resolution is visibly less than that of the ShotPro, this disadvantage can be mitigated by going closer. Also the solution of the double image is not as elegant, but that is just simply the disadvantage of a plug-in module that does not have two cameras on board itself. But with all this we should not forget that the price is considerably different at 300 euros.
In any case, both solutions are well suited for measuring the temperatures in electronic circuits, for example, but you can do much more than that. Walk around your lab or home and point the camera at all kinds of devices, windows and walls. You will be surprised with the things that you will "see". ⏮

(190321-02)



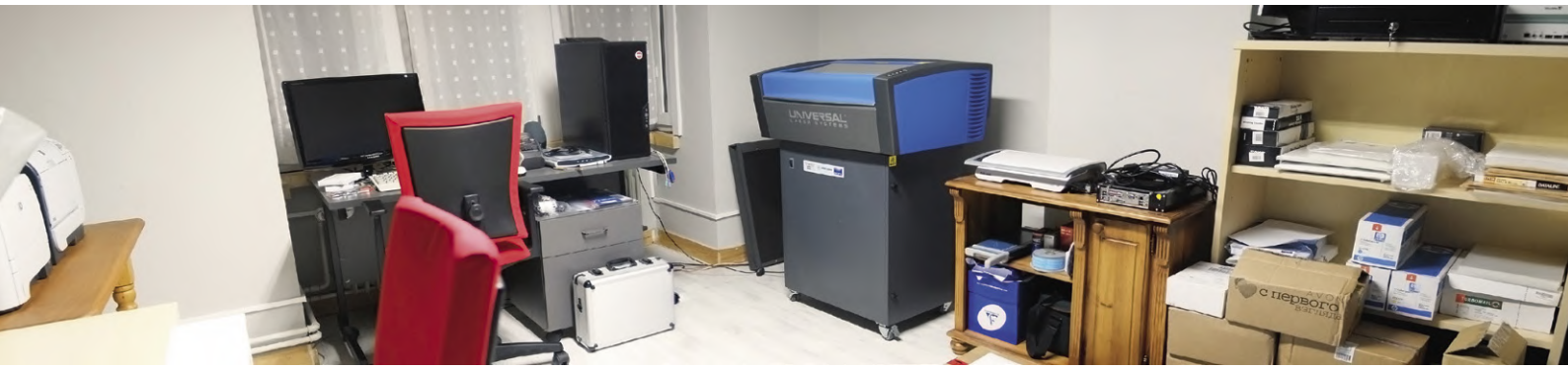
@ WWW.ELEKTOR.COM

→ Seek Shot Pro Thermal Imaging Camera (320x240)
www.elektor.com/18900

→ Seek Compact Thermal Imaging Camera (206x156)
www.elektor.com/18901

Steeped in Electronics

Outfitting the e-lab and e-workshop



By **Ilse Joostens** (Belgium)

In my spare time, I like to browse online electronics forums to see what's going on, not just among electronics hobbyists but also professionals. Setting up a lab or workspace is frequently discussed there, with or without visual material thrown in. Among hobbyists in particular, two archetypes stand out: the 'braggart' and the 'low-budget man'. While the former typically posts photos of a workspace filled to the ceiling with all kinds of obscure measuring equipment, the low-budget man (or woman) prefers to buy his (her) stuff in the local hardware store, the cheaper supermarket and, recently and increasingly, from China.

If you are going to work professionally and want to develop and market your own products, you won't be able to make it with a 'hot poker' from the 'tenner' price range. Let's try to make a few things clear.

Measuring equipment and other electronics tools

Right now, I don't use much more than three lab power supplies, two multimeters, an elderly 'scope, two soldering stations, a hot air station, a universal programmer, a stereo microscope and a thermal camera I actually bought for other purposes. Personally, I prefer quality over quantity, and when it comes to equipment and tools for intensive use, I stick to A-brands — but only what I really need. Ease of use, reliability and longevity are paramount. Sure, the price plays a role, but many other, occasionally less obvious, aspects are also important.

It is not uncommon for a piece of measuring instrument to appear on YouTube or in an assembly manual for a kit. In these cases, showing a decent professional multimeter really makes a better impression than a low-budget one from the local DIY store. Perception is everything, and you want to make a professional and reliable impression on your customers.

The tax burden in my country is among the highest in the Western world and sometimes an investment is an attractive way to reduce the said taxes. At the end of last year, by recommendation of my accountant, I bought as well as renewed some equipment. In that specific case, the tax deduction/quality ratio was particularly important, and I purchased two very solid but quite expensive soldering stations, among others.

Any equipment that's unlikely to see much use, such as the universal programmer, or that's less critical, I also buy second-hand every now and then, although this always makes me anxious.

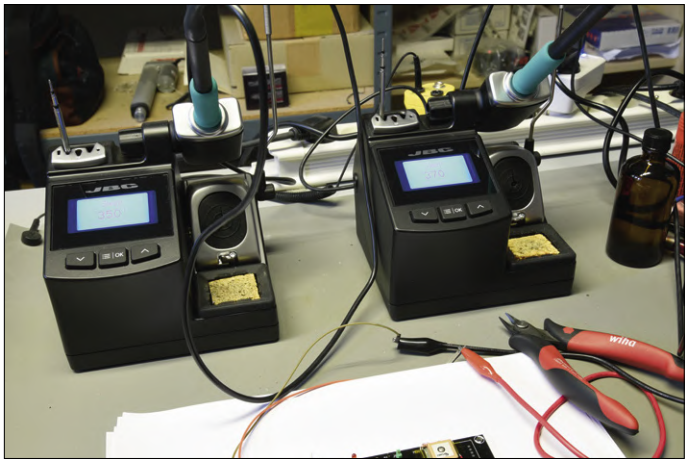
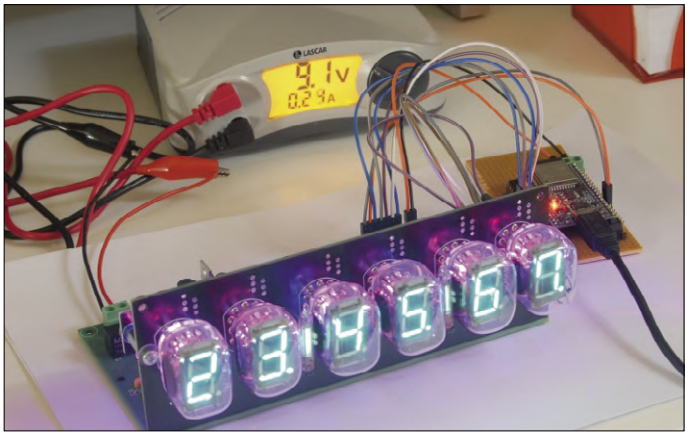


Figure 1: Up-market soldering stations — courtesy of the Belgian taxman.

Production machines

If you want to market your own products and object to outsourcing everything, there is a good chance that you will have to purchase one or more production machines. In my case it was a laser cutting machine and a reflow oven.

Above all, do not give in to the tendency to invest in cheaper machines that are actually designed for hobby use. Nothing is more frustrating than hundreds of back orders and dissatisfied customers al owing to a machine having crashed suddenly in the middle of a production process. In the long run, you're assured of being more profitable with a cheaper machine, not to mention recurring annoyances such as having to replace a laser tube 30 times. With a professional machine, you can also be sure that replacement parts will remain available in the long term; you can also enter into a service contract if you wish.



Other equipment

In addition to the classics, when you want to market your own products, you probably need a lot of other equipment and materials that's unlikely to be of interest to the average hobbyist. You can think of a solid SLR for product photos, lighting, photo backgrounds, heat sealers, a colour laser printer, a paper cutting machine, racks for finished products and parts, power tools, and so on. These things quickly get very expensive, so you better take that into account right from the outset.

Software

Of course, you also need a lot of software and multiple computers. As far as these computers are concerned, I won't recommend any to you, you'd better find out for yourself — and ElektorLabs is not a computer magazine. And no, I don't intend to repair your computer! Not now and not ever!

In terms of software, in addition to a classic Office package, I currently use a PCB design package, two different CAD programmes (2D), photo editing software, DTP software and software for a number of very specific applications. An acquaintance of mine used to say: "You don't buy software, you copy it." Unfortunately, unless you can use free open source software, for commercial applications you will usually end up with the more expensive licenses. Unfortunately, again, in recent years, the licensing model of many software packages has evolved from a one-time purchase to an annual or monthly subscription. To me, it is not an insurmountable problem to pay a reasonable one-time fee for a license, but with the current subscription rates, it is a bigger problem. Paying between 600 and 700 euros a year to be able to use a package — that's close to extortion. For all my software together, that quickly amounts to a few thousand euros a year. And what happens when you cancel your subscription — can you still open your files? That's a question... and with all the taxes I already have to pay here in Belgium, things are gradually getting too bad.

For now, I will stick with my old familiar Windows 7 (for the Microsoft haters among you: yes, I also use Linux). By the way, all my vintage, lifetime licensed' software suites still work fine. Okay, it may not be as slick, but it does what it's supposed to do. All those new-fangled beeps and bells in the latest versions tend to make a program even slower. I also hardly get any emails and appeals to upgrade to a subscription anyway. I guess they've given up by now, hurrah! ⏮

(191152-04)

@ WWW.ELEKTOR.COM
→ Toolcraft ST-100D Digital Soldering Station (100 W)
www.elektor.com/18993

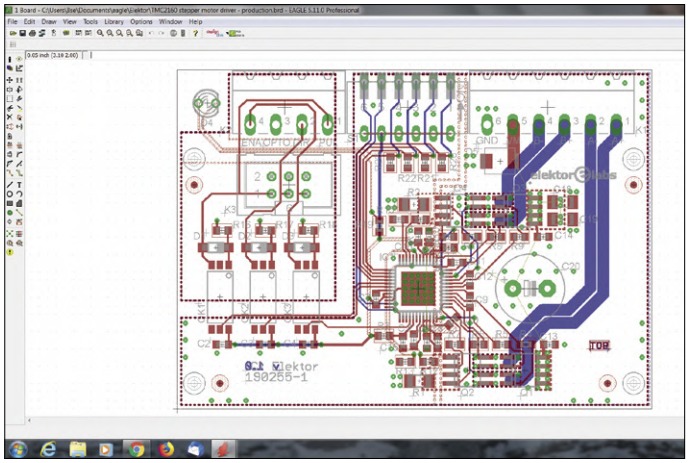


Figure 2: Version 5 of Eagle is not the latest but is good enough for me.



Figure 3: Two printers and a stereo microscope.

Monsanto MAN1 LED Display

Peculiar Parts, the series

By **Neil Gruending** (Canada)

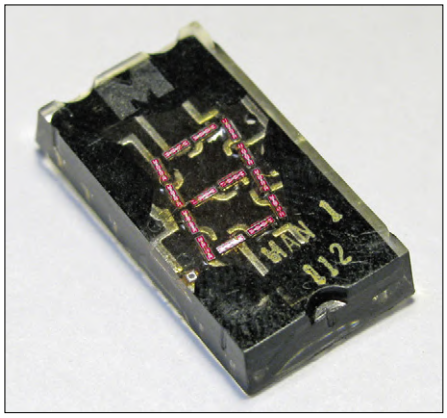


Figure 1: Monsanto MAN1 construction [1].

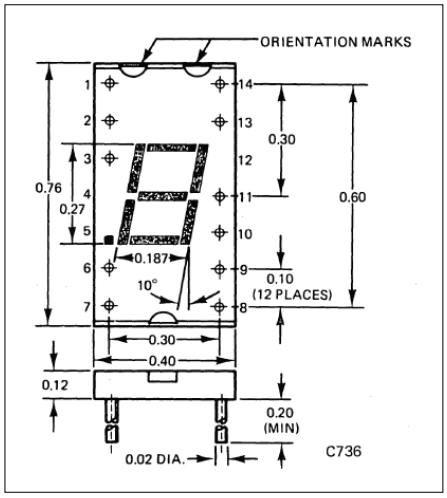


Figure 2: Monsanto MAN1 dimensions [2].

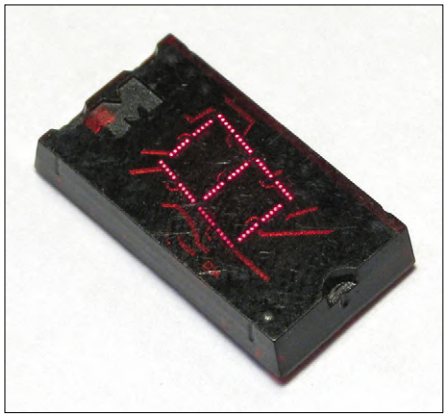


Figure 3: The Monsanto MAN1A [1].

The idea of representing alphanumeric characters with seven segments started in the early 1900s in order to transmit characters via telegraph and then print them onto paper tape. Shortly after 7-segment displays were being added to large displays where incandescent light bulbs were used to illuminate the individual segments. But the displays weren't very popular until miniaturized LED versions became available in the early 70s. Let's take a look at one of the first examples, the Monsanto MAN1 (Monsanto AlphaNumeric).

LED-based 7-segment displays were an important milestone because they consumed much less power than other available technologies at the time, like Nixie tubes, Minitrons and Panaplex displays. Their relatively low power consumption also made battery-powered devices like watches like the Pulsar Time Computer and calculators like the HP35 possible. Battery life was measured in hours which is pretty poor by modern standards, but the ingenious designers extended it as much as possible by blanking the display whenever possible.

The Monsanto MAN1 used two gallium-arsenide phosphate (GaAsP) dies which contained four LED diodes to illuminate each digit segment for a total of 57 diodes including the decimal point as shown in **Figure 1**. The overall character was 0.27 in (6.8 mm) tall (**Figure 2**) which helped them fit into many portable applications like calculators. However, it was also common to see magnification lenses over the display to make the digits appear larger. The diodes also weren't very bright, so they were usually covered with a red filter to increase the contrast which also gave displays from that era a characteristic red hue. The MAN1 was housed in clear plastic which showed all of the internal circuitry, so a later version called the MAN1A was introduced that used red plastic to mask the internals (**Figure 3**).

Competition was fierce and LED display technologies changed quickly. It wasn't long until HP introduced 5×7 dot matrix LEDs with integrated binary coded decimal decoders (BCD) and by 1977 Litronix, a Monsanto spinoff, had introduced an intelligent LED display with row/column drivers and integrated character ROM. But it's still interesting to see these cool little MAN1 displays in vintage electronics and test equipment. Detailed catalogue information for them is available [2] under the General Instruments name as they acquired Monsanto in 1979 [3] should you want to learn more about them. ⏮

190383-01

Web Links

[1] Monsanto MAN1 construction: www.decadecounter.com/vta/articleview.php?item=463

[2] Monsanto MAN1 dimensions: https://bit.ly/2BuRRgf

[3] Monsanto LED history:www.datamath.org/Display/Monsanto.htm

HOMELAB PROJECT

Thermoelectric Tea Timer

First dealings with energy harvesting



By **Lothar Göde** (Germany)

This project controls the brew time of a tea bag in hot water. The mechanics are controlled electronically, and the electrical energy required comes from contemporary energy-harvesting technology, specifically, from the temperature difference between ambient air and the hot water. Meaning batteries (dry or rechargeable), or even a power adapter, are redundant in this project.

Some time ago, I read an article about energy harvesting. This technology was new to me and I found it very interesting and exciting at the same time. I received an Energy Harvesting kit through an organization that, among other things, provides kits to youngsters to introduce them to elementary electronics as part of their school curriculum. The core of the kit is the LTC3108 integrated circuit from Linear Technology [1] (today Analog Devices). I was fascinated by the fact that a direct voltage of just 20 mV, a level no normal transistor or MOSFET can work off, produces a much higher voltage suitable for powering a microcontroller, for example. The LTC3108 is designed to draw its input voltage from the temperature difference of a thermoelectric generator (TEG).

Thermoelectric generator

A thermoelectric generator can generate electrical energy from a heat difference or conversely generate a heat difference by supplying electrical energy. A TEG consists of two isotropic, homogeneous semiconductor materials with different (p-,n-) doping (**Figure 1**). The most popular application of the thermoelectric or Peltier effect are inexpensive coolers plugged into the cigarette lighter in the car and draining the car battery very quickly (without the beer really getting cold).

But it also works the other way around: the reverse of the Peltier effect is the Seebeck effect. An existing heat difference gets converted into electrical energy. The voltage generated depends on the properties of the thermoelectric materials and, of course, the temperature difference. By the way, a thermoelectric generator operates similar to a thermovoltaic generator: the current tapped is selected so that the terminal voltage drops only slightly, but the output is maximum (maximum power point – MPP).

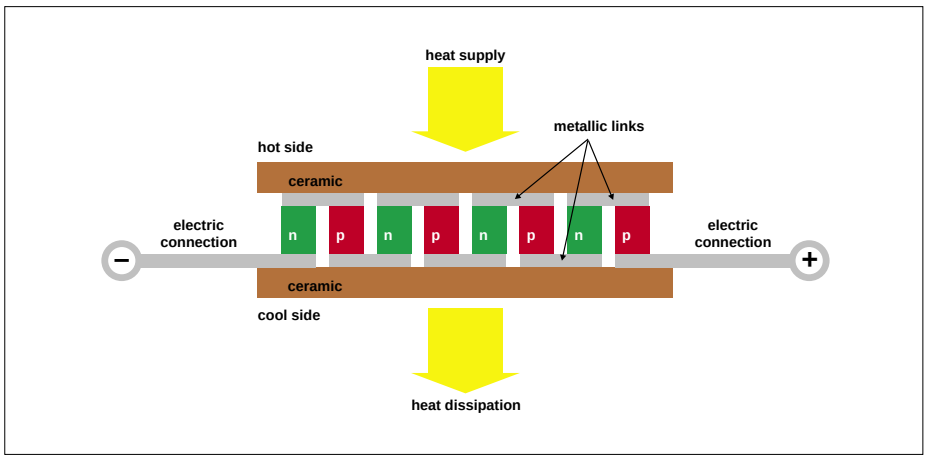


Figure 1: Basic structure of a thermoelectric generator.

Hot tea, cold tea?

While experimenting with the interesting educational package, I often drank tea on the side. But then, in your enthusiasm you usually forget the teacup and your tea gets cold with a bitter taste. To solve the problem, I asked myself: 'Is it possible to capture or convert so much energy from the temperature difference of a tea glass filled with boiling water and

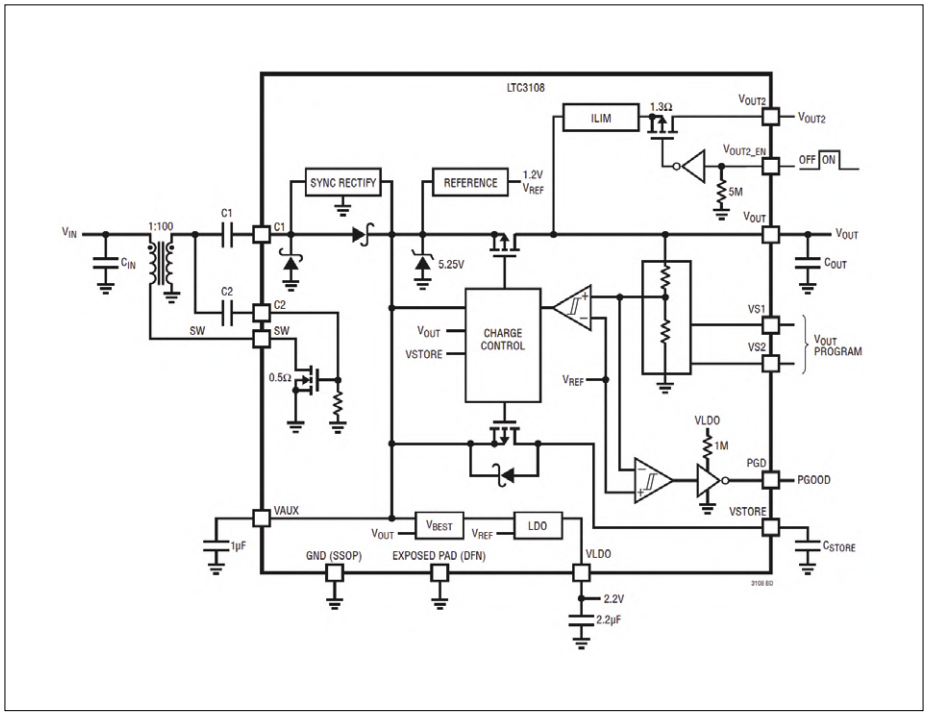


Figure 2: The LTC3108 can harvest energy from “virtually nowhere” to supply a small circuit (source: Analog Devices).

its environment, for an energy-saving electronic or electromechanical device to lift the tea bag out of the tea glass at the end of the intended brewing time?’ To answer the question I first had to carry out some experiments with the TEG. For the effective heat dissipation from the cold plate of the TEG, an aluminium heatsink from a CPU cooler from the hobby stash was used. However, you have to try different tea glasses and heat conducting materials to achieve optimal heat transfer. By solving these problems, in five minutes of brewing time you can harvest enough energy from a tea glass containing boiling water for the LTC3108 to reliably charge a 2200-µF electrolytic capacitor (C_{STORE} in **Figure 2**)… and for the control electronics to do its job!

Conveyor mechanism

What can be done with the harvested energy? Of course, ‘power-consuming’ functions are out of reach, and all parts of a tea timer must use energy sparingly.

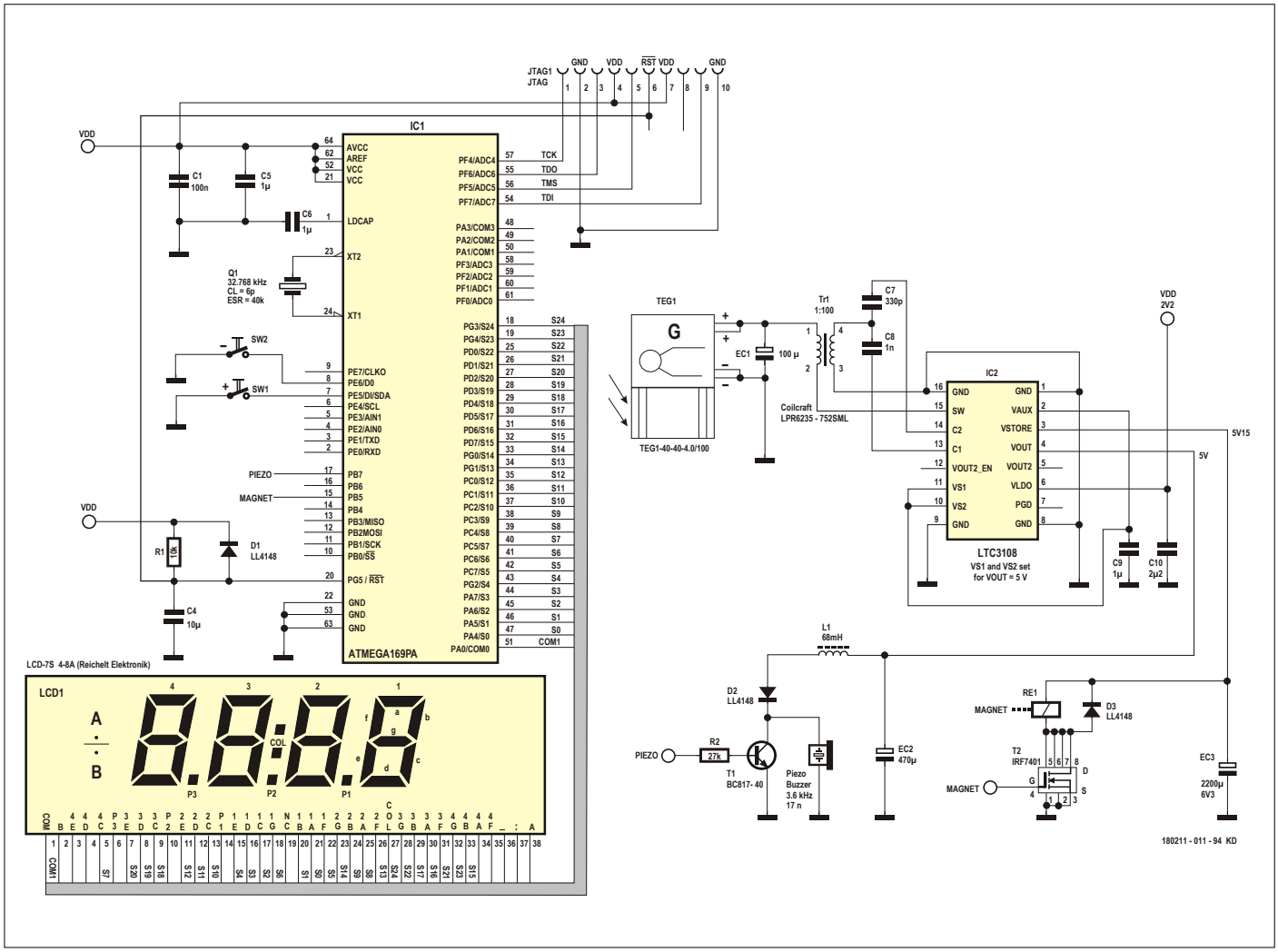


Figure 3: Circuit diagram of the Thermoelectric Tea Timer.

First, the mechanics: a tea glass is placed on the TEG, the tea bag (or tea ball) is lowered manually into the cup and the tea bag label is connected to the 'lift mechanism' wire through a mini clamp. With a traditional circuit, you could now simply equip the lift mechanism with a motor that pulls the bag out of the brew. But that doesn't make sense here. Instead, as shown in **Figure 4**, the wire is both guided and equipped with a small counterweight (slightly heavier than a tea bag). In this way we harvest further energy, which is gravity. Since the counterweight is heavier than the tea bag, it must be locked by a bolt blocking the counterweight's path by spring force as soon as a certain position is reached. The bolt is released electromagnetically with a single electric pulse. Consequently, there is no need for much current to be supplied over a longer period of time.
For the bolt lock, I modified a relay and fixed the bolt to the contact set of the relay. There are also special lift magnets which could be used here. So that the tea does not splash when lifting the bag, I had to equip the counterweight with a rubber damper.

The microcontroller as timer and driver

Because we use energy so sparingly for the mechanism, we can allow ourselves some luxury at the control centre of the tea timer: a microcontroller type ATmega169PA from Atmel [2]. The ATmega169PA has a built-in LCD controller, which allows the convenient connection of a display to indicate the brewing time. It is even actually more energy efficient to use a hardware function of a controller than to emulate this function via software.
Boiling water is now poured into the tea glass to generate electrical energy. As soon as the Energy Harvesting circuit provides the supply voltage for the microcontroller, the brewing time flashes on the display. It takes about 30 seconds for the display to appear. Most teas have to brew for 5 minutes, so the default value is 4 minutes and 30 seconds.
Two buttons are connected to the microcontroller allowing the brew time to be set. If none of the buttons is pressed for about 12 seconds, the display stops flashing and the set brewing time is counted down per second.
The acoustic signal of a piezo beeper sounds a few seconds before the brew time expires. This should be an indication that the tea is ready. When the brew time has finally expired, the microcontroller sends a short signal to the relay coil, which releases the locking bolt so that the counterweight pulls the tea bag out of the tea glass in a damped movement. The acoustic signal generated by the piezo signal transmitter must be slightly delayed at the end of the brewing time in order to compensate for the energy lost in the remaining time, so that sufficient electrical energy is available for actuating the relay coil. If the acoustic signal is ignored, the tea timer does not halt; further acoustic signals follow every minute. This alarm can be switched off by pressing a button. That should not be necessary though, because if you take the hot cup from the TEG, the power supply disconnects in no time anyway.

Conclusion

Designing a circuit in such a way that it works with the tiny amount of energy obtained through harvesting is a major challenge. This applies not only to the hardware design, but also particularly the energy management when creating the firmware. For example, the firmware had to be optimized in such a way that a minimum current is required when setting the brew time by pressing the buttons, so that the supply voltage for the microcontroller does not fail. However, that's exactly the purpose of such experiments! The thermoelectric tea timer can also be extended with a radio module (Bluetooth Low Energy; BLE) to send a message to a smartphone when your 'cuppa' is ready. A power-saving stepper motor could also be used to lift the tea bag out of the tea glass.
Even if it does not look much in the photo showing the prototype (**Figure 4**), the thermoelectric tea timer represents a distinct high-tech device through the use of modern energy harvesting technology [3]. That's in stark contrast with many commercial products performing a similar task, but using a downright old-fashioned approach. Annoying, mandatory tasks such as exchanging or charging batteries no longer exist. Since no batteries are used, no chemical substances can spill. The 'T-T-T' is maintenance-free, except for occasional cleaning of course. No dangerous voltages are generated, obviating the need for protective measures. The T-T-T is unique! ⏮



Figure 4: Still in the experimental stage – the working prototype of the tea timer.

180211-03

Publishers' Note: This project was previously printed in *Reichelt Elektronik Magazin*.

Web Links

[1] LTC3108: www.analog.com/media/en/technical-documentation/data-sheets/LTC3108.pdf
[2] ATmega169PA microcontroller: www.microchip.com/wwwproducts/en/ATmega169PA
[3] Tea Timer video: www.youtube.com/watch?v=z-ias4IwbSQ

@ WWW.ELEKTOR.COM
→ Peltier Lamp
(DIY kit no. 160441-71)
www.elektor.com/peltier-lamp-1

eLABS PROJECT

Single-phase 1-kilowatt AC Motor Drive

Three-mode design: integral-cycle switching, phase-cut trailing, or phase-cut leading edge.

By **Elektor Labs** & **Elektor Labs India**

This low-cost and easy to build drive is designed to control AC motors typically found in household appliances and other equipment like electric tools. The drive presented here does a good job using an MCU running software to match different AC load types like inductive and capacitive. The proposed hardware is capable of driving loads of up to 1 kW. Thanks to the hardware being designed with adaptability in mind, the motor drive's interface can be tailored and enhanced to suit many personal requirements.

Many electrical appliances used at home require AC power for the operation of their motor(s), i.e. 115 VAC or 230 VAC, 50 or 60 Hz from the power outlet and depending on your location. Usually, AC line power reaches appliances through electronic power switches. For smooth operation of the load(s) and the ability to control motor speed it's necessary to govern the AC power applied to the load(s). This is usually achieved by controlling the operation of several electronic power switches.

Single-phase AC motors are still a primary solution for air displacement, pumping and compressor applications. Their low cost and availability make them ideal for low-performance systems. DC brushless platforms can also be used for these applications, but their high cost and complexity creates limitations, or 'challenges' in modern speak.

Design considerations

The circuit was designed in such a way that its use extends from ordinary AC motors to many other AC loads where power control is required or beneficial. To achieve this, an easy user interface needs to be provided to select the different power control methods for, indeed, different types of AC load you may be faced with. Consequently, the drive should easily 'handle' loads like a bunch of high-power LEDs, a builder's halogen power lamp, or grandma's mini electric heater. Simple user panels should be present for readout and selection, with the help of an OLED display and a rotary encoder/pushbutton. The 'user panel' should be extendable to suit additional functions.

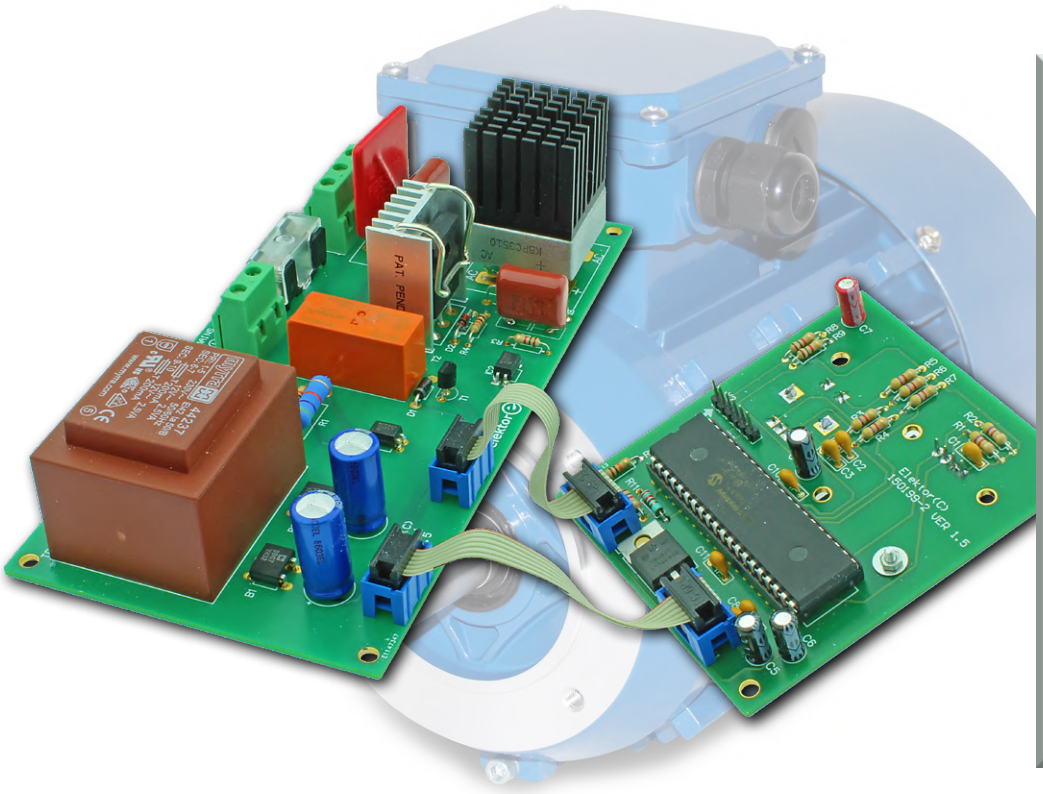
Features

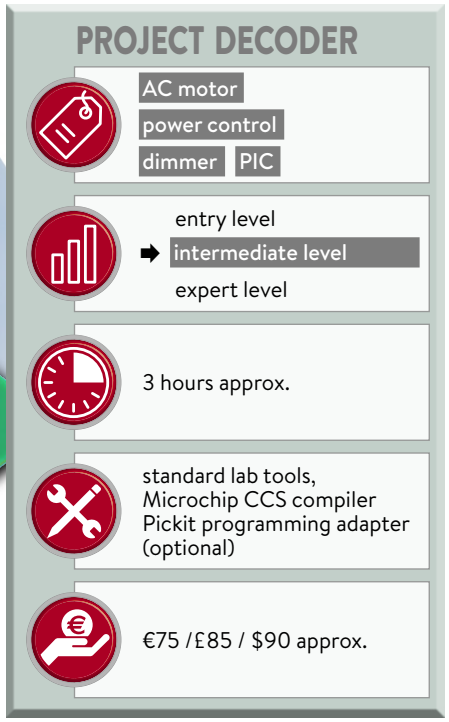
- Single-phase AC output:230 VAC, max. 1 kW
- Modes of operation:
  - Integral-cycle switching
  - Phase-cut Trailing Edge
  - Phase-cut Leading Edge
- 0.96" OLED display panel for readout and user interface
- Zero-SMD construction
- 3-way connector for motor / load
- Relay controlled motor forward / reverse direction
- PIC18F45K22 based drive control
- Internal, isolated MCU based control section
- Forward / Reverse direction control by rotary encoder switch
- Output power / motor speed control by rotary encoder
- Motor / load drive parameters stored in EEPROM
- Soft start initiated from zero on changing mode or motor direction

Methods to control AC power

Let's list and discuss a few established methods of controlling the amount of power applied to an AC load like a motor.

1. Integral-cycle Switching. (a.k.a. Cycle-stealing or Skipping). This method is used for direct conversion of AC to AC known as zero switching or cycle selection. Integral-cycle triggering relates to alternating current switching circuits, particularly to integral-cycle zero-voltage alternating switching circuits. Cycle-skipping is often used to control high-power electrical loads whose response time constant is much longer





than the period of the AC line used to power the load. An example is a kitchen outfit with electric heater elements ("burners") whose thermal time constant may be of the order of one second or greater. Cycle-skipping control has the advantage that the power control switch is turned ON and OFF when the load current and voltage are zero. As a result, electromagnetic interference generated by the switching action is essentially eliminated, and electrical stress on the switches is reduced, which increases reliability.
Most if not all types of electric load accept this method including resistive, inductive loads like motor and capacitive loads. In Cycle-Skipping mode our project can drive loads up to 1 kW.

2. Phase-cut Control. Load switching is controlled with reference to the phase of the AC line signal. At the zero crossing of the AC line waveform, trigger pulses are applied to the gate terminal of the control device. In case of controlling the AC power, the application of these pulses is purposely delayed by increasing the pulse interval. There are two ways to control the phase.
2a. Phase-cut Trailing-edge Triggered (Reverse-phase dimmers). The 'trailing-edge' method is more sophisticated than 'leading-edge' method, and usually involves a MOSFET or IGBT switching device rather than a triac and coil. This benefits the user with smooth, silent dimming / power control, and the absence of any buzzing noise.
A trailing-edge dimmer has a lower minimum load (often just 5 watts) than its leading-edge counterpart, making it a better choice for dimming modestly sized low-powered lighting circuits. Particularly beneficial for incandescent and halogen bulbs, the 'soft start' feature in trailing-edge dimmers, which prevents filament bulbs from dying or exploding of thermal shock when first switched on. Trailing-edge type dimmers are the most suitable choice for the capacitive load of an LED driver to warrant flicker-free operation. This method is suitable for resistive and capacitive loads but not inductive, as it will produce strong back EMF pulses from the load which may damage the MOSFET and any protection devices on board. Since the surge currents are not particularly high, our

AC power control method selection by load type and application

AC power controllers/dimmers used to control…

- … resistive and inductive loads such as incandescent, neon, cold-cathode, and low-voltage (inductive/magnetic) lamps should incorporate leading edge (LE) or forward-phase control (FPC).
- … electronic transformers with their different switching characteristics should employ trailing edge (TE) or reverse-phase control (RPC). It is important to select the right dimmer/power controller for the load at hand. The need for the two dimming waveforms is a result of the drive requirements of different load types.
- … electronic low-voltage loads like SMPSUs with their capacitive input (i.e. voltage lagging current) should cater for a trailing-edge (TE) waveform. TE or RPC dimmers use transistors such as FETs or IGBTs as the power devices instead of the typical triac or SCR devices.

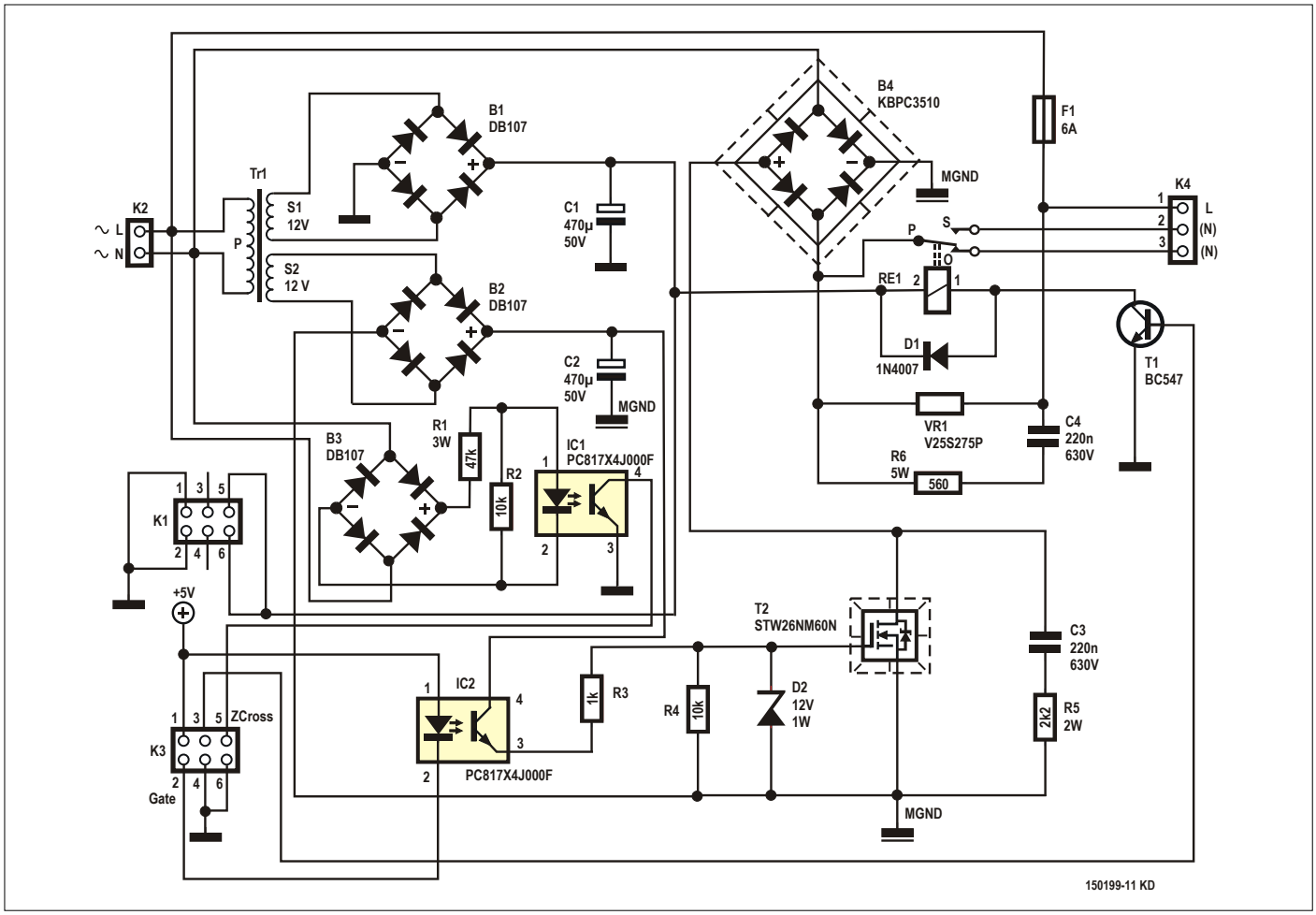


Figure 1: Schematic of the power electronics board. Notice the two ground potentials: GND with its regular symbol and MGND (motor-driver ground) marked differently. To ensure electrical safety, GND and MGND should be kept distinct.

Why use a trailing-edge dimmer?

The cheapest and most common lamp dimmers use triacs 'fired' at a predetermined time each half cycle of the AC line waveform. These are commonly known as leading-edge dimmers, because the AC waveform is turned on partway through the AC waveform. They are also known as 'forward phase' dimmers. If the triac is turned on soon after the zero-crossing voltage of the mains, nearly all of the AC waveform is passed to the load. As the firing time is delayed, less of the AC waveform is passed and the load gets less power. Once a triac is turned on, it remains on until the current drops below the hold current (i.e. the minimum current the device can hold without gate drive) and only then turns off. Because it's bidirectional, positive and negative half-cycles get passed to the load.

A large part of the problems with triac-based dimmers used with electronic loads is caused by the triac turning on very rapidly. This pulses strong current surges into the load and may eventually cause serious damage to capacitors and other parts. For this reason, triac dimmers are not recommended with any dimmable electronic lamp — neither CFL nor LED.

Nearly all household dimmers are only 2-wire types, and this applies almost 100% to triac dimmers. As a result, the dimmer not only stresses the electronics in the lamp, but also loses its reference whenever an electronic load is used. This makes it unsuitable for most electronic loads, even if we ignore the issues with the current surges.

It's important to understand that only leading-edge dimmers can be used with inductive loads, such as iron-cored transformers or fan motors (many triac dimmers can be used as fan speed controllers).

controller can govern loads up to 1 kW in this mode.

2b. Phase-cut Leading-edge Triggered: Leading-edge (LE) or forward-phase control (FPC) are used for resistive and inductive loads like motors, incandescent lamps, neon lights, cold-cathode and low-voltage (inductive/magnetic) lamps. SMPS electronic low-voltage loads tend to have a heavily capacitive input (i.e., voltage lagging current) and require a trailing-edge waveform for 'dimming'. Surge currents are very high in this method. The driver discussed here was tested with loads up to 350 watts and was designed to operate with loads of 1 kW max.

The power electronics board

The schematic of the power electronics board that's one part the project is printed in **Figure 1**. Tr1 is a dual 12-VAC output transformer. The voltage on secondary S1 of Tr1 is rectified by bridge rectifier B1 (DB107) and is used to make the +5-V supply voltage for the MCU cir-

cuit (to be discussed). The other secondary winding feeds bridge rectifier B2 (another DB107) and is used for the power driver section, which is isolated from the MCU section by optocoupler IC2 (a type PC817X4J000F). This will avoid any noise and undue loading of the MCU supply rail. The gate drive signal created by the MCU is connected to IC2 which effectively isolates the MCU from the power driver section.

Bridge rectifier B3 and optocoupler IC1 are used to detect the zero crossing instant of the AC line voltage, and flag it to the MCU for intelligent processing. IC1 also keeps the AC line potential far from the MCU, electrically speaking.

The actual mains power governor consists of power MOSFET T2 (a type STW26NM60N) and power bridge rectifier B4 (a type KBPC3510). The MOSFET is connected across the (DC) +VE and –VE terminals B4, while the AC terminals are in series with AC-N (Neutral), AC-L (Live), and the load. This bright configuration allows cutting or switching the AC signal at any phase angle without the need for two MOSFETs, i.e. one for switching the +VE AC cycle and another for the -VE AC cycle. Applied circuit economy and simplification!

Although a power MOSFET with a low $R_{DS\text{-}on}$ specification was chosen to keep the dissipation low, the worst-case power dissipation is still 3.5 watts or so, requiring a heatsink when the driver is loaded to the max with 1000 watts. Diode bridge B4 also requires fitting with a heatsink as its worst-case power dissipation is about 10 W, again assuming a 1000-W load.

A power MOSFET was chosen instead of the traditional triac to implement the Trailing-edge Phase-cut power control method, which has many advantages over 'Leading-edge' as explained above and in the **'Control method selection by load type and application'** text frame. Resistor R5 and capacitor C3 form the 'snubber' for MOSFET T2 and help to suppress voltage spikes caused by circuit inductances. Resistor R6 and capacitor C4 serve the same purpose. Metal-oxide varistor (MOV) VR1 is used to keep transient-voltage surges at bay. Terminal block K4 is the output connector for the motor or another type of AC load, as discussed.

Relay RE1 is driven by the MCU through transistor T1. It's used to operate the motor in reverse and forward direction by selecting different motor windings by way of connector K4. Fuse F1 is a slow-blow type fuse (a.k.a. delayed action; time lag; or 'T') for protection of the driving circuit against excessive load current.

MCU board

We switch to **Figure 2**. IC2, a programmed PIC18F45K22-E/P microcontroller, is responsible for controlling the AC output power by sending appropriate signals to IC2 on the power driver board. Connector K2 is used to program the micro in ISCP mode. Connector K1 brings the MCU board supply voltage to regulator IC1 which steps it down to +5 V. K3 carries all signals that need to be exchanged between the MCU board and the power driver board.

The microcontroller generates the display signals for .96-inch display OLED1 as well as the interfacing signals for ENC1. The 0.96" OLED display is on I2C. It shows the power output by the drive, while the pushbutton assists in selecting the motor direction and the three different modes of drive operation. The display shows the motor power as a bar, the motor direction, and the mode of operation.

Software time!

The MCU software for the PIC18F45K22 microcontroller in the project was written using Microchip's CCS compiler. It is freely downloadable at [1]. The MCU operates at an internal oscillator frequency of 64 MHz, so the core clock becomes 16 MHz.

At power-on, the project first displays

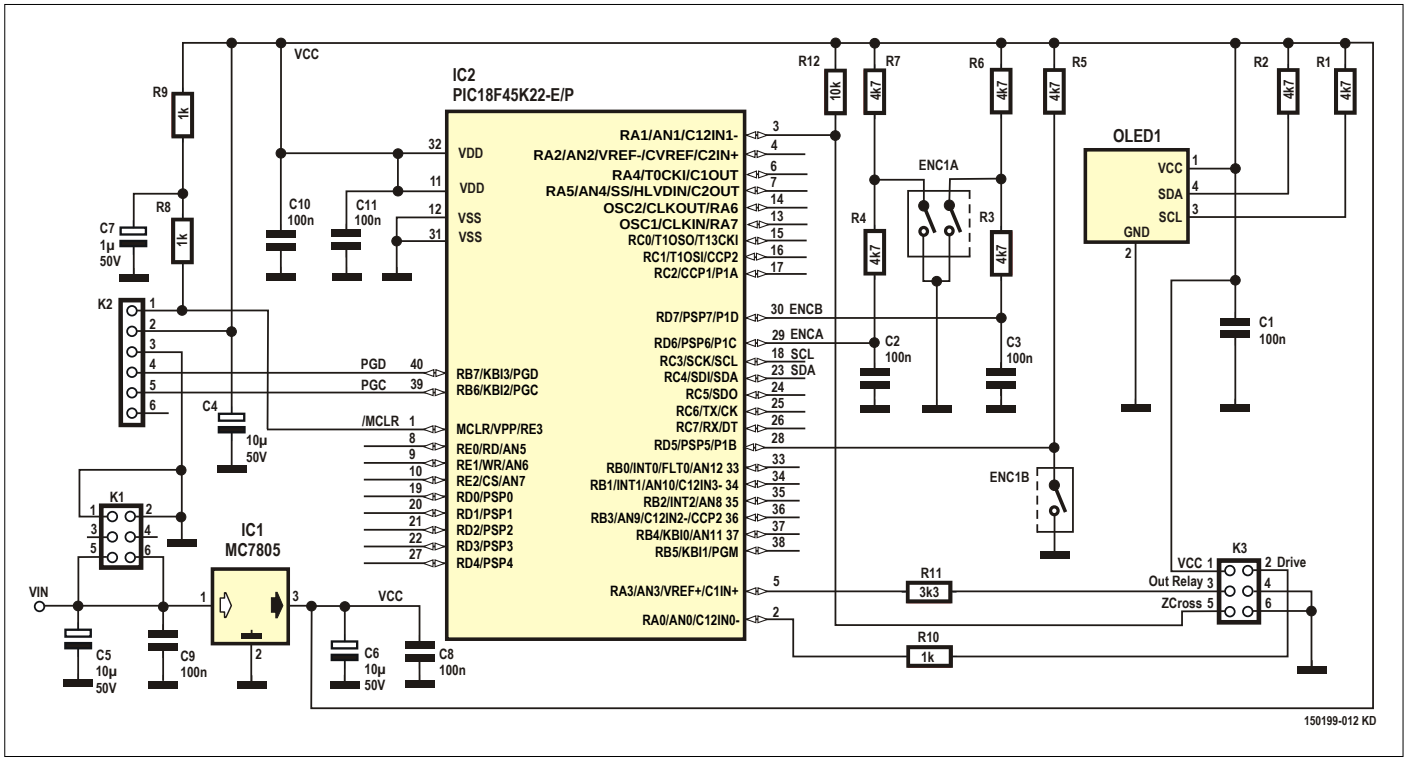


Figure 2: Schematic of the MCU/user interface board. It comprises a PIC microcontroller and a 'human interface' in the form of a rotary encoder and an OLED.

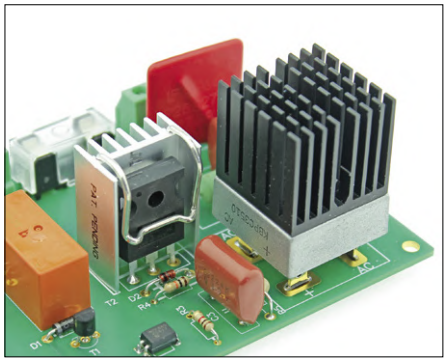


Figure 3: Bridge rectifier B4 gets assisted cooling from a heatsink secured on top by sticky thermal pad. MOSFET T2 is also cooled with a special heatsink clamped on the device. The ceramic isolator is just visible.

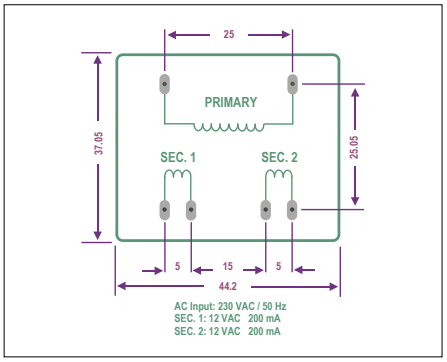


Figure 4: Myrra type 44237 power transformer pin connections and mechanical layout.

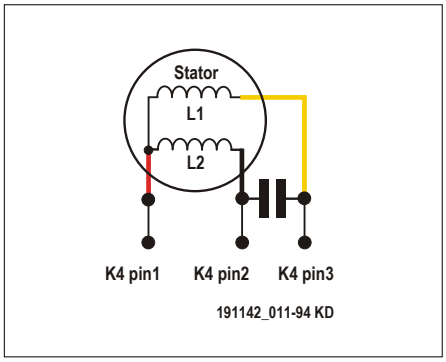


Figure 5: Typical AC motor connections on K4. Be sure you "know your motor", i.e. the individual wires reaching the run winding, the start winding, and the phase-shift capacitor.

a startup message on the OLED display and then reads all the system parameters and settings.
AC sine wave sensing involves interrupt-based flagging of the zero-crossing instant, as the MOSFET has to be switched on or off at exact instants depending on the selected power control mode.
The pulsewidth needed for phase-cut dimming is derived using timer0, which also generates an interrupt at the end of the pulse when in Trailing-edge Phase-cut mode and an interrupt at the start of the pulse when in Leading-edge Phase-cut mode.

Next, depending on the user-adjusted AC power the MCU drives the MOSFET accordingly and provides power to the load, rising slowly from zero to the set power. A 'soft start engine' takes care of smooth startup. The relevant routine also kicks in when the user changes the power level with the motor running. In that case, the unit will not allow sudden change in output power and so helps to prevent damage to the motor or degradation of its performance.
For the user interface one OLED display and a 24-step rotary encoder are provided. This user interface with utterly simple operation allows setting the load power and motor direction, as well as choose the AC load control method. By turning the rotary encoder, you can vary the output power between '0' and '100'. A long press on the encoder shaft selects the control method: Integral, Leading-edge Phase-cut, or Trailing-edge Phase-cut. A short press reverses the motor direction.

Important note: A trailing-edge dimmer must never be used with inductive loads such as iron-cored transformers or motors. Doing so will cause extremely high current and voltage surges and may damage or destroy the dimmer, the load or both.
To ensure safety, the software provides an on-screen message when Trailing-edge mode is switched on. Reversing the motor direction involves the output power decreased to zero first as the motor must be stalled first. Without this measure the motor winding, the drive or both may get damaged. When Trailing-edge mode is selected a warning message appears on the OLED display and remains there 10 seconds. If no key press is detected within this period, the unit rolls back to Leading-edge mode. Conversely, if a key press is detected, the unit will select Trailing-edge mode.

Construction

To avoid issues with electrical isolation, the project is divided into two boards fitted into a single plastic case:

- power electronics board (Elektor Store no. 150199-1)
- MCU board (Elektor Store no. 150199-2).

The boards are interconnected but sufficiently isolated electrically to comply with electrical safety regulations. The motor driver case is closed, made from hard plastic, and has IEC-style AC-in and AC-out connectors only.
Nonetheless, we caution beginners in electronics not to embark on the project without the help or supervision of an experienced engineer familiar with electrical installation and the dangers of working with live grid potentials like 230 VAC.

Power electronics board (150199-1)

Please refer to the pictures of the board, the board's component overlay and the parts list.
This board is spacious and to add to the general festivities **it's single-sided with through-hole parts only**. Start by soldering the lower-profile components like the resistors and diodes. Because they may run hot, R1, R5 and R6 are best mounted 2 millimetres or so above the board surface.
The component outline 'HS1' marks the special clamp-on heatsink for MOSFET T2. A ceramic or thick mica isolator should be used between the heatsink and the metal area on the back of the MOSFET. 'HS2' should be glued on top of B4 using a sticky thermal pad (see **Figure 3**).
Tr1 is crucial to the electrical safety of the project and must never be replaced with something mounted off the board. It's a small, fully encapsulated power transformer type 44237 from myrra (*sic*) with 2 × 12 V, 2 × 2.5 VA, secondaries and a **single 230-VAC 50/60-Hz primary**, meaning those of you in countries or areas with 115-VAC grids should seek an alternative. **Figure 4** shows the myrra transformer's footprint and pin connections.
The AC line (L/N) and motor wire connections are made on sturdy PCB screw terminal blocks with a safe pin spacing of .3 inch (K2 and K4 respectively). In the interest of your wellbeing and continued savouring of Elektor projects, do not cheat & botch by soldering wires to the board. Instead, use the screw terminal blocks we recommend.

The PCB-mount holder for fuse F1 should have a plastic cap to fully cover the fuse and the holder terminals.

MCU board (150199-2)

Here too it's all TH parts even including the microcontroller! Use a classic 40-pin DIL socket for it. Mount the (Elektor) 4-pin OLED display **at the reverse side of the board** using a 4-pin SIL pinheader whose long pins are inserted in the holes from the display side of the PCB. Be sure to use the Elektor 0.96-inch OLED display as other types exist with different pin-outs. The short pins go through the holes in the display board and are soldered also. Also mount the rotary encoder at the reverse side of the board. Secure the OLED display to the MCU board with four small screws in the corners of the board, and use nuts as spacers.

Safety first!

The completed and interconnected boards must be mounted in a fully isolated, transparent case that prevents any part of the circuit being touched except of course the rotary encoder shaft. The AC line IN connector is preferably a chassis mount IEC male receptacle, and the Motor OUT connector is an IEC female receptacle. Both require a clearance to be cut in the case. Never make these connectors yourself or even think of 'improvising' them. New ones cost nothing and afford the safety you should seek. Finally, use approved AC line cords to power the unit and feed the output power to the motor or another AC load.

The motor driver board and the MCU board must be secured electrically isolated inside the plastic case. In our proto-

COMPONENT LIST

Power electronics board, no. 150199-1

Resistors

R1 = 47kΩ 5%, 3W
R2,R4 = 10kΩ 5%, 250mW, 250V
R3 = 1kΩ 5%, 250mW, 250V
R5 = 2.2kΩ 5%, 2W
R6 = 560Ω 5%, 5W

Capacitors

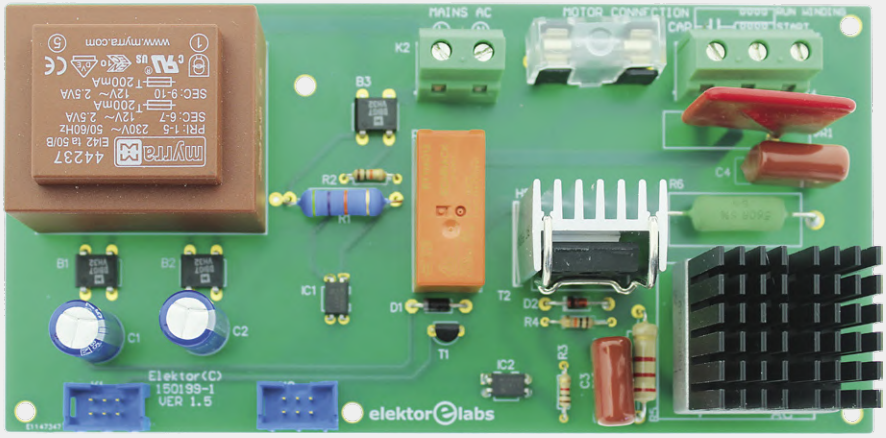
C1,C2 = 470µF, 50V
C3,C4 = 0.22µF, 630VDC, MPET

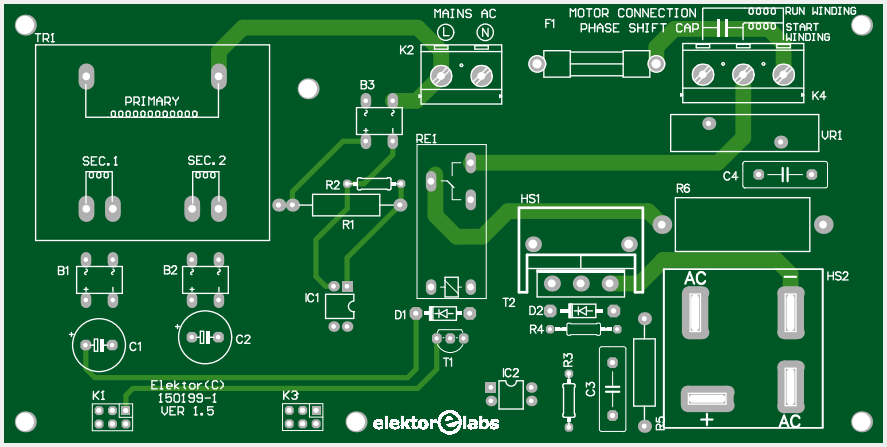
Semiconductors

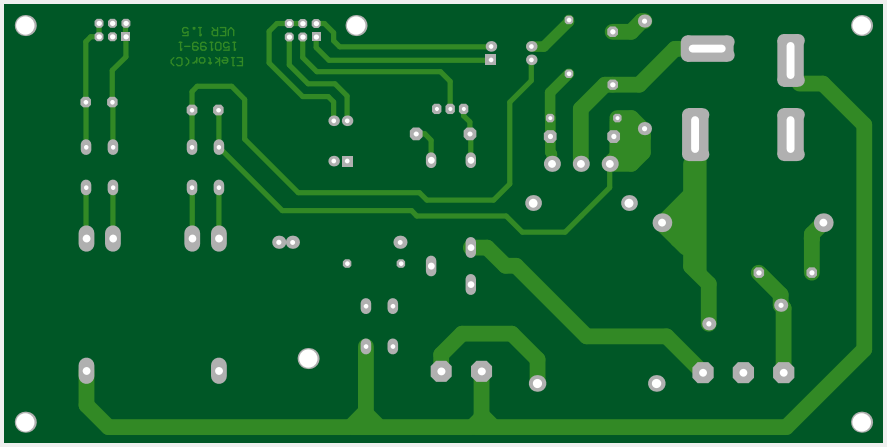
B1,B2,B3 = DB107, 1000V, 1A bridge rectifier
B4 = KBPC3510, 1kV, 35A bridge rectifier
D1 = 1N4007-T
D2 = 1N4742A, 12V, 1W zener diode
IC1,IC2 = PC817X3NSZ1B optocoupler
T1 = BC547B
T2 = STW26NM60N, 20A, 600V, N-Channel MOSFET

Miscellaneous

K1,K3 = 6-way (2x3) boxheader
K2 = 2-way PCB screw terminal block, 0.3" (7.62mm) pitch
K4 = 3-way PCB screw terminal block, 0.3" (7.62mm) pitch
HS1 = TO-247 heatsink type WV-T247-101E
HS2 = heatsink type 658-60ABT1E (incl. adhesive film)
F1 = 6A(T) (time delay; slow blow), 5x20mm
PCB mount fuse holder, 5x20mm, with cover
RE1 = G2R-14-DC12 (Omron)
TR1 = Power transformer, 2 x 12V, 200 mA, 230V primary, myrra type 44327, Farnell # 1214601
VR1 = V25S275P varistor, 275VAC 470J, 700V clamp, 25mm pitch
PCB 150199-1 v.1.5 from Elektor Store







70% of real size

COMPONENT LIST

MCU board, no. 150199-2

Resistors

R1-R7 = 4.7kΩ 5%, 250mW, 250V
R8,R9,R10 = 1kΩ 5%, 250mW, 250V
R11 = 3.3kΩ 5%, 250mW, 250V
R12 = 10kΩ 5%, 250mW, 250V

Capacitors

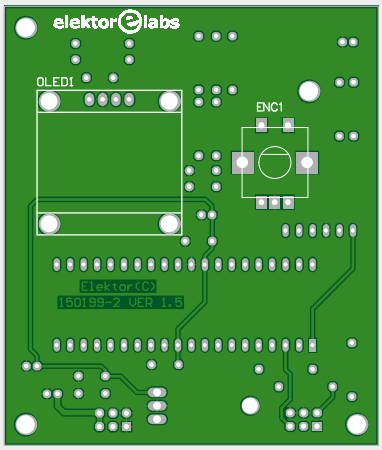
C1,C2,C3,C8-C11 = 0.1µF, 50V
C4,C5,C6 = 10µF 50V, electrolytic
C7 = 1µF 50V, electrolytic

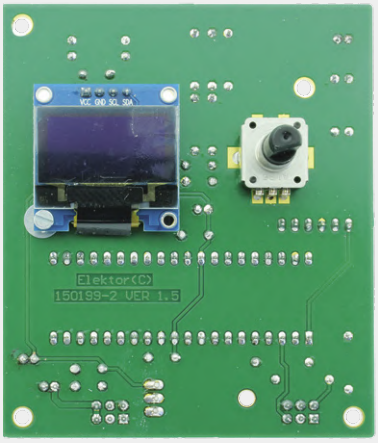
Semiconductors

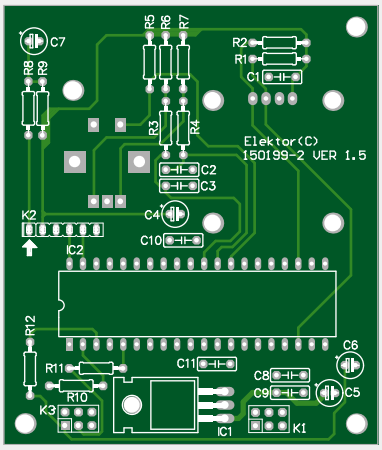
IC1 = MC7805CTG
IC2 = PIC18F45K22-E/P, programmed, no. 150199-41 from Elektor Store

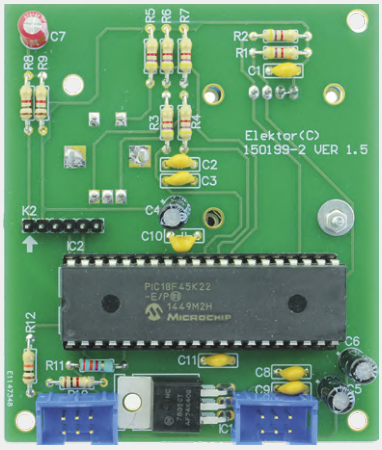
Miscellaneous

K1,K3 = 6-way (2x3) boxheader
K2 = 6-way SIL pinheader, 0.1" pitch
LCD1 = 0.96" OLED display module, I2C, from Elektor Store, SKU 18747
ENC1 = rotary encoder with pushbutton, Alps EC12E2424407
40-way DIP IC socket, 0.1" pitch
Length of 6-way flatcable and 4 IDC connectors for K1,K3 board interconnections
PCB 150199-2 v. 1.5 from Elektor Store









70% of real size

type we used nylon standoffs with internal threading at one end, which afford isolation even if secured by metal screws on the outside.
Inspect your construction thoroughly and if necessary, ask a second opinion from an expert.

Testing

Out of its case and without the internal cabling to the IEC AC-in and AC-out connectors, the electronics should only be tested by an experienced engineer using an approved, fully isolating transformer. A known-good AC load such as a 200 to 500-watt 230-V single-phase AC motor should be used for the load. All cabling to and from the motor driver should be approved and intrinsically safe. Your workbench should be clear. Do not proceed if one of the steps below yields wrong or unclear results.

- Interconnect the boards with the two 6-way IDC terminated flatcables (**Figure 6**).
- If already fitted, remove MCU IC2 from its socket.
- Connect the isolated 230-VAC supply so it reaches connector K2 on the power electronics board.
- On the MCU board, check the supply voltage on IC1 pin 1 (7805 input), it should read 14-16 VDC. Pin 3 (7805 output) should read 5 VDC.
- On the power electronics board, check the voltage across rectifier B2's + and – pins; it should read 14-16 VDC.
- Disconnect the isolated 230-VAC supply and insert IC2 on the MCU board.
- Skip this step if you are using the Elektor-supplied preprogrammed controller no. 150199-41. Otherwise, program the PIC with the hex file provided at [1] using a PIC programmer like the PICKIT3. AC power needs to be on for the unit to be programmed. Switch off the isolated 230-VAC supply, pull the programmer ICSP connector from the MCU board.

@ WWW.ELEKTOR.COM

→ **Single-phase 1-kilowatt AC Motor Drive, Power Electronics board v.1.5**
www.elektor.com/150199-1

→ **Single-phase 1-kilowatt AC Motor Drive, MCU board v.1.5**
www.elektor.com/150199-2

→ **Single-phase 1-kilowatt AC Motor Drive, Programmed PIC18F45K22-E/P**
www.elektor.com/150199-41

→ **0.96", 128x64 OLED display, I²C, 4-pin, Elektor SKU 18747**
www.elektor.com/blue-0-96-oled-display-i2c-4-pin

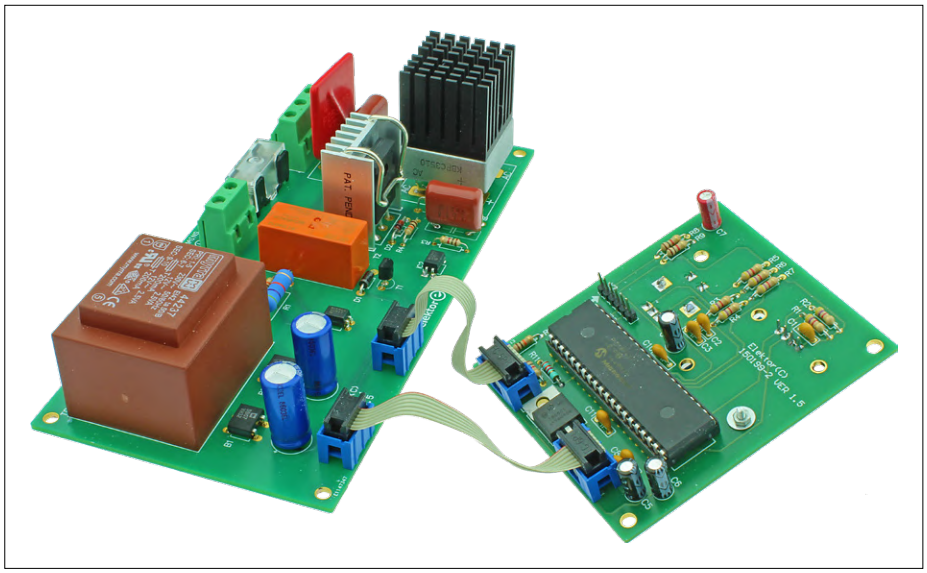


Figure 6: The MCU board and the power electronics board connected with two 6-way flatcables.

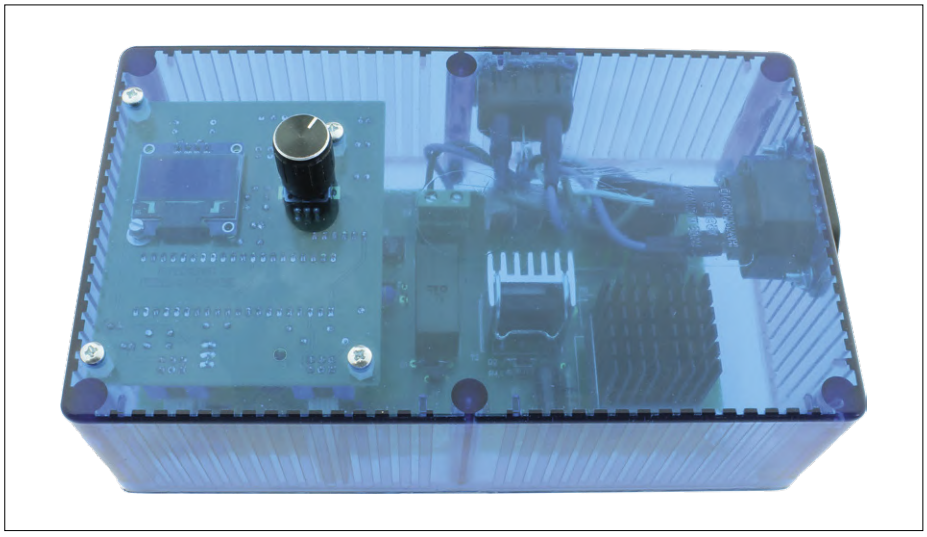
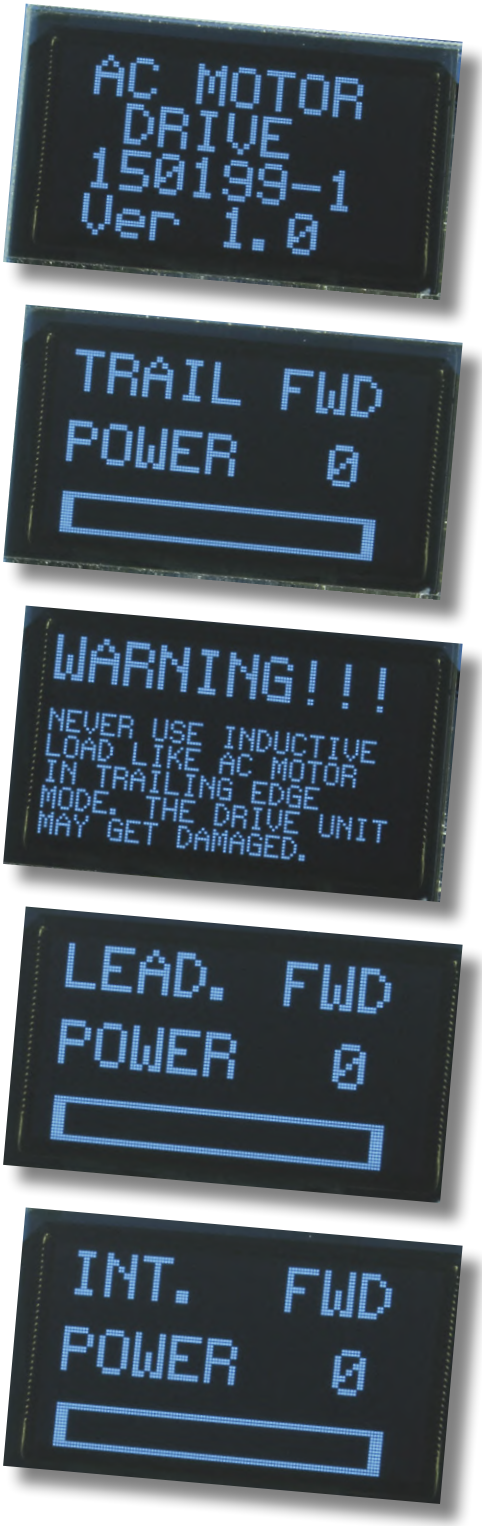


Figure 7: Tested, approved, and well protected in its blue-transparent case: the 1-kilowatt, 3-mode AC power control.



Web Link

[1] Project web page: www.elektormagazine.com/191142-01

- Connect the AC motor to K4 as indicated in **Figure 5**.
- Connect the isolated 230-VAC supply; the OLED will show the welcome message and offer the first mode of operation: that's "Int." for Integral-cycle Switching. Motor power should be 0%. Motor direction is shown in the right-hand top corner of the display.
- If the motor is running unloaded or out of its normal mounting or cradle then precautions should be taken for jerk, mechanical shock or vibration.
- Turn the encoder to vary the motor power up to 100%. A short press of the rotary encoder shaft should reverse the motor's direction at 0% of power every time.
- Disconnect the motor.
- Check driver modes available by long presses on the rotary encoder shaft. The response should cycle through: [Int. → Lead. → Trail. →].

Congratulations if everything checks out.

Now proceed to fitting the electronics and reconnecting all wiring inside the case and then closing it. We did that in the Elektor Labs and then shot the photo in **Figure 7**.

That completes our discussion of the project. However, with "much written & more to say", the last line should read, undisputedly:

safety first. ⏮

191142-01

RPi Gets a Fast 3.5” Touch Display

More power at no extra charge

By **Mathias Claußen** (Elektor Labs)

The Waveshare 3.5” touchscreen LCD is not only affordable, but also a special alternative to previous small displays for the Raspberry Pi, all thanks to its 50 fps. It’s ideal where other displays in this price range only deliver a ‘slide show’ because of their low frame rates.



Waveshare has updated its product range and introduced a new 3.5” display for the Raspberry Pi. Most affordable RPi displays are connected serially through SPI. The maximum clock frequency is then 20 MHz and occasionally 26 MHz for writing data. This significantly limits the update rate of the screen.

Speed

If a complete screen content is to be transferred to the display, 480 × 320 pixels at 16 bits are required for the colour per pixel. So a total of 307.2 kB is to be conveyed to create a completely new image. In addition, there is an overhead of a few bytes for each transfer. Under ideal conditions, this requires a 125 ms transfer time at 20 MHz SPI clock, which would only be enough for about eight frames per second assuming if the display is permanently rewritten. This is no longer acceptable for a video and is not even acceptable for the operation of some programs, as the sluggish screen output makes it difficult to click on buttons – not to mention games requiring a quick response, or the difficulty of positioning a video camera. But if you’re looking for a faster display for easy plugging into an RPi, you’ll find it here. The new 3.5” display from Waveshare offers the usual 480 × 320 pixels with 16-bit colour at a remarkably high SPI clock rate of 125 MHz! Since the overhead is relatively small, the resulting 302.7 kB per image can be transmitted 50 times per second. This allows a smooth image composition and makes the display ideal for video, games or the smooth operation of a browser.

Differences

On the surface, the new display for RPi from Waveshare (**Figure 1**) is similar to the conventional models from other man-



Figure 1: The new Waveshare display hardly differs from ‘normal’ displays supplied by other manufacturers.

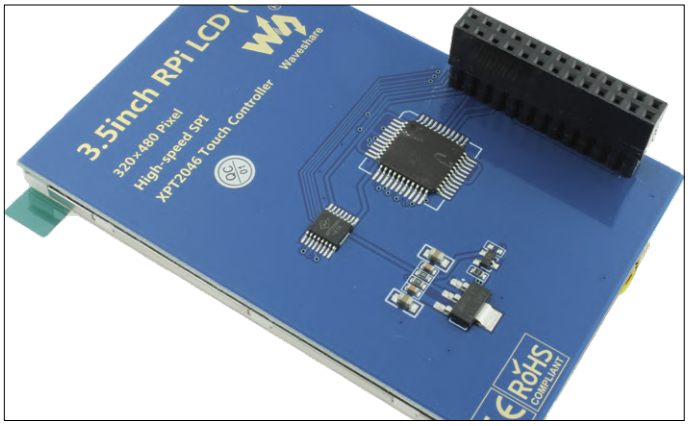


Figure 2: Rear side of the Waveshare display. Only two ICs, a voltage regulator and some capacitors are visible.

Web Links

[1] Display Drivers: www.waveshare.com/w/upload/1/1c/LCD-show-181201.tar.gz

[2] 'Big Buck Bunny' film: http://bbb3d.renderfarming.net/download.html

[3] 'Big Buck Bunny' and Doom on the Waveshare display: www.elektormagazine.com/190271-03

ufacturers. But a look at the rear (**Figure 2**) reveals elementary differences: there are only two ICs, a voltage regulator and a few capacitors left. Normally you'd find a shift register (74HC4094), a counter (74HC4040) and a hex inverter (74HC04) on the back of the display (**Figure 3**) acting as SPI/Parallel converters. Apparently, these ICs have been replaced by a new chip, whose type number has been sanded off – the exact type of silicon remains a company secret. In any case, it should be clear that this chip enables operation with an SPI clock rate of 125 MHz. Okay you could play detective, connect a logic analyser and examine the signals from the IC to the display. But this kind of reverse engineering is beyond the scope of this discussion and possibly illegal.

Setup

Enough of the technical details — here comes a short practical test: First, plug the display into the RPi and connect the keyboard, mouse and monitor. A current Raspbian with updates is used for setup and the latest firmware is provided. The display driver can be found on the Waveshare website. The current version can be downloaded under [1]. Users experienced in the disciplines Terminal or Console can get the file on the RPi via `wget`, or start a download via a browser.

The downloaded file has to be unpacked first. Then you have to make the installer (i.e. a shell script for the display) executable. The easiest way to do this is via the GUI by unpacking the contents of the archive into the home directory and making the LCD35C-show file executable there. Running the installer then sets up everything necessary for the operation of the display, meaning that the playback of videos by the fast display hardware is now possible without problems.

For the installation you need Internet access to download some packages and tools. The driver is installed via Terminal. Next you can switch to the folder of the driver you just unpacked. From there you make the installer executable with the command `chmod 777 LCD35C-show` and then execute it with `sudo ./LCD35C-show`. Once the script for setup has been run, restart the RPi once.

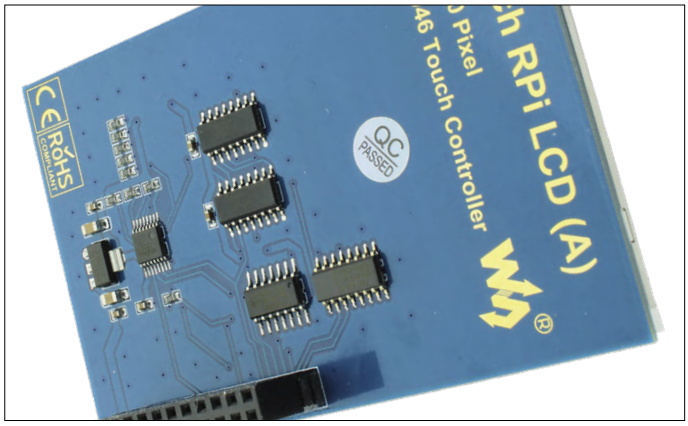


Figure 3: This is what the rear of conventional RPi displays looks like.

First tests

If everything went as expected, the graphical output should start shortly after booting and the desktop should be shown on the display. The video 'Big Buck Bunny' [2] was used to test the display 'under stress' i.e. in 1080p resolution at 30 frames per second. The video is played in FullHD (1920 × 1080 pixels) at 30 fps via OMXPlayer. For you to see the result yourself, we have prepared a test video which is available on the Elektor Labs Magazine website at [3]. You will see that the display has no problems with video playback. Even a round of DOOM is no problem. What remains basically problematic with displays connected via SPI is the use of OpenGL software such as SuperTuxKart or other applications that employ 3D acceleration. As soon as the overlay for the driver *vc4-KMS-overlay.dtb* is specified in */boot/config.txt*, the output is forced via HDMI. The most interesting question might be if the fast version of Waveshare is more expensive than the other ones listed in the Elektor Store. Fortunately, it is not more expensive. However, the advantage of high speed comes with a slightly lower viewing angle stability and brightness than with the displays from other manufacturers. This can be relevant in office environments with direct light on the display.

Conclusion

In telegram style: at this price, there is hardly any reason to use another display that's slower than the 50 frames per second offered by the Waveshare device. The "hardly" refers to the use under difficult lighting conditions, because then other displays are the better choice. Elektor would recommend the new 3.5" Waveshare display for almost all cases where a display is to be built directly into a compact sandwich with an RPi. Only those who use applications that rely on OpenGL should better orient themselves elsewhere to avoid surprises later. A look at the 5" JOY-iT touch display would be valuable. Apart from that, why should a 3.5" display be sluggish if you get this luxury at no extra charge? ⏮

190271-03

@ WWW.ELEKTOR.COM

→ Waveshare 3.5" touch screen for Raspberry Pi
www.elektor.com/18936

→ JOY-iT 5" touch screen for Raspberry Pi
www.elektor.com/18146

Lab, Sweet Lab

A glimpse of The Holy Place — no unauthorized entry.

By **Eric Bogers** (Elektor Netherlands)

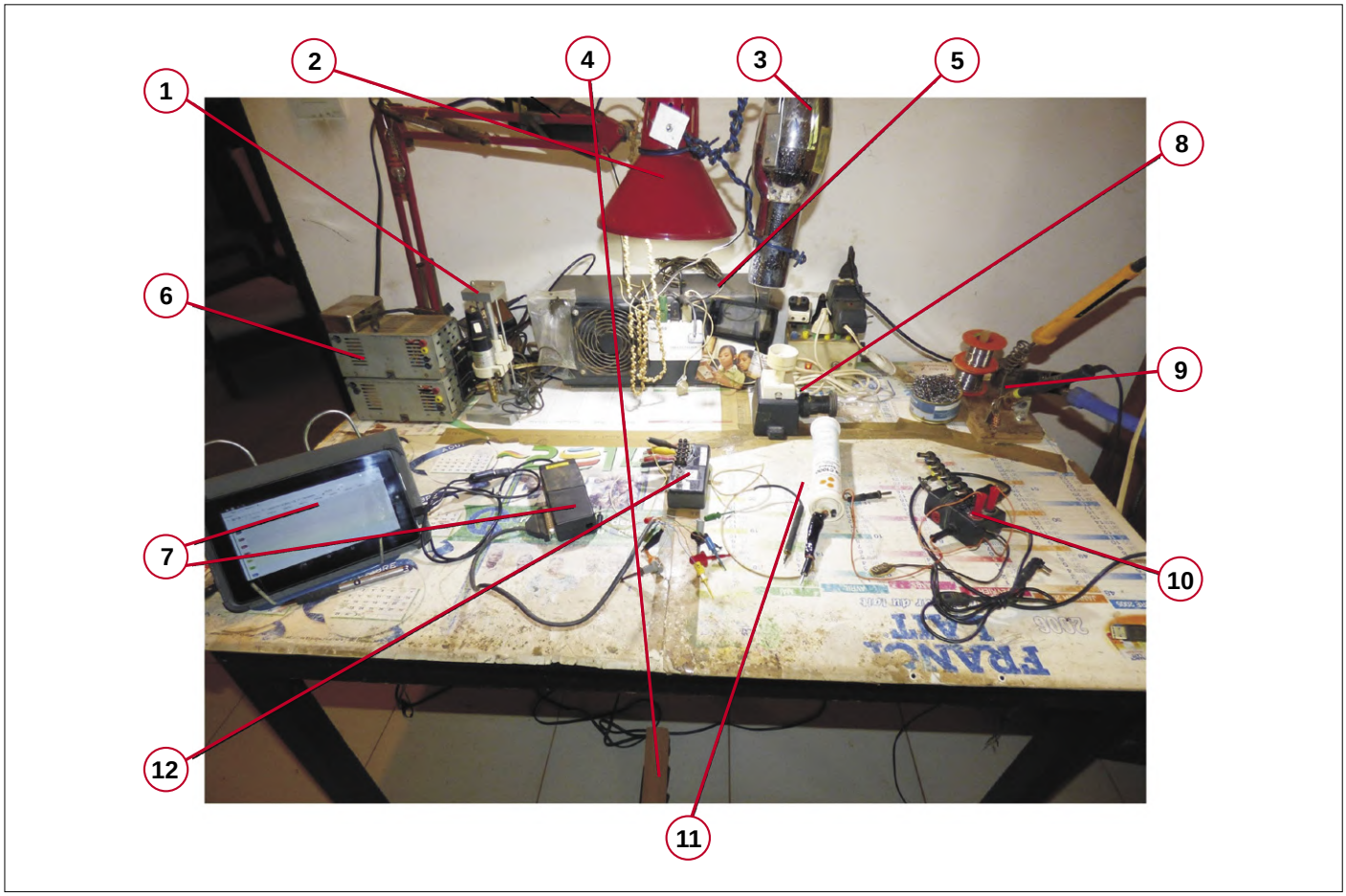
Back in July 2019, Elektor organized a competition on their Labs website to find out who has the most beautiful / messiest / interesting / most remarkable home laboratory. In the coming issues of ElektorLabs Magazine we will publish a selection from the entries.

We seldom care to think about it, but to an electronics enthusiast, (Western) Europe is close to paradisal as far as the pastime is concerned. Admittedly, the 'parts shop around the corner' is dying. Alas, in those shops with endless rows of parts drawers there was always a kind soul ready to listen and return good advice. On the positive side, our large mail-order companies deliver what we need quickly and for relatively little money.

A totally different situation exists for André Aguila who lives and works in Ouagadougou in Burkina Faso. It's not exactly easy to find affordable electronics tools or parts there, and getting them delivered by mail order is expensive and requires a lot of patience. André writes:

"In my student days, I started electronics as a hobby; after that I was off the e-track for a long time and only picked it up again a few years back. I can't spend too much money on my hobby, so I made a lot of tools and ancillaries myself. The images in this article give an impression of the range.

Right now, I am building my own mini reflow oven for housing in an 'empty' 5¼-inch floppy-disk drive case. Two halogen lamps should provide the necessary heat." Here at Elektor we are keen to know how that project will end!



Some of André's comments with the photos:

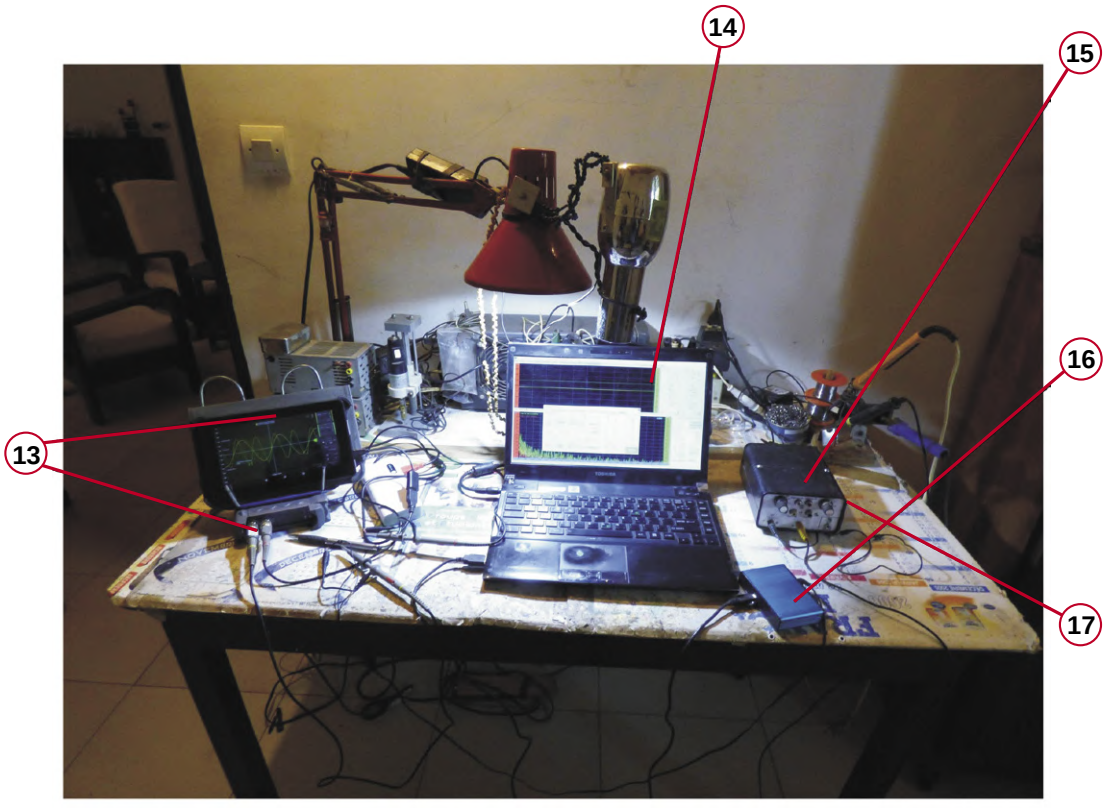
1 and 2: "The mini drill and the lamp are from my student days; the shell necklace was a gift from a fellow student from Tahiti."
3: "This is not a hairdryer anymore — it now extracts badl fumes (airflow reversed, of course, with a carbon filter on the back)."
4: "With this foot pedal I can switch the extractor on and off."
5: "This isn't a UPS or anything, it's an isolating transformer. In fact, these are two 220/110-volt transformers whose 110-V windings are linked; only one of them is a genuine isolating transformer."
6: "Power supplies scrapped from 5¼" floppy-disk drives."
7: "Cheap tablet PC with USB on-the-go, and a logic analyzer (8 channels at 24 MHz or 16 channels at 12 MHz)."
8: "A bulb in series with the 220 V mains protects the connected circuit in case of a hidden short circuit."
9: "Home-made stand for soldering irons."
10: "Simple multi-adapter for testing purposes; I described it on my Elektor Labs page [1]."
11: "One of my first measuring instruments: a simple continuity tester (probably an Elektor design)."
12: "A simple Go/No-Go transistor tester from Elektor [2]."
13: "Cheap USB oscilloscope, connected to a cheap tablet and powered by a powerbank. The Android software was not free, but it was cheap and worth every penny I paid for it."
14: "My hobby laptop — built back in 2011! I have a lot of electronics and programming software on it. The picture shows Visual Analyzer acting as a signal generator (unfortunately only for audio frequencies...).".
15: "This 'amplifier' boosts the signal levels generated by the laptop. With this amp I tested a homemade Class-A hi-fi amplifier, among other things."
16: "Cheap external USB sound card."
17: "On the back of this amplifier sits an attenuator (also from Elektor) to back off input signals if necessary.

(191139-02)

Web Links

[1] Multi-Adapter: www.elektormagazine.com/labs/comfortable-and-inexpensive-v-a-controller-civac-1-1

[2] Transistor Tester: www.elektormagazine.com/magazine/elektor-201407/26904

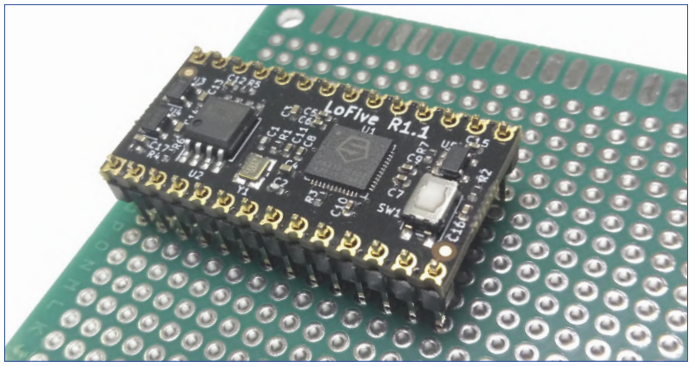


First Steps with RISC-V

Trying out the LoFive Board

By **Tam Hanna**

After ARM, RISC-V could be the next big thing on the processor market. The instruction set architecture driven by a large consortium [1] is license-free and suitable for both embedded and computer processors. However, there are still only a few microcontrollers and boards to buy. Elektor author Tam Hanna bit the bullet and put the inexpensive 'LoFive' board to the test.



After ARM, RISC-V could be the next big thing on the processor market. The instruction set architecture driven by a large consortium [1] is license-free and suitable for both embedded and computer processors. However, there are still only a few microcontrollers and boards to buy. Elektor author Tam Hanna bit the bullet and put the inexpensive 'LoFive' board to the test.

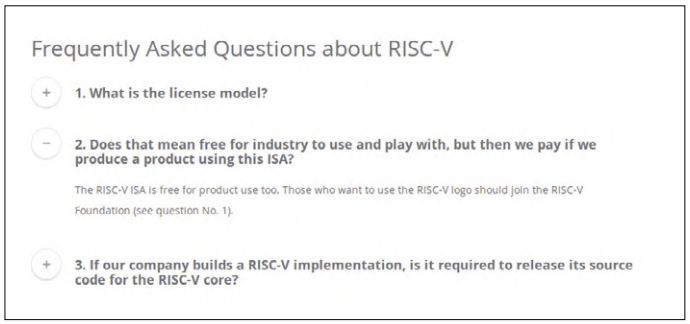


Figure 1: The RISC-V architecture can be used free of charge and does not include any viral licenses.

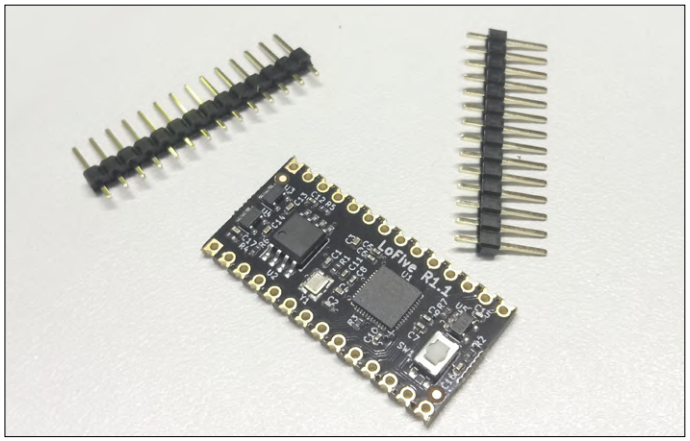


Figure 2: The board can also be used as a surface mount module.

While the ARM architecture as a whole is strictly patented, the permanent further development of the x86 architecture ensures that a provider of a current x86 core also has to pay massive license fees.

Developed at the University of Berkeley, the RISC-V architecture has the explicit goal of poaching in both the embedded and 'larger' environments. By far the most important argument of the vendors is the freedom of license fees — **Figure 1** was taken from the website under [2] and shows that the use of the architecture is free of charge.

While the RISC-V architecture has been undergoing development for many years, real hardware has only been available for a relatively short time. GroupGets offers a cheap evaluation board bearing the name LoFive with a Freedom E310 RISC-V processor from SiFive [3].

With the rapid development of the architecture, there are now two versions of it — we want to work with the more up to date version called R1 or 1.1. Note that the older version is technically very different from its successor, and the instructions discussed here cannot be implemented 1 on 1.

Prepping

If you unpack the LoFive, you will see a group of exposed headers (**Figure 2)**. The manufacturer relies on 'combinatorial' pinouts, which, in addition to soldering 0.1-inch pitch headers, also allow the board to be soldered directly to another planar. For convenience, the author will work with a classic plug-in board in the following steps.

The LoFive board currently sells for around €25, which is possible because the provider does not integrate a programming chip, among other things. Instead, you need to get hold of an FT2232H-56Q USB/Serial-Adapter board from FTDI yourself; the correct connections between the boards are summarized in **Table 1**.

If you are serious about the LoFive, you can build a 'carrier board' instead. However, for first experiments it is sufficient

Table 1. Connection of LoFive and programming adapter.	
LoFive Pin	**FTDI Breakout Pin**
+5Vin	VBS
GND	GND
TRSTN	AD5
TCK	AD0
TDO	AD2
TMS	AD3
TDI	AD1
UART0.TX	BD1
UART0.RX	BD0

to use Dupont cables (**Figure 3**). With EMI in mind and many LED lamps present in the author's rather unstable laboratory, 10-centimetre-long wires worked for a good part of the time without problems. Occasionally occurring glitches in the recognition of the target had disappeared at the next pass.

Dev work kickoff

At the author's workstation running Ubuntu 18.04, most of the required elements were already on board. Enter the following command to load the packages:

```
sudo apt-get install autoconf automake libmpc-dev
   libmpfr-dev libgmp-dev gawk bison flex texinfo
   libtool libusb-1.0-0-dev make g++ pkg-config
   libexpat1-dev zlib1g-dev
```

Windows and Mac OS users however are left in the cold. Linux has more or less established itself as the standard in the world of RISC-V development; especially when working with boards that are not widely used, Windows offers little prospect. The Eclipse-based Freedom Studio, developed by processor manufacturer SiFive, works with the 'official' HiFive development board. The LoFive finds no love in the IDE at the time of printing this issue. In the next step, you can request the download of the SDK via the well-known git client:

```
tamhan@TAMHAN18:~$ cd riscvneu/
tamhan@TAMHAN18:~/riscvneu$ git clone --recursive
   https://github.com/mwelling/freedom-e-sdk.git
. . .
tamhan@TAMHAN18:~/riscvneu$ cd freedom-e-sdk
tamhan@TAMHAN18:~/riscvneu/freedom-e-sdk$ git
   checkout lofive-r1
M            freedom-devicetree-tools
. . .
```

What is new here is that after downloading the main archive we have to load a submodule. It is necessary to provide the LoFive-R1 specific modules.
The next step is to request Github to synchronize and then download another submodule:

```
tamhan@TAMHAN18:~/riscvneu/freedom-e-sdk$ git
   submodule sync
Synchronizing submodule url for 'doc/html'
. . .
tamhan@TAMHAN18:~/riscvneu/freedom-e-sdk$ git
   submodule update --init --recursive
. . .
Submodule path 'freedom-devicetree-tools': checked
   out '4f25a7512696f0b41c17e517d39d097499b931a7'
```

The call of `sync` is not absolutely necessary — in the author's tests there were error messages, so in case of doubt a more cautious approach is reasonable.
The RISC-V toolchain generally relies on the GCC compiler and OpenOCD; some other evaluation boards use a Segger-series programming interface instead.

Since the compilation of the toolchain on the workstation takes a relatively long time, SiFive now provides a more or less finished package. In the first step, open the URL https://www.sifive.com/boards in a browser of your choice and scroll down to the Prebuilt RISC-V GCC Toolchain and Emulator passage as shown in **Figure 4**.

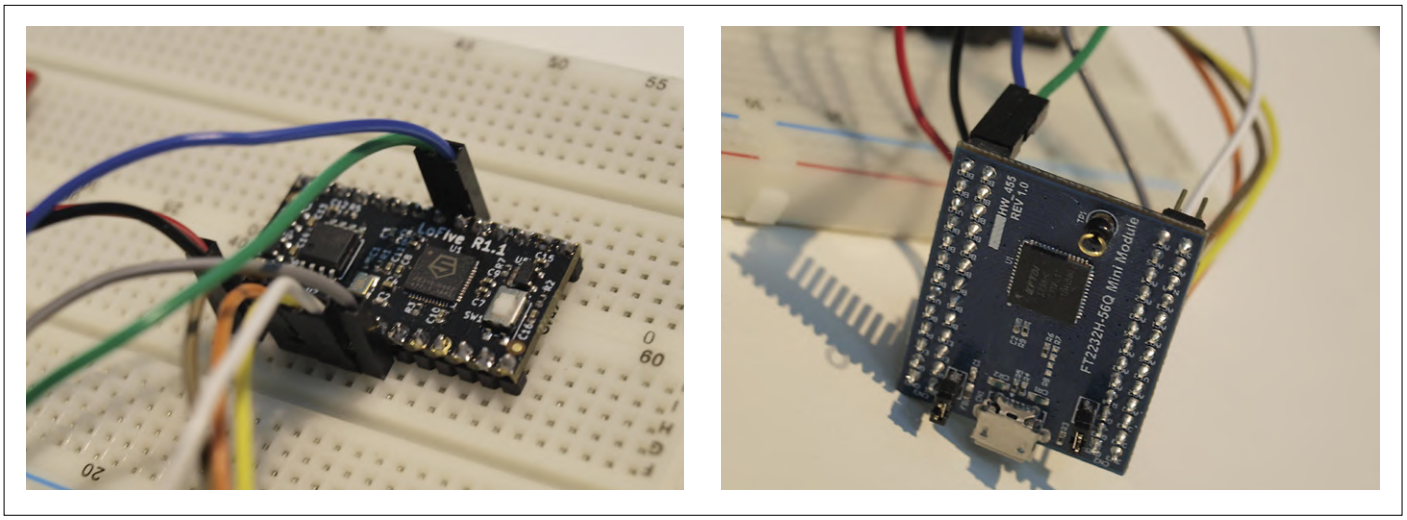


Figure 3: A USB/serial module from FTDI is used for programming.

In any case, for the following steps we need the versions *GNU Embedded Toolchain - v2019.08.0* and *OpenOCD - v2019.08.2*. Click on the two links and then extract your contents into subdirectories of the working directory. The author works in the following steps with the directory name *riscvneu*; the output of the `dir` command is as follows:

```
tamhan@TAMHAN18:~/riscvneu/$ dir
freedom-e-sdk/
riscv64-unknown-elf-gcc-8.3.0-2019.08.0-x86_64-linux-
   ubuntu14/
riscv-openocd-0.10.0-2019.08.2-x86_64-linux-ubuntu14/
```

For the sake of convenience, the toolchain assumes that compilers and co. can be found via environment variables — its purpose being: installing a new toolchain becomes effective by making a small change to the environment-variable.
Since changes to the PATH parameter are generally degenerating into work, we want to work with temporary paths in the following steps:

```
tamhan@TAMHAN18:~/riscvneu/freedom-e-sdk$ export
   RISCV_OPENOCD_PATH=/home/tamhan/riscvneu/riscv-
   openocd-0.10.0-2019.08.2-x86_64-linux-ubuntu14/
tamhan@TAMHAN18:~/riscvneu/freedom-e-sdk$ export
   RISCV_PATH=/home/tamhan/riscvneu/riscv64-unknown-
   elf-gcc-8.3.0-2019.08.0-x86_64-linux-ubuntu14/
```

Note that the export declaration is only valid in the terminal window in which it was entered. If you want to use another window or close the window unexpectedly, you have to start the process again. It should also be noted that the printed directory values only apply on the author's workstation — it is very unlikely that your username is pronounced *tamhan*.

OpenOCD configuration

OpenOCD is a relatively complicated tool for Linux embedded programming. It contacts a target system via various systems, also known as *probes*, in order to receive commands via a network port on the workstation. Normally OpenOCD appears 'on board' with the GDB debugger.

The problem with OpenOCD is that the configuration in .conf files is very strict. The developer, who adapted the toolchain to the RISC-V processor, does not work with Ubuntu 18.04. Therefore, errors of the following type may occur during execution:

```
tamhan@TAMHAN18:~/riscvspace/freedom-e-sdk$ sudo make
   upload PROGRAM=led_fade BOARD=freedom-e300-lofive
[sudo] password for tamhan:
. . .

Error: unable to open ftdi device with vid 0403, pid
   6010, description 'FT2232H-56Q MiniModule', serial
   '*' at bus location '*'
```

If your system encounters the error shown here, the first step is to determine which description text the kernel driver assigns to the connected FTDI module. The most convenient way to do this is to use lsusb:

```
tamhan@TAMHAN18:~$ lsusb
. . .
Bus 001 Device 009: ID 0403:6010 Future Technology
   Devices International, Ltd FT2232C Dual USB-UART/
   FIFO IC
```

In the next step, we have to look for the problematic description. This is most easily done by the command line Oldie grep; the output looks like this:

```
root@TAMHAN18:~/riscvneu/freedom-e-sdk#  grep -r
   FT2232 *
bsp/lofive-r1-bootloader/openocd.cfg:ftdi_device_desc
   "FT2232H-56Q MiniModule"
bsp/lofive-r1/openocd.cfg:ftdi_device_desc "FT2232H-
   56Q MiniModule"
```

Then open the two files with an editor of your choice to find the following declaration block:

```
interface ftdi
#ftdi_device_desc "Dual RS232-HS"
ftdi_device_desc "FT2232C Dual USB-UART/FIFO IC"
ftdi_vid_pid 0x0403 0x6010
```

In the first step, this informs OpenOCD that it is dealing with an FTDI interface. In the next step, in addition to the two numeric IDs, a description text follows, which together should make it possible to find the target device.
In this case, the combination of VID and PID is sufficient, so we can simplify the configuration:

```
interface ftdi
ftdi_vid_pid 0x0403 0x6010
```

Ready, steady, go

One problem one has to deal with since the advent of the first evaluation board is definitely outdated firmware. That's quite logical since the boards don't arrive on the developer's desk directly from the manufacturer — they are in the hands of a distributor for some time, where the included software gathers dust.
So, our first act is to update the bootloader. To do this, we need to uninstall the configuration tool in the first step, as it contains artefacts from the vendor's machine:

```
root@TAMHAN18:~/riscvneu/freedom-e-sdk# make
   PROGRAM=lofive-boot TARGET=lofive-r1-bootloader
   clean
make -C /home/tamhan/riscvneu/freedom-e-sdk/software/
   lofive-boot PORT_DIR= clean
make[1]: Entering directory '/home/tamhan/riscvneu/
   freedom-e-sdk/software/lofive-boot'
. . .
```

Those who interact with a RISC-V board via the Freedom SDK generally do so via the Make tool. It expects a command to be executed in addition to the two variables `PROGRAM` and `TARGET`. The actual delivery of the bootloader code then takes place as follows:

```
root@TAMHAN18:~/riscvneu/freedom-e-sdk# make
   PROGRAM=lofive-boot TARGET=lofive-r1-bootloader
   upload
cd /home/tamhan/riscvneu/freedom-e-sdk/bsp/lofive-r1-
   bootloader/build/debug/ && \
```

The author uses the various Make commands from a root shell for convenience. If you don't want to do this, you can set the various permissions manually.
After the successful delivery of the bootloader we can focus on a first program:

```
root@TAMHAN18:~/riscvneu/freedom-e-sdk# make
   PROGRAM=sifive-welcome TARGET=lofive-r1 upload
```

After completing this command, you will see an, admittedly slow, waveform at the PWM outputs. If you see this on your oscilloscope (roll mode required or recommended!), you have successfully completed the configuration. Our example program also provides status information during processing, which can be extracted using the Screen command or any other terminal emulator:

```
tamhan@TAMHAN18:~$ screen /dev/ttyUSB1 115200
[screen is terminating]
tamhan@TAMHAN18:~$
```

When using this command, please note that you must start Screen before the actual program. Incidentally, the screen terminal cannot be stopped easily — the easiest way is to press Control-A and then press K. Screen then asks if you really want to close the window. The correct answer here is pressing Y.

Where is the code

Last but not least, in this article I would like to briefly address the question of where the 'code to be processed' gets stored. The answer is the software directory; in the case of our project example, the content is presented as follows:

```
root@TAMHAN18:~/riscvneu/freedom-e-sdk/software/
   sifive-welcome# ls
debug  LICENSE  LICENSE.Apache2  LICENSE.MIT
   Makefile  README.md  sifive-welcome.c
```

The Makefile responsible for controlling the various compilation processes is generally limited to including a Makefile specified by SiFive — it then ensures that the various commands such as Upload are implemented.

For reasons of brevity we have to end our review at this point — just note that the toolchain still doesn't know what to do with C++ at the time this issue went to print.

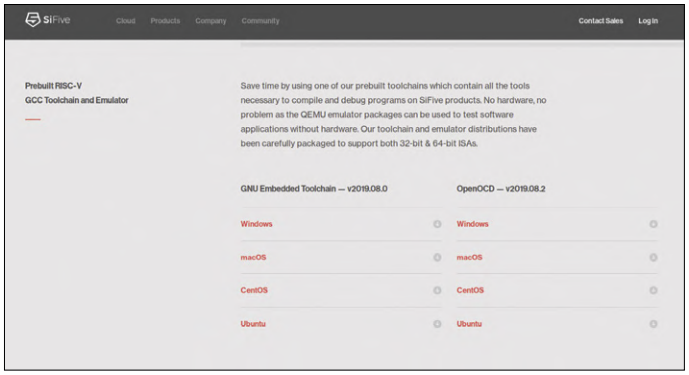


Figure 4: The download option for the RISC-V toolchain is well hidden.

Is it worth it?

Anyone hunting for an inexpensive microcontroller today will certainly find ST, Microchip and the various Chinese suppliers easier than the RISC-V. Even an 'open' architecture doesn't ensure that the silicon is free of charge: Because of the small number of copies sold, the scaling effects of ARM and other proprietary architectures are actually cheaper at the end of the day.
The work with RISC-V is certainly worthwhile for people who have either "academic" or hacking interest in the RISC-V platform. However, due to the immensely broad support by many participating companies, it can be assumed that the platform can at least achieve some respectable success in the embedded sector. ⏮

191138-04

@ WWW.ELEKTOR.COM
→ E-Book: ARM Microcontroller Projects
www.elektor.com/arm-microcontroller-projects-e-book

Web Links

[1] RISC-V Foundation: https://riscv.org/
[2] RISC-V FAQ: https://riscv.org/faq/
[3] LoFive on GroupGets: https://groupgets.com/manufacturers/qwerty-embedded-design/products/lofive-risc-v

eLABS PROJECT

Elektor Labs Challenged: a DIY LoRa Tracker

Glitches and results in electronics development

By **Mathias Claußen** (Elektor Labs)

Numerous messages received from Elektor readers indicate beyond doubt that a mobile GPS tracker sporting LoRa functionality is a highly desirable device to have. In good spirit, Elektor Labs set out to fulfil this wish. Some of the pitfalls encountered during the development phase of the project can be read here!

A LoRa GPS tracker is currently being built at Elektor Labs, one which has already gone through several circuit board iterations and is now slowly approaching final production and publication. The idea behind the LoRa Tracker is quite simple: a mobile, battery-powered device that transmits its position through a LoRa WAN at a defined interval. Besides, the whole thing should fit into a 1551K case from Hammond and be powered by rechargeable cells that can be exchanged by the user. Batteries are eliminated to avoid waste. So only NiMh or Lithium batteries come into play.

Supply

The decision was made in favour of replaceable Lithium batteries in 10440 or AAA format, because with their typical voltage of around 3.6 V, the circuit can then simply be operated via an LDO voltage regulator. Since Lithium batteries are damaged by a deep discharge, an undervoltage switch-off was also considered. That way nothing can go wrong. The circuit can be seen in **Figure 1**.

As soon as the electronics starts up, a temporarily increased current will flow for charging the capacitors behind the voltage regulator. This current can rise high enough for the battery voltage to drop below 2.95 V for a short time. IC1 would then trigger an 'Under Voltage Lockout' and IC2 would be switched off. This can be remedied with C16, a 470-µF electrolytic capacitor parallel to the 100-nF capacitor C15 ahead of the voltage regulator, which buffers the temporarily increased current and thus keeps the voltage ahead of the regulator above the 3-V threshold. Consequently, IC1 only becomes active when the battery is 'flat'. When the first board was produced, we failed to consider this, but the problem was solved by retrofitting C16 and C15 piggybacked on the board. The modified circuit diagram can be seen in **Figure 2**.

Barriers...

The MCU was then switched on and connected to a debugger. And it responded like a good boy. So, everything seemed to work fine so far. Now for the exciting part: the GPS module

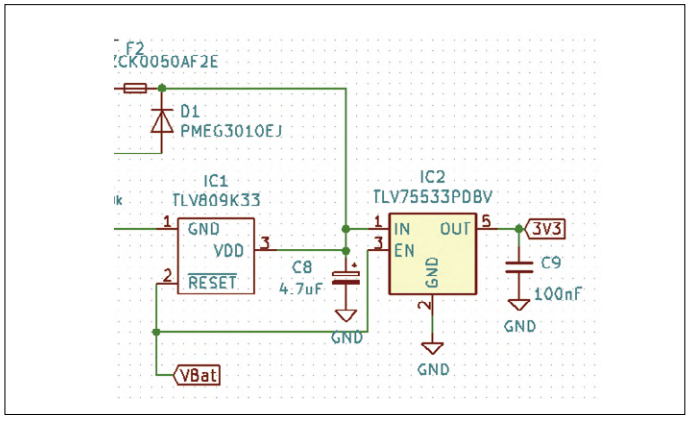


Figure 1: Circuit section comprising the power supply.

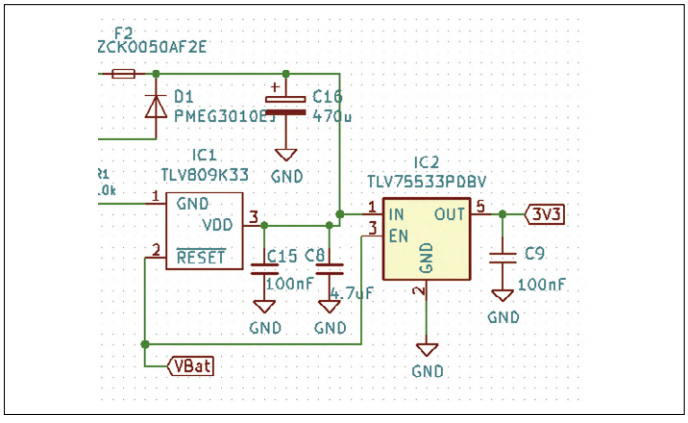


Figure 2: Modified circuit with electrolytic buffer capacitor.

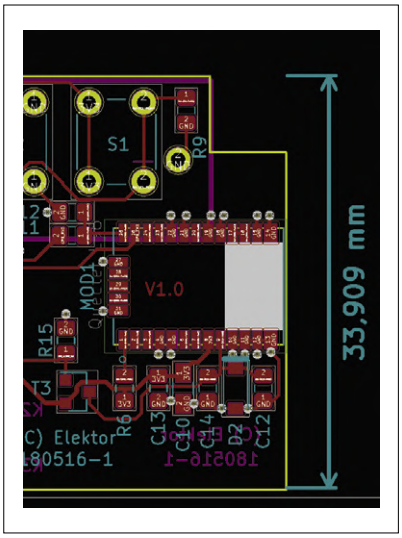


Figure 3: With this PCB version, insufficient space was left around the GPS module for a good reception.

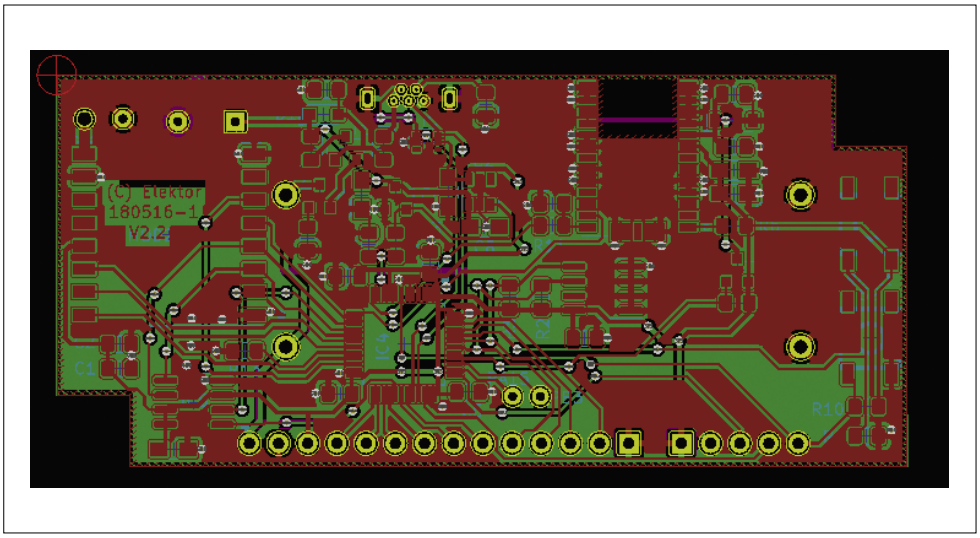


Figure 4: The other position on the long side and the ground plane together enable good reception with the improved PCB version.

had to be tested. Unfortunately, the copper was not removed from the board under the antenna of the module, which made radio reception impossible. After removing the copper, theoretically, the module should receive GPS data. But the design notes already refer to certain minimum ground plane dimensions necessary for reception.

The minimum required width is 45 mm + 10 mm left and right, and 20 mm depthwise. But as you can see from the layout of the circuit board **Figure 3**, unfortunately a mere 34 mm are available. Even a continuous ground plane without components was not feasible given the desired compact dimensions. So, no reception was possible, and a further improved version of the board was necessary.

The GPS module is only 14 × 10 mm in size and therefore the on-board antenna is only a few millimetres long and wide. Unfortunately, you can't miniaturize antennas as well as you can with semiconductors. The smaller they are, the larger the 'counterweight' (the ground plane) required. The dimensions and positioning of the improved circuit board in Figure 3 have been changed to comply with the specifications of the datasheet. When the module was tested again it received more than two satellites. After the usual ten to twelve minutes, a GPS fix was reached.

… and problems

For the test, the module was powered with 5 V from a notebook USB port. This is not a good idea for GPS receiver power supply as the cable also carries RF residue and stray signals which can couple into the receiver and interfere with it. When powered from a laboratory power supply, the reception was further improved, but still not optimal. It was found to hardly offer bad-weather headroom for use in the field. In addition, the setup with the GPS module used was not quite easy. Through deep pondering over that one though we found a solution that's better to solder, offers better reception and costs even less. There were also other bombshells regarding MCU, software and the mechanics. You can read more about this in the upcoming editions and currently on Elektor Labs [3]. ⏮

191155-03

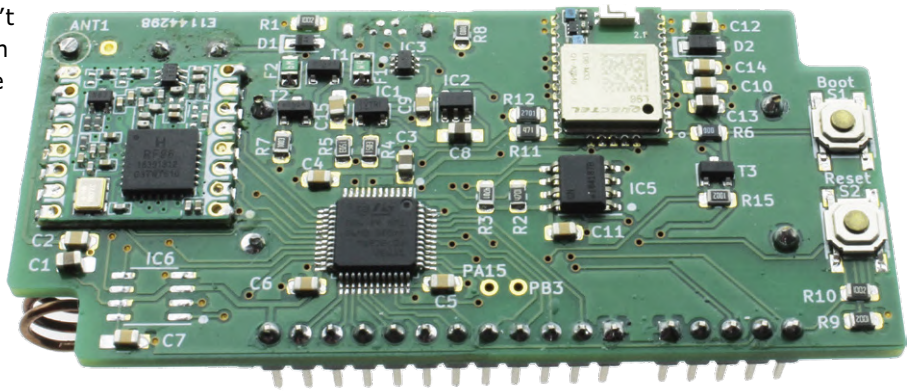


Figure 5. This is what the tested prototype looks like.

Web Links

[1] GPS Module (L96) Design Guide: www.quectel.com/UploadImage/Downlad/Quectel_L96_Hardware_Design_V1.2.pdf

[2] GPS Module (L96) Datasheet: www.quectel.com/UploadFile/Product/Quectel_L96_GNSS_Specification_V1.1.pdf

[3] Project page at ElektorLabs: www.elektormagazine.com/labs/lorawan-node-experimental-platform



Sigfox and the IoT (2)

Registration in the Sigfox network

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

In this part of the brief series on Sigfox we first explain the basic structure of the Sigfox's network and backend structure. Then we integrate our MKR FOX 1200 board into this worldwide communication network.

Figure 1 shows the basic structure of the Sigfox network. The Sigfox-Objects (also called Devices send their telegrams as broadcasts over the license-free 868-MHz ISM band. In addition to the station identification (*Station-ID*) each telegram contains a user data field, called *Payload*, with a maximum size of 12 bytes, meaning the user can transmit a maximum of 12 bytes of measurement or status data or other data with each transmission. This does not sound much at first, but since Sigfox is used at field level as a 0G network, this is more than enough. For example, 12 measured values of 1 byte or three measured values of 2 bytes each can be transmitted together with a GPS data set of 6 bytes.

Since the Sigfox network operates in the license-free ISM band,

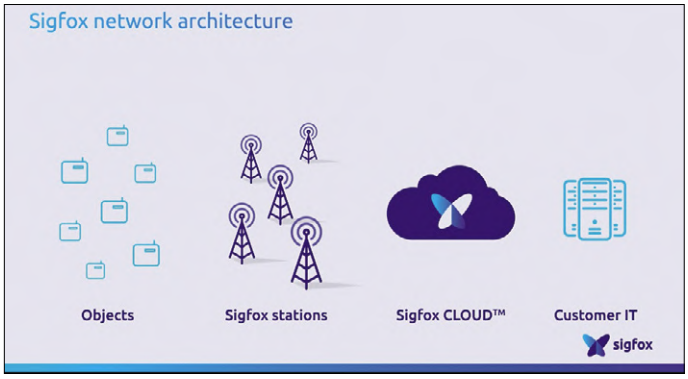


Figure 1: Basic structure of the Sigfox network (source: Sigfox).

only a maximum of **140 transmissions per device** per day are permitted due to legal regulations. So, our MKR FOX 1200 board may officially only send a telegram at 11-minute intervals. These station transmissions are then received by one to three Sigfox base stations (*Sigfox stations*) depending on coverage. All this is comparable to the way mobile radio networks work, except that the Sigfox network transmits pure measurement, status or position data and not speech, music, pictures or films. In order to cover the whole of Germany, about 1,200 Sigfox base stations are required, with coverage currently at around 85% already. Each Sigfox base station can receive, process and forward broadcasts from up to one million stations.
The base stations send all received data to the Sigfox Cloud via Internet or GSM links, from where users can retrieve their data and further evaluate it in their IT system (*Customer IT*). Within the cloud, the data is automatically forwarded to the corresponding user account in which the sending device got registered. The configuration interface of the user accounts is called Sigfox Backend. Here, the Sigfox nodes are registered, assigned to groups and the data forwarding to the customer IT (*callbacks*) is set up.
You can see that there is a lot of complex Sigfox hardware and software between Sigfox station and user. But this should not concern us here at all, because the use of a Sigfox station within the network is very easy to realize with just a few steps:

- In addition to the Sigfox modem, you need suitable Sigfox communication software for your microcontroller. In our case that's no problem at all because there is a ready Arduino library, which simply has to be linked to Sketch.
- Next, you have to register as a Sigfox user at the Sigfox backend via the Internet, which takes just a few mouse clicks. When you purchase an MKR FOX 1200 board, you also receive a subscription for one year of free operation in the Sigfox network.
- You then have to register your station with the Sigfox backend. This also only requires a few mouse clicks.
- Then you develop your Sigfox application (the Arduino Sketch) with the finished functions from the Sigfox library and send the data to the Sigfox backend.
- From the Sigfox backend, you can then 'forward' the data worldwide through the Internet very easily and process it on your computer (for example with a freeware dashboard program).

The first three steps are explained in this part of the article; the next part is devoted to the fourth point, while the last part is devoted entirely to the fifth point.
Since Sigfox is a radio-based data transmission system, you should know the Sigfox coverage at your location in advance. You can do this easily by using the Sigfox coverage map [1].

The Sigfox backend

Figure 2 shows the general structure that users need to create on the Sigfox backend for their Sigfox systems. The endpoint of any data transfer from the individual Sigfox stations is the Sigfox cloud with the backend computer in each country. From there, users can subsequently download the data onto their computers via the Internet, and have it processed accordingly. First, we explain how the data from our MKR FOX 1200 board is stored in a structured way in the backend computer.

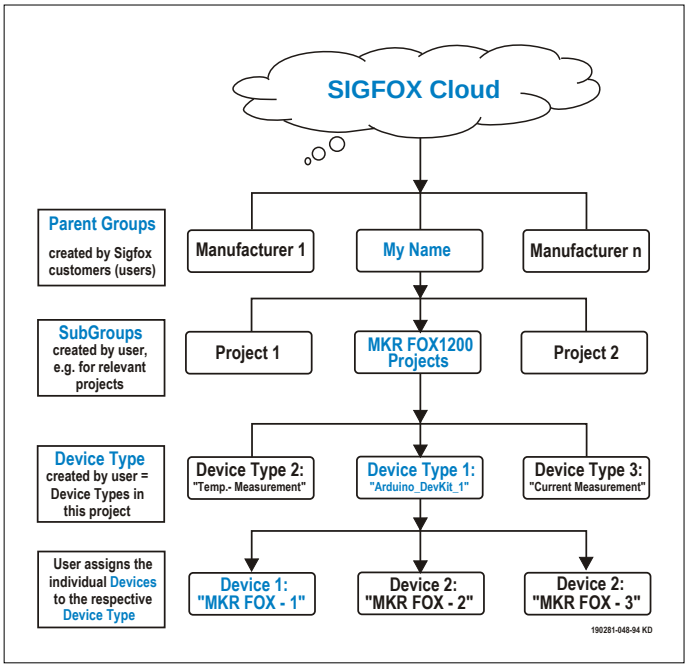


Figure 2: Basic structure on the Sigfox backend.

The first time you log on as a user in the Sigfox backend, you enter your (arbitrary) *Business name*. From this, the backend automatically generates the *Sigfox-Parent Group* with the same name, in this case *MyName*. This means that you are registered as a user in the Sigfox network. The name cannot be changed afterwards.
Now you can create so-called Subgroups within this group (below) and structure all your Sigfox applications according to individual projects. So you could create a subgroup called 'MKR FOX 1200 Projects', but this makes little sense if you only use one MKR FOX 1200 board. Therefore, we did not create any subgroups.
At the next lower level, the devices used can be combined to form a *Device Type*. A Device Type contains all devices of the same type that have the same structure, and the same functions and thus also sending the same data types and the same amounts of data. In general, these are identical devices from the same manufacturer, such as the same devices for temperature measurement, pressure measurement, flow measurement, and so on. The idea behind this is that the data sets of a device type class can be treated the same, but with different contents. In our case, it is very simple again because we have only one device of the single type, namely our MKR FOX 1200 board. So when we later register the board with the Sigfox backend, the backend automatically creates the device type 'Arduino_DevKit_1'. This name can be changed later, just as further Device Types can be created individually and deleted.

In the final step, the individual devices used (*Devices*) are assigned to a Device Type, or in concrete terms: when you register our MKR FOX 1200 board, the backend automatically assigns this board to the Device Type 'Arduino_DevKit_1' and automatically assigns the same device name *Arduino_DevKit_1_device* to it. We will change this not very original name later to: *MKR FOX - 1*. The following is very important:
All Devices of a Device Type later send the same structured

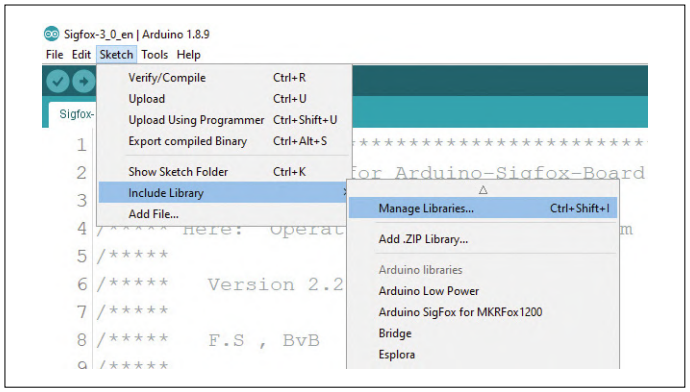


Figure 3: Calling the library manager.

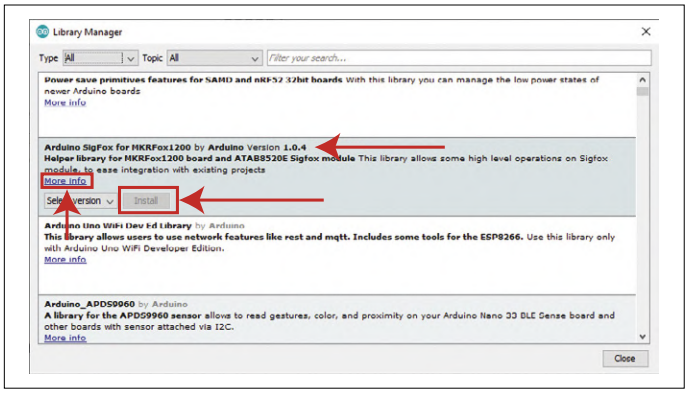


Figure 4: List of already existing original Arduino libraries.

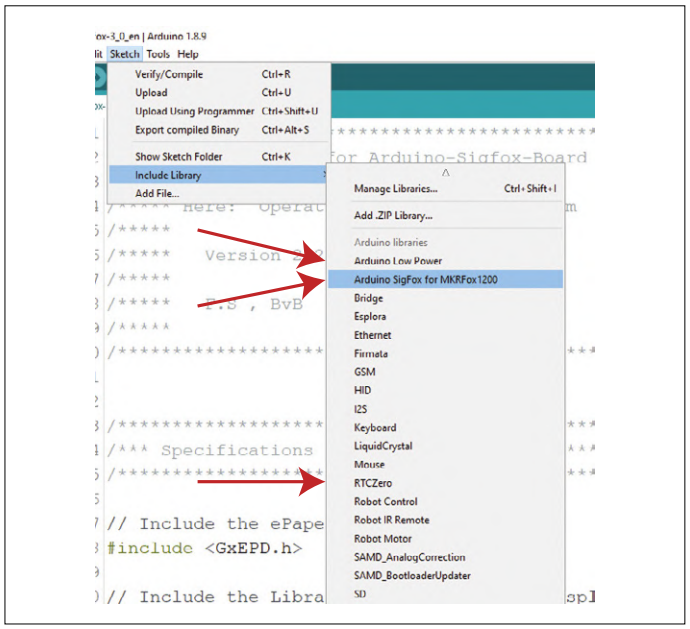


Figure 5: The three new libraries are available.

data records (*callbacks*) from the Sigfox backend to the user computer, only with individually different contents! The individual devices are then differentiated on the basis of their unique device identification number (ID).

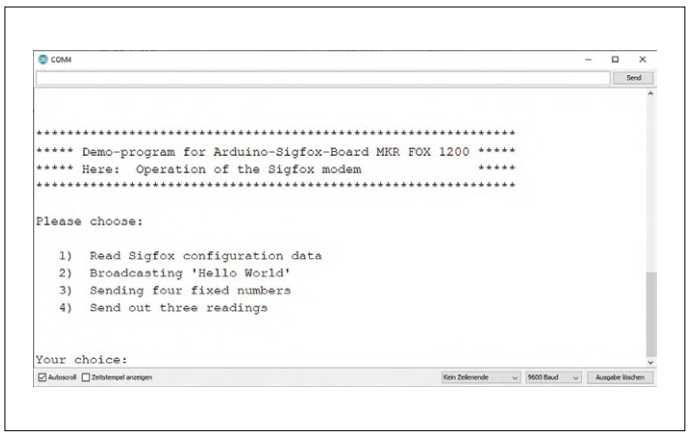


Figure 6: Main menu of the Sigfox Sketch Sigfox-3_0.ino.

Since we use only one MKR FOX1200 board, the structure tree is simply reduced to:

- Group: My Name
- Subgroup: not applicable
- Device Type: Arduino_DevKit_1
- Device Name: MKR FOX - 1
- Device ID: 47110815 (for example)

Our MKR FOX 1200 board sends its messages to the Sigfox backend. There the data is assigned internally and can then be forwarded to the user. The form in which this is done is defined in the *callbacks* (which we will discuss in the following parts of this article).

Installing the Arduino Sigfox Library

Before you can work with the MKR FOX1200 in the Sigfox network, you should install the appropriate Sigfox library *Arduino Sigfox for MKRFox1200* containing the necessary Sigfox communication routines. To do this, open the library manager in the Arduino IDE through Sketch\Include Library\Manage Library (**Figure 3**). A (very extensive) list opens with the original Arduino libraries already contained in the Arduino IDE and, depending on the application, only need to be installed (**Figure 4**). In newer versions of the IDE, this includes the three libraries we need. Now load and install the library by clicking on *Install*. The two original Arduino libraries *Arduino LowPower* and *RTCZero* are installed in the same way.
These original Arduino libraries from the IDE have the advantage of being well documented and quite extensive. If you click on the More info button (bottom left in each library field), you will get a detailed description of the functions of this library and its application. If you select *Sketch\ Embed library* again, you can see that the three new libraries are now available in the IDE (**Figure 5**).

In a new Sketch, you simply include the Sigfox library by clicking on the library name (the other two libraries are automatically included). The IDE then automatically inserts the corresponding #include statement into the Sketch:

```
#include  <SigFox.h>  //Integrating the Sigfox library
```

All functions from the three libraries become available and can be used immediately.

Table 1 shows a short description of the most important Sigfox core functions that we will use below.
In the Elektor software package [2] you will find the *Sketch Sigfox-3_0.ino*, in which all the program sequences described below are immediately executable. Load this Sketch, start it, and with it the Serial Monitor. The main menu of our Sketch appears in the Serial Monitor (**Figure 6**).

Caution: The Sigfox antenna should be connected to the module by now to prevent the module's RF output stage being damaged during (unintentional) transmission attempts! And remember: Officially, only 140 messages per day may be sent per Sigfox device!

Reading the configuration data of the Sigfox modem

Next we will only explain the actual core parts of the Sigfox operation from our well-documented Sketch. In the first step, the login (registration) of the MKR FOX 1200 in the Sigfox network — more precisely at the Sigfox backend — is necessary.

Table 1. Main Sigfox Core functions in Sigfox Library

Function	Description	Syntax	Parameter	Returns
`SigFox.begin()`	Initialises the Sigfox library and the module.	`SigFox.begin();`	none	Returns true if everything's all right, otherwise a false
`SigFox.beginPacket()`	Preparing to send a package	`SigFox.beginPacket();`	none	none
`SigFox.print()`	Sends values as strings to the Sigfox backend.	`SigFox.print(val);` `SigFox.print(str);` `SigFox.print(buf,len);`		none
`SigFox.write()`	Sends binary data to the Sigfox backend (byte or series of bytes).	`SigFox.write(val);` `SigFox.write(str);` `SigFox.write(buf,len);`	val: a numerical value as byte str: a string as a sequence of bytes buf: an array containing bytes len: Length of the buffer	none
`SigFox.endPacket()`	Terminates the transmission process that was started by SigFox.beginPacket()	`SigFox.endPacket();`	none	Returns a 0 as integer if transmission was successful and a 1 if an error occurred.
`SigFox.debug()`	This activates the debug mode. All energy saving functions are deactivated, the LED on the board is activated during data transfer.	`SigFox.debug();`	none	none
`SigFox.SigVersion()`	Returns the firmware version of the Sigfox module.	`SigFox.SigVersion();`	none	String of 2 bytes
`SigFox.ID()`	Returns the unique module ID that has been permanently stored by the manufacturer.	`SigFox.ID();`	none	String of 4
`SigFox.PAC()`	Returns the unique PAC that belongs to the ID.	`SigFox.PAC();`	none	bytes
`SigFox.end()`	De-initialises the Sigfox library and the module.	`SigFox.end();`	none	String of 16

Listing 1. Reading out essential Sigfox station parameters.

```
// Sigfox-Modem/Lib. Activation and
initialization
  if (!SigFox.begin())
  {
    Serial.println("Sigfox-Modem not found ! -
Proceed with RESET !");
    while (1);  // In this case: endless loop
  }
  else
  {
    Serial.println("Sigfox-Modem OK !\n");
  }


  //Firmware-Version, read modem ID, PAC and
Temp. and save
  String version = SigFox.SigVersion();
  String ID = SigFox.ID();
  String PAC = SigFox.PAC();
  float temp = SigFox.internalTemperature();

  // Send modem information to ser. Monitor
  . . .

  SigFox.end(); // send modem to deep slumber
```

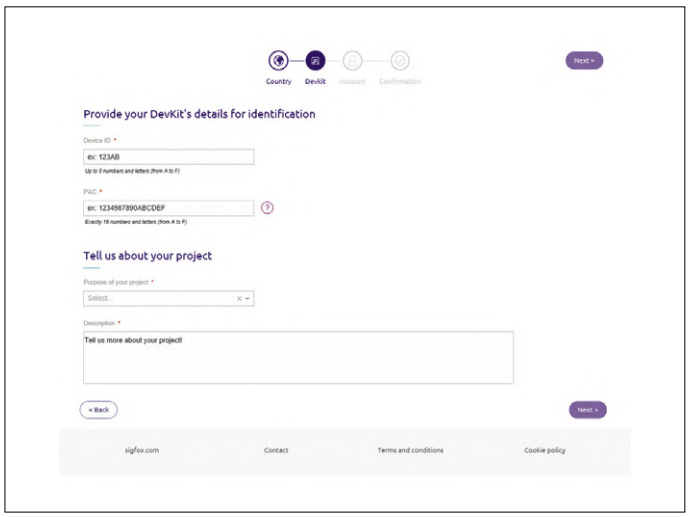


Figure 7 Registration of the MKR FOX 1200 at the Sigfox backend (I).

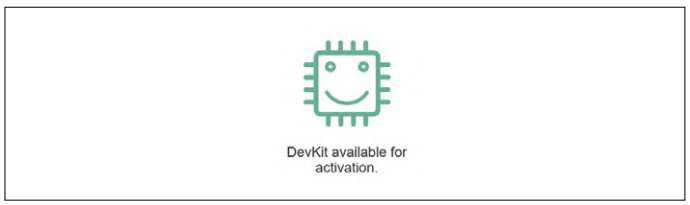


Figure 8. ID and PAC are OK, and we continue ...

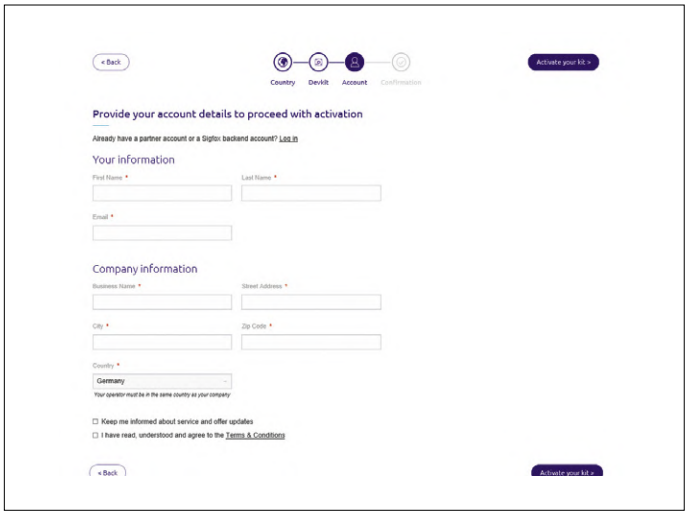


Figure 9: Registration of the MKR FOX 1200 at the Sigfox backend (II).

This requires the ID and PAC number of the Sigfox modem. These two bits of information are unique and clear for each Sigfox unit (permanently stored in the Sigfox modem) and can be easily read out.
Now download our demo Sketch and select menu item 1, the core function of which is elucidated by **Listing 1**. First the Sigfox modem and the Sigfox library are activated and initialised with Sigfox.begin(). Any errors that might occur are displayed only. The required information is then read with the appropriate Sigfox functions, stored in variables and finally output on the serial monitor with appropriate Serial.println statements. Reading the firmware version and the internal modem temperature is interesting but not mandatory. With the last function call `SigFox.end();` the Sigfox library gets uninstalled, the Sigfox modem deactivated and switched to power-saving mode. The ID and PAC numbers determined and noted in this way are (examples):
ID = 00123456 (always 8-digits)
PAC = 1234567890abcdef (always 16-digits)

Registration of the MKR FOX1200 board at the Sigfox backend

Now our MKR FOX 1200 board should be registered at the Sigfox backend under [3]. On the start page, complete the necessary fields as shown in **Figure 7** and click on the Next button at the bottom right. The ID and PAC data will be checked. If everything is ok, you'll get some friendly feedback (**Figure 8**). Then click on the Next button again and go to the second activation window (**Figure 9**). Again, fill in the fields accordingly and click on the button Activate your kit in the lower right corner. If everything has been completed correctly, you will receive confirmation of your registration (**Figure 10**).

As you can see here, the activation of the MKR FOX 1200 board will not be completed until the board has sent its first message to the Sigfox backend (which will be dealt with in the next instalment). Only then is the device recognized and activated in the Sigfox network, and you can upload your 140 messages of 12 bytes ar day for one year free of charge (four



Figure 10: Registration of the MKR FOX 1200 at the Sigfox backend (III).



Figure 11. Confirmation e-mail.

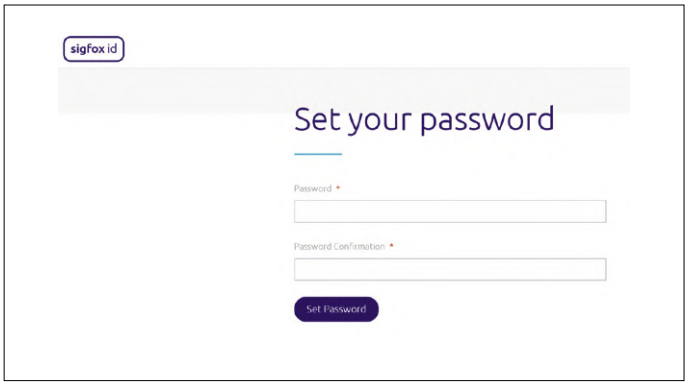


Figure 12: Assigning the password.

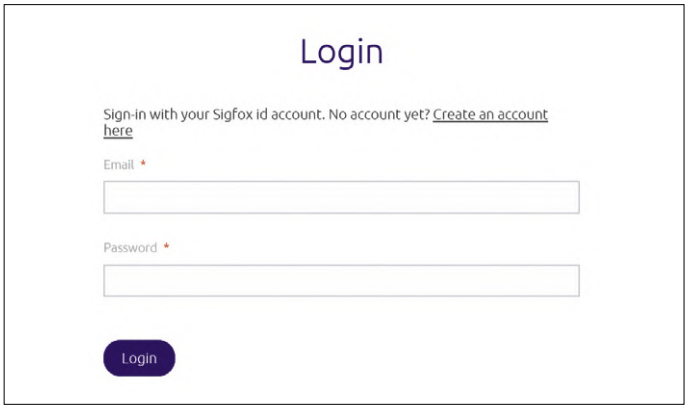


Figure 13: Entering the login data.cf

messages of 8 bytes per day can be received as a download, for example for remote configuration).
Before closing this page, you should check out the Sigfox tutorial videos on this page. You can do this later too, because you can also find these videos on YouTube [4][5][6]. By now you should also have received a confirmation via the e-mail address provided (**Figure 11**). You should use this mail to assign a password to log in to the Sigfox backend. Click on the SET YOUR SIGFOX ID PASSWORD field so that the input mask appears in **Figure 12.**
Fill in the fields as usual and enter your login data in the next window (**Figure 13**). After clicking on Login the created profile will be displayed. The Sigfox backend login procedure is complete! Now log out at the top right of the window via Profiles & Settings and practise patience. In the next part, we will program the Arduino and make the first steps in the Sigfox network!

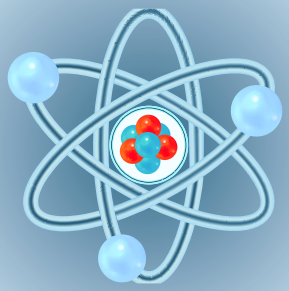
190281-B-03

@ WWW.ELEKTOR.COM
→ **Arduino MKR FOX 1200:** www.elektor.com/19096
→ **Arduino 868-MHz Antenna:** www.elektor.com/19095

Web Links

[1] Sigfox coverage: www.sigfox.com/en/coverage
[2] Project page: www.elektormagazine.com/190281-B-03
[3] Sigfox Login: https://buy.sigfox.com/activate/devkit/DE
[4] Sigfox introductory video: www.youtube.com/watch?v=6ZBGDtmDGRU
[5] Sigfox introductory video: www.youtube.com/watch?v=7gTwFbiiJwE
[6] Sigfox introductory video: www.youtube.com/watch?v=dDNY-xAxECE





Starting Out in Electronics

Easier than imagined!

By **Eric Bogers** (Elektor Netherlands)

Electronics is a fascinating hobby but to have a reasonable chance of success, some elementary knowledge is indispensable, otherwise your soldering iron will be covered in dust soon. Of course, you can take potluck by buying a DIY kit online and then a solder iron picked from the Daily Discounts Bin at the local hardware store – and then pray hard that the joint Gods & Saints in Electronics Nirvana will bless & protect you. But just in case something does go wrong…

… it's a good thing that you bought, borrowed, stole or "otherwise obtained" this very edition of *ElektorLabs Magazine*, because from now on every edition aims to pay attention to the principles of electronics, permitting you to build up a solid foundation for your pastime and education. And for the 'old hands' at the trade: if you think you already know everything, you can skip these pages with no loss of sleep.

Definitions

First of all, we should propose and agree on definitions of some elements of applied physics as far as electrical engineering is concerned. For that, short introductions are essential as a foundation on which your knowledge of electronics can be built and expanded.

Elementary charge

Both in electronics and in electrical engineering, everything revolves around electrons — in fact that's where the names of the two disciplines originate from. All chemical elements consist of atoms, which in turn consist of a nucleus (in which almost all mass is collected) and orbiting electrons, the number of which determines the chemical properties and number of the element in question.
Electrons have an indivisible negative charge called the elementary charge *e*. It can be written as:

$$e = 1.6021892 \times 10^{-19} \text{ [C]}$$

Charge

The C in the above definition stands for the coulomb, the unit of electrical charge, hence: [C]. Just for the record: a **quantity** is something that can be measured (in any way) — for example length or temperature. A **unit** expresses the value of that measured **quantity** — in the case of length, for example, metres written as [m].

Since this elementary charge has an unmanageable curve and an extremely small value, the coulomb 'C' has been defined as the unit of quantity charge Q, namely:

$$1 \text{ C} = 6.2414601 \times 10^{18}\, e$$

This value originates from a time when the elementary charge could not yet be measured. At that time, it was agreed fairly randomly that one coulomb equals one ampère-second: if a current of one ampère flows for one second, then a charge of one coulomb got conveyed through a wire.
By the way: a unit is written in full in lower-case letters but abbreviated with a single capital letter. So it's 'one coulomb' or '1 C'.

Current

The official definition of electric current (unit: ampère, abbreviated as A) is a relic from the past that we will not bore you with. Here we derive the electrical current from the electrical charge:

$$1 \text{ A} = 1 \text{ C s}^{-1}$$

In words: "one ampère is a current amounting to 6.2414601 times 10^{18} electrons per second."

Voltage

Okay, so there's a current flowing in a wire or another medium — but why? Compare this with water: water flows thanks to propulsion: a pressure difference (pump) or a height difference. In the case of an electric current, the propulsion is a charge difference between two points, and that difference is called voltage. The unit of voltage is the volt (V). Note: a voltage is always measured between two points.
Voltage is defined by a detour: when a current of one ampère

runs in a metal conductor and a power of one watt gets converted into heat in that conductor, there is a voltage of one volt across the extremes of that conductor. That's represented in a formula as:

$1\ W = 1\ V \times 1\ A$

Power

And that brings us to the final definition, which is that of electrical power. Here, we have to distinguish between two concepts that are often confused: power and 'labour'. For example, in order to boil an egg, a certain amount of "labour" has to be provided over a certain amount of time to bring the water to the boiling point and keep it there. That is the labour — it is therefore measured throughout the entire period under discussion (e.g. 8 minutes). The power is the same labour per unit of time, for example per second.
When we talk about electricity and electrons, power is defined as the product of current and voltage. The unit of electrical power is the watt (W).

Symbols

Of course, it is not particularly helpful if we need to describe in words what current, what voltage or what power is being discussed exactly ("... the current through the resistor numbered 10 ..."). That's why we use symbols. Like: I_{R10}. Charge is the capital letter *Q*, current is represented by the capital letter *I*, for voltage we use the capital letters *V* or *U* and for power, the capital letter *P*, all italicized. Summarized in a small table:

Quantity	Unit	Symbol
Charge	coulomb (C)	*Q*
Current	ampère (A)	*I*
Voltage	volts (V)	*V* or *U*
Power	watts (W)	*P*

So much for the dry theory. In the next edition, things will get interesting as we start to calculate. ⏮

(191166-03)

The magazine article series "Starting Out in Electronics" is based on the book *Electronics Basic Course* by Michael Ebner, published by Elektor.

@ WWW.ELEKTOR.COM

→ Book: Basic Electronics Course
www.elektor.com/13950

→ E-book: Basic Electronics Course
www.elektor.com/18232



Analogue Electronics Design

Case Study #1 — Section 1: MEMS Microphone… Testing 1-2-3

By **Ton Giesberts** (Elektor Labs)

In this new article series, experts in the field explore aspects of analogue electronics design that should benefit an increasingly 'digital-only' audience, and underpin that analogue is not black magic. We kick off at rapid pace with an oddball part, the MEMS microphone. Extremely small and little used in DIY projects so far, it's nonetheless a key part in the successful Bat Detector-Plus project published in Elektor's September & October 2016 edition. The use of a MEMS microphone is not "drop-in-and-forget" but has ramifications both for the application range and the amplifier that follows it. Here are some details, compliments of Ye Olde Analogue Design Department.

MEMS is an acronym for Micro ElectroMechanical Systems, meaning a mechanical part is placed close to or on a semiconductor chip by using additional substrates and high-quality micro-machining. MEMS are SMDs and in the production process can be fitted using pick & place machines, along with other parts. A multitude of applications are possible. MEMS components are used in almost all market segments including consumer electronics, medical, communication, automotive, etc. Applications are found in image stabilization for cameras, hard disk security, pacemakers, insulin pumps, medical pressure transducers, mobile phones, airbags and much more. It's impossible to imagine today's world without MEMS.

The MEMS microphone

Perhaps the most important part of the Bat Detector^{PLUS} project [1] is the microphone that's used to record the mainly ultrasonic sounds of bats. Although some species 'operate' below 20 kHz but above 10 kHz, most produce ultrasonic sound. As with an electret microphone, a membrane, which is the actual sensor for sounds, causes minimal differences in capacitance that can be translated into a change in voltage. Most electret microphones house a JFET acting as a buffer. By contrast, in a MEMS microphone the diaphragm is mounted right next to a semiconductor chip (ASIC). This also permits translating the analogue voltage into a digital signal — usually 'PDM'.

As shown in **Figure 1** there are different types of MEMS microphone, for example with the acoustic port on top a simple hole in the metal part, or the with the acoustic port between the pads on the bottom, requiring a hole in the PCB. MEMS microphones are generally quite insensitive to supply voltage noise, shocks, vibrations, humidity and condensation.

Of course, the use of a MEMS microphone on a PCB also has disadvantages. To avoid damaging the microphone, after the reflow process, do not wash or brush the PCB with or without a solvent. For other instructions always follow the datasheet and design guides furnished by the manufacturer.

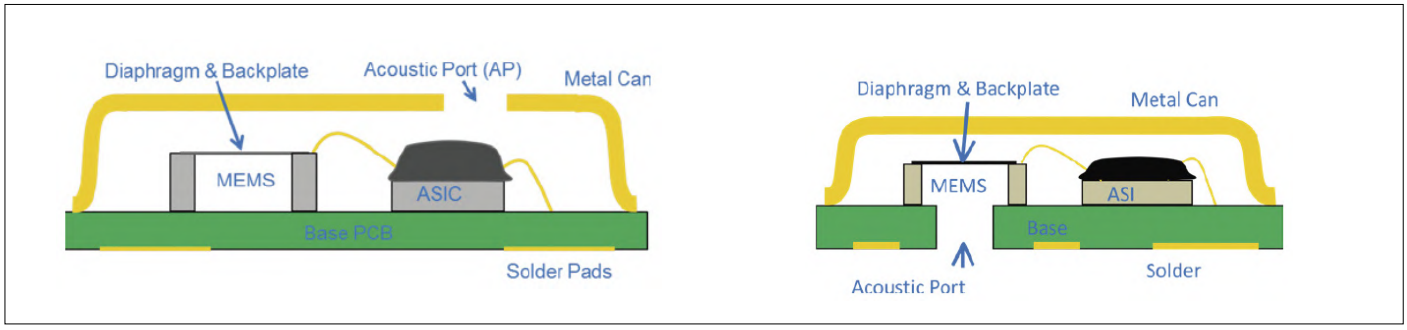


Figure.1: Cross sections of a 'top-port' (left) and 'bottom-port' (right) MEMS microphone. Source: Application Note 'sisonic-design-guide.pdf' by Knowles [2].

Because the membrane in the MEMS microphone is very small, the bandwidth is high. The sensitivity will be a few dB higher above 10 kHz roughly. MEMS microphones are not only suitable for speech but also for burglar alarms, motion detection, etc. The manufacturer of the device used in the Bat Detector[PLUS] offers device series specified up to a whopping 80 kHz. Where not specified, it can be assumed that the sensitivity in the ultrasonic range, and their bandwidth, also makes these types usable for our purpose. Still, Figure 1 shows that in the first (left-hand) type the hole is not above the diaphragm and that will affect the frequency response. This suggests that the second (right-hand) type would be more suitable. The frequency response of the SPH0641LU4H-1 closely resembles the upper curve in **Figure 2**, and puts the device in the 'second' class.

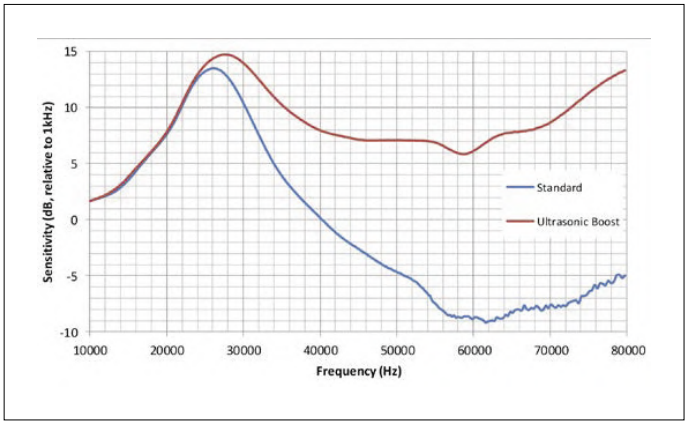


Figure 2: Frequency characteristics of MEMS microphones. Bottom-port and MEMS-on-Lid top-port SiSonic microphones from Knowles are naturally very useful in the ultrasonic range of 20 kHz to 80 kHz. Selected types have additional gain in that range. Source: Application Note 'sisonic-design-guide.pdf' from Knowles [2].

The microphone board

Aiming to make the Bat Detector[PLUS] easy to replicate at home, at least as far as the MEMS microphone is concerned, a small PCB was designed for the "circuit" (**Figure 3**; ha-ha) and made available separately with the MEMS-microphone already assembled [3]. It is universally usable. In case the supply voltage is ridden with noise or interference, or the PCB is connected through a long 3-wire cable, a decoupling capacitor at the bottom of the PCB can help — best of all, a 100-nF 1206 SMD type soldered at the bottom of the PCB. 1206 is quite large and therefore still easy to solder by hand. In the Bat Detector[PLUS] the supply voltage is separately stabilized with a 3-V Zener diode and because of the short distance (PCB mounted in a 3.5-mm audio jack plug!), further decoupling is not necessary. If a decoupling capacitor is installed, a type with C0G/NP0 dielectric is recommended. This material suffers the least from microphony, i.e. converting movement due to vibration or shock into an electrical voltage. This is a problem with ceramic capacitors, and they are also the worst capacitors to use as coupling or filter devices in audio amplifiers as their capacitance depends on the applied voltage. The C0G/NP0 dielectric has the least problems with this.

As with BGA enclosures for SMD ICs, the connections of the MEMS microphone are on the underside of device and it is best to use a reflow oven to solder it on the PCB. The MEMS microphone chosen for the project is a top-port type. For the second type (i.e. bottom-port) a hole is needed in the PCB surface under the microphone. This affects the frequency response due to the size of the hole and the thickness of the PCB. But not only the PCB has influence. With the top-port device also, the PCB housing is a factor to consider. Through reflections, even surfaces in the immediate vicinity of the microphone affect the final frequency response. In order to connect the opening in the housing to the acoustic port of the microphone, a gasket is used that must fit properly to prevent leaks (i.e. unwanted air gaps).

Adaptations to the MEMS microphone analogue amplifier

Before designing a microphone amplifier, it is important to know what the amplitude of the input signal will be, as well as the signal level applied to the next stage, no matter if a MEMS-, electret- or even a dynamic microphone is applied. The actual microphone doesn't make much difference in principle, except for correct matching and correct supply voltage. Often an input impedance is specified by the manufacturer (and certainly the pull-up needed for an electret). If not, there is always a 'typical application schematic', or the internal JFET's drain current is specified so that even a higher supply voltage is possible for the electret microphone if the resistance is adjusted accordingly. Be prudent though with the maximum voltage for the JFET.

The output impedance of an electret microphone will be higher than that of a MEMS one due to the pull-up and may therefore produce more noise. With an electret microphone, it's the JFET biasing resistor from the supply to the output that determines the output resistance. The typically found value is 2.2 kΩ. To prevent excessive voltage division at the input, the input resistance must be 5 to 10 times higher. With a non-inverting amplifier this resistance (which it can be much higher at the plus-input) has much less consequences as far as noise generation goes. The resistors in the feedback network do have serious effects but can be chosen considerably lower because they don't put a load on the microphone. Only when the roll-off frequency at the input is relatively high, a higher input impedance is created around that frequency. Else, that

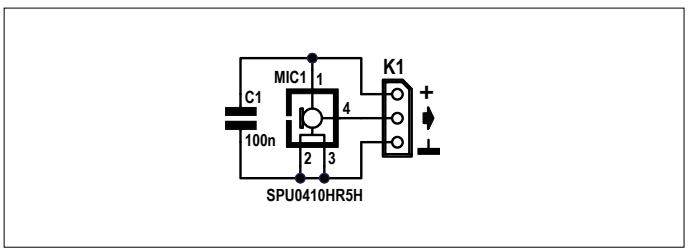


Figure 3: Bat Detector[PLUS] microphone board schematic.

Web Links

[1] Bat Dectector-PLUS article: www.elektormagazine.com/magazine/elektor-201607/29128

[2] Knowles' Sisonic Design Guide: www.knowles.com/subdepartment/application-notes/dpt-microphones/subdpt-sisonic-surface-mount-mems

[3] Elektor MEMS microphone board: www.elektor.com/bat-detector-with-amplitude-recovery-mems

Surf & Learn

For further reading (web links not referenced in article)

1. Analog Devices: www.analog.com/en/analog-dialogue/articles/mems-microphones-future-for-hearing-aids.html
2. Digikey: www.digikey.de/en/articles/techzone/2019/feb/mems-vs-ecm-comparing-microphone-technologies
3. MEMS intro: www.allaboutcircuits.com/technical-articles/improving-on-the-electret-an-introduction-to-mems-microphones/
4. Bat detectors: www.skillbank.co.uk/bat_detectors/afd1.html
5. TDK InvenSense ICs: www.mouser.co.uk/new/tdk/tdk-invensense-ics-40740/ (read the additional resources)

impedance it is only determined by the output resistance of the microphone.
If we take an inverting amplifier, a too high resistance means extra input noise, then the choice for the input resistance must be a compromise, because it is part of the feedback. Take, for example, R4 in the Bat Detector amplifier, it's 7.5 kΩ. The noise of a resistor increases with the square root of the increase. So, 3 dB more noise for a twice higher (6 dB) resistance value.
If a microphone is intended for a concert or speech, the output signal is within certain limits. But if the signal is ultrasonic, the characteristic of the microphone is not specified, and the distance to the sound source varies as with a flying bat, things become a bit more difficult. Over a certain distance, higher frequencies are subject to higher attenuation in air than lower frequencies, and the effect of varying distances is even greater with ultrasound. As the graph in Figure 2 showed, the characteristics are anything but straight, especially the 'standard' characteristic. First, you can determine the amplification you want. In the case of the Bat Detector that depends on what the comparator (IC3B) needs to work properly, and the signal level needed for rectifier IC4A to do the modulation recovery. Probably this amplification is also a value that needs to be determined empirically or adapted to match the microphone used. This makes it difficult if different microphones are used and the amplifier has a fixed setting. Furthermore, it is necessary to limit the frequency range of an amplifier to that for which it is intended. For bats, a bandwidth of 10 kHz to 120 kHz is generally mentioned — a frequency range for which the microphone amplifier must be tailored.

Next time we continue with an evaluation of the original microphone preamplifier used in the Bat Detector^{PLUS} and seek solutions to optimizing it for use with the proposed MEMS microphone. ⏮

191143-01

Step Forward



If you are an expert in analogue electronics design with authoring skills, step forward and report to series editor Jan Buiting, email jan.buiting@elektor.com, with your resume.

2ew20P
Your e-code for free admission
embedded-world.de/voucher

Nuremberg, Germany
25 – 27.2.2020

embeddedworld

Exhibition&Conference

... it´s a smarter world

DISCOVER INNOVATIONS

Over 1,000 companies and more than 30,000 visitors from 84 countries – this is where the embedded community comes together.

Don't miss out! Get your free ticket today!

Your e-code for free admission: 2ew20P

embedded-world.de/voucher

@embedded_world #ew20 #futurestartshere

Exhibition organizer
NürnbergMesse GmbH
T +49 9 11 86 06-49 12
visitorservice@nuernbergmesse.de

NÜRNBERG MESSE

Small Circuits Revival

Capita Selecta from the Elektor Project Suggestions Box

By **Eric Bogers** (Elektor Netherlands)

ElektorLabs editors are inundated with reader e-mails expressing praise as well as criticism. One of those criticisms in recent years was that, after the demise of the Summer Circuits, there weren't enough 'small' circuits, tips and tricks that are especially suitable for the novice electronics hobbyist. Good news: as of this now we are going to do something about it!

Energy-efficient Relay Circuits

From an idea by **Michael A. Shustov** (Russia)
and **Andrey M. Shustov** (Germany)

The relay circuits discussed here can be connected in series with a load (lamp) and a series of normally-closed (NC) pushbuttons. When one of these switches is pressed, the lamp lights up for a certain time. After that time, the lamp will turn off automatically [1].
This looks a bit like the familiar stairwell lamp or 'multi-switch' circuit [2], where you can switch on the lighting in a stairwell or corridor with the push of a button and switch it off again with the push of another button. However, the circuits described here automatically switch themselves off again - and are in principle a lot more energy-friendly.

Version 1

The first variant (**Figure 1**) is not difficult to fathom. The circuit runs off a 12 VDC source (battery, mains adapter) — and can therefore also be safely reconstructed by a beginner. Immediately after the circuit is connected, capacitor C1 will be charged by lamp LA1 and diode D1 via the series connected normal-closed pushbuttons S1—Sn (only three have been drawn). The lamp will light up for a while. LED1 indicates that the circuit is armed. For the record: the current flowing through the lamp at rest is too low to let it light up.

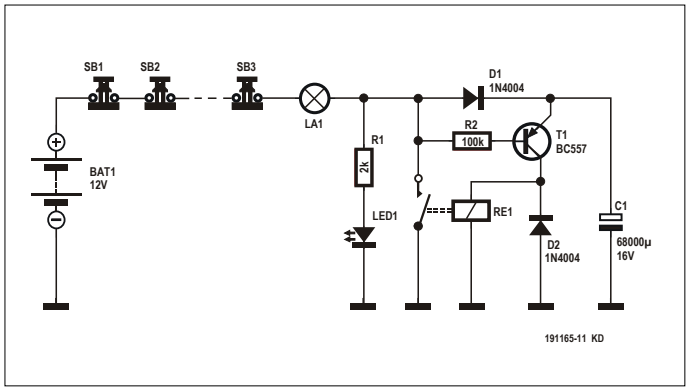


Figure 1: The simplest version contains an electromechanical relay.

As soon as one of the pushbuttons is operated, no more current can flow into C1 through D1. The base of transistor T1, which via the lamp is held at + potential of the power supply, is now pulled to ground via resistor R1 and LED1. The result: the transistor switches on and capacitor C1 discharges to ground via the relay coil. The relay is energized and via the relay contact there is now enough current to keep the lamp on. Also, the charging current into C1 is interrupted.
After some time, the capacitor has discharged so much that the relay can no longer remain activated. The relay contact opens, and the initial situation is restored (whereby we quietly assume that the pushbutton is not held pressed).
For the time that the lamp remains switched on, the following approximate relationship applies:

$t = 0.67 \times R_{relay} \times C1$ [seconds, ohm, farad]

The EMR type mentioned in the diagram has a coil resistance of 1050 Ω; for the G6DS type it is 1200 Ω. With the former relay and a capacitor of 68,000 µF we arrive at about 40 s.
The value of resistor R1 must be chosen so that the current through LED1 is just large enough to make it light up weakly. The value mentioned in the diagram will usually suffice, but there is room for some experimentation.

A word of warning: this circuit should **NOT** be used for lamps that are connected to the AC power line (mains)!

Version 2

In this version (**Figure 2**), the electromechanical relay got replaced by a MOSFET type 2N7075 or 2N7085 as the switching element. Important: in contrast with the previous incarnation, the 'lights-on' time can be conveniently set using preset or potentiometer R3. Roughly speaking, 1 kΩ corresponds to 1 second, so with the value of 50 kΩ shown in the diagram, an adjustment range of 1 to 50 seconds is achieved.
The big advantage of the relayless version is that a much smaller capacitor can be used in position C1. This improves the repeatability of the circuit. The maximum current that can be

switched (with sufficient cooling) is about 30 A for the 2N7075 and 20 A for the 2N7085.
Again, this version is only suitable for low-voltage direct-current applications and should never be used for lamps connected to the AC line voltage.

Version 3

In practice you'll wish to switch lamps and lighting fixtures that are normally connected to the AC line voltage (50 or 60 Hz). It's not just the higher AC voltage that makes it necessary to completely rethink the design – the circuit being connected to the AC power grid also requires careful selection of the components and an even more careful assembly!
AC voltages cannot be switched using a 'normal' semiconductor switch (transistor or MOSFET); for this we require either an electromechanical relay or a special component that is called a triac. The very abbreviated summary: we can imagine a triac as two thyristors connected in anti-parallel [4]. And a thyristor in turn can be seen as a diode switched to (unidirectional) conduction at its gate by means of a firing voltage, and then keeps conducting until the current through it drops to zero.
The general effect of this version (**Figure 3**) is not difficult to understand. In the idle state with none of the series connected, push to break switches SB1-SBn pushed, a small current flows through the lamp, capacitor C1, resistor R2, diode D1 and resistor R1, causing LED1 to light and holding off transistor T1. Via D3, capacitor C3 is fully charged to a little under the Zener voltage set by D2 (i.e. about 13 V minus the forward voltage drops of D1 and D3).
As soon as one of the switches is pushed briefly (!) and then released, pnp transistor T1 starts to conduct (its base is pulled 'low') and capacitor C3 gets discharged via T1 and R6 into the gate (control input) of the triac.
With the indicated component value for C3 (10,000 µF), the triac will continue to conduct for about 27 seconds (a little less than 3 seconds per 1000 µF). Once the gate current has diminished to a level that is too small to keep the triac in conduction, the lamp will go out and after a few seconds, during which C3 is charged again, the idle state is restored.
The triac is not used optimally in this circuit. Normally, a short pulse is used to ignite a triac after each zero-crossing of the mains voltage. The triac will 'extinguish' at the next zero-crossing and will be fired again a little later. By varying the time between the zero-crossing and the firing, the brightness of the lamp can be adjusted — this is how a dimmer works. Here, the gate of the triac is continuously controlled for the entire 'on' time of the lamp (27 seconds with C3 = 10,000 µF); this of course increases the dissipation in the triac.

Construction

The low voltage variants can be built up on a piece of perfboard board without any problems as there are no dangerous voltages anywhere. The third version differs radically in that respect, requiring a 'real' printed circuit board with at least 6 mm PCB track distance guaranteed. However, assembly on a printed circuit board is feasible by removing any unused solder pads between the soldered connections, so that a safe distance is maintained. And please note: with the third version, all components are at live AC grid potential so that touch-proof installation is an absolute necessity.

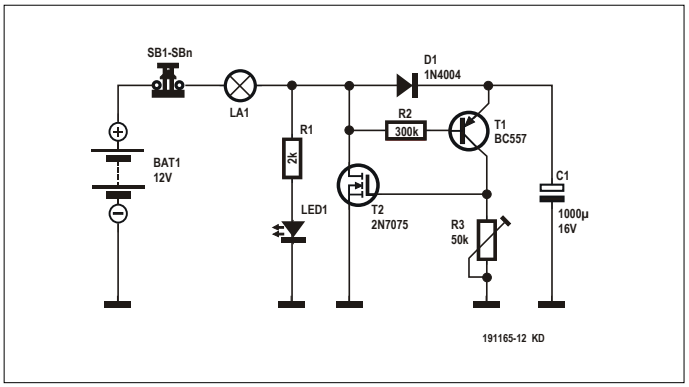


Figure 2: The 'all transistor' DC variant.

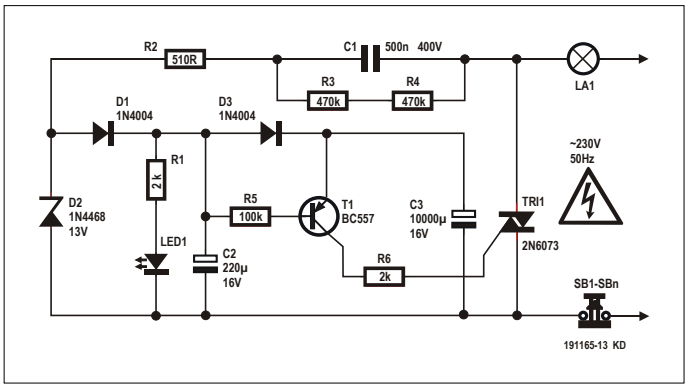


Figure 3: The AC mains incarnation. Attention: this circuit must be installed such that no part of it can be touched!

If you don't want a LED to indicate that the circuit is on, you can simply omit it in all three variants with total peace of mind. Resistor R1 should remain in the circuit, however, and the LED becomes a wire bridge in the circuit diagram. Without the LED, R1 should take a value of 2 to 20 kΩ. ⏮

(190165-03)

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

Links and literature

[1] Multi-switch Lights Control: Elektor June 2014: www.elektormagazine.com/magazine/elektor-201406/26522
[2] Multiway switching: https://en.wikipedia.org/wiki/Multiway_switching
[3] The Triac: https://en.wikipedia.org/wiki/TRIAC
[4] The Thyristor: https://en.wikipedia.org/wiki/Thyristor

In a Nutshell: Text for Microcontrollers

Using compression to save memory

By **Thomas Beck** (Germany)



Many microcontroller projects output menus and help text, error messages or diagnostic messages via a graphic user interface or diagnostic interface. However, a large volume of text requires a large amount of memory space. With suitable compression, it is nevertheless possible to store enough text in microcontroller memory to make external memory unnecessary in many cases. This article presents a possible solution using dictionary compression.

All effective compression methods are based on the redundancy of the data to be compressed. In text files, redundancy results from characters that occur more often than other characters or from character strings that occur repeatedly. A form of entropy coding, such as Huffman coding [4], can therefore be used to code letters that occur relatively often in the text with shorter bit sequences than letters that occur less often. Another option is to use a substitution method such as the LZ77 algorithm [5], which is also known as dictionary compression, to store recurrent character strings in a dictionary. In the compressed text, the character string is replaced by a reference to the dictionary entry. An attractive feature of the LZ77 algorithm is that it is not even necessary to store a separate dictionary in the compressed data. A portion of the decompressed data forms a dictionary that is gradually filled out in the course of further decompression, and when the maximum dictionary size is reached, it is shifted using a sliding window method. Compression programs such as 7z or Zip use a combination of the two methods — dictionary compression followed by entropy coding — to achieve better compression. However, the potential savings with both methods are relatively small for short texts. Furthermore, it is not possible to extract individual sentences from a compressed text without first decompressing the entire text.

Analysis of the text to be compressed

If you examine the text strings of the specific application in **Listing 1**, you can see that dictionary compression looks promising because it takes advantage of redundant words and word sequences Space characters occur frequently in every text because they are used to separate words. For space characters, either the compression routine creates its own dictionary entry or the decompression routine inserts space characters between successive words automatically.

The latter option only works smoothly when there is no special treatment for special characters located before or after words without an intermediate space character — for example, with *(Gauge)* being a dictionary entry instead of being represented by the three entries *(*, *Gauge* and *)*.

- The entire text consists of one or more C string arrays ➞ individual text strings can be specifically accessed via array indices — which is one of the requirements for the compressed text.
- The text contains many frequently occurring words or word sequences ➞ the redundancy necessary for compression is present.
- The text is in English ➞ the text uses only the 7-bit ASCII codes (0–127). The codes greater than 127 are free. This simplifies step 5 of the proposed method described below.
- Most words in the text start with an upper-case letter ➞ *Engine* and *engine* would be two separate entries in the dictionary. Compression can be improved if all words start with upper-case letters. For good comprehension, articles and preposition should be written with lower-case letters.
- The text contains relatively few special characters, such as *( ) # / - ,* ➞ dictionary generation produces only a few entries that differ from other potential entries, such as *(Gauge)* and *Gauge*.
- The longest text string in the overall text is 57 characters (including the end-of-string character) ➞ the decompression routine needs a RAM buffer of at least 57 bytes.

Listing 1. Excerpt of the text to be compressed as a C string array.

```
const char * const text[] = {
  /*   0 */ "Fuel System 1 Status",
  /*   1 */ "Fuel System 2 Status",
  /*   2 */ "Calculated Load Value",
  /*   3 */ "Engine Coolant Temperature",
  /*   4 */ "Short Term Fuel Trim Bank 1",
  /*   5 */ "Short Term Fuel Trim Bank 3",
  /*   6 */ "Long Term Fuel Trim Bank 1",
  /*   7 */ "Long Term Fuel Trim Bank 3",
  /*   8 */ "Short Term Fuel Trim Bank 2",
  /*   9 */ "Short Term Fuel Trim Bank 4",
  /*  10 */ "Long Term Fuel Trim Bank 2",
  /*  11 */ "Long Term Fuel Trim Bank 4",
  /*  12 */ "Fuel Pressure (Gauge)",
  ...
  /*  46 */ "OBD Requirements to Which Vehicle or
   Engine is Certified", // len=57
  ...
  /* 149 */ "Commanded Boost Pressure A",
  /* 150 */ "Boost Pressure Sensor A",
  /* 151 */ "Commanded Boost Pressure B",
  /* 152 */ "Boost Pressure Sensor B",
  /* 153 */ "Boost Pressure A Control Status",
  /* 154 */ "Boost Pressure B Control Status",
  ...
  /* 177 */ "Manifold Surface Temperature",
  /* 178 */ "Intake Manifold Absolute Pressure A",
  /* 179 */ "Intake Manifold Absolute Pressure B",
  ...
  /* 188 */ "Exhaust Gas Temperature Bank 2 -
   Sensor 7",
  /* 189 */ "Exhaust Gas Temperature Bank 2 -
   Sensor 8"
};
```

How it all started

The incentive for the development of a lossless text compression method came from my Elektor OBD2 vehicle diagnosis projects [1][2][3]. Like professional diagnostic testers, they needed to provide the users with several thousand descriptions for diagnostic trouble codes (DTCs). The total volume of these descriptions is approximately 211 KB, which can be reduced to around 35 KB with the compression method presented here. As a result, the text fits in the AT90CAN128 8-bit AVR microcontroller used in the project [1], which has 128 KB of flash memory.
In the Arduino project [3], the small Arduino Uno R3 and the Elektor Uno R4 with just 32 KB of flash memory should also be supported. Of course, this memory space is not large enough for the compressed trouble descriptions, but the descriptions for the OBD2 parameter identifiers (mostly descriptions of sensor data) previously supported in the projects, occupying approximately 7 KB, could be compressed to around 2 KB. The performance of the compression method I developed can be illustrated using the current 190 short PID descriptions in Listing 1 as an example.
The following values for the memory space requirements of the decompression routine have been determined for an AT90CAN128 8-bit AVR microcontroller with the AVR GCC compiler version 4.9.2 (with optimisation Os enabled):

	Original text [bytes]	Compressed text [bytes]	Decompression routine [bytes]	RAM buffer for decompression [bytes]
DTC description	216,311	35,340	566	106
PID description	7,273	1,820	314	57

Dictionary compression

Step 1: Creating the dictionary

This step breaks down the text to be compressed into its individual words. The separator here is the space character, which will later be reinserted y the decompression routine. The frequency and memory space requirement of each word (string length plus end-of-string character) are also calculated. At the end, the created dictionary is sorted in order of decreasing frequency. In case of the same frequency, the longer length takes precedence. **Listing 2** shows the dictionary after the optimisation in Step 2. The frequency and length are noted there as comments .

Listing 2. Excerpt of the optimised dictionary as a C string array.

```
const char * const dictionary[] = {
   /*   0 */ "Sensor", // cnt=116 len=7
   /*   1 */ "Bank", // cnt=104 len=5
   /*   2 */ "-", // cnt=89 len=2
   /*   3 */ "2", // cnt=67 len=2
   /*   4 */ "1", // cnt=66 len=2
   /*   5 */ "Fuel", // cnt=45 len=5
   /*   6 */ "Temperature", // cnt=43 len=12
   ...
   /*  16 */ "Exhaust Gas", // cnt=16 len=12
   ...
   /*  30 */ "Status", // cnt=9 len=7
   ...
   /*  45 */ "System", // cnt=4 len=7
   ...
   /*  77 */ "8", // cnt=2 len=2
   /*  78 */ "Currently Being Utilized by the",
   // cnt=1 len=32
   /*  79 */ "Advance for #1 Cylinder",
   // cnt=1 len=24
   ...
   /* 125 */ "F", // cnt=1 len=2
   /* 126 */ "G" // cnt=1 len=2
};
```

Listing 3. Array for index pairs (calculated for 8-bit indices).

```
const dict_index_t pair[][2] = {
   /*  0 => 127 */ {   2,   0 },
     // "-", "Sensor", cnt=78
   /*  1 => 128 */ {   1,   4 },
     // "Bank", "1", cnt=40
   /*  2 => 129 */ {   1,   3 },
     // "Bank", "2", cnt=40
   /*  3 => 130 */ { 128, 127 },
     // "Bank 1", "- Sensor", cnt=31
   ...
   /* 57 => 184 */ {  15,  51 },
     // "Air", "Flow", cnt=4
   ...
   /* 67 => 194 */ { 184,   0 },
     // "Air Flow", "Sensor", cnt=3
   /* 68 => 195 */ {  56, 194 }
     // "Mass", "Air Flow Sensor", cnt=3
};
```

Step 2: Optimising the dictionary

If words occur in the text with the same frequency and are adjacent in each occurrence (for example, *Exhaust* and *Gas*), these dictionary entries can be merged into a single entry (in this case, *Exhaust Gas*). Although this requires adding the space character between the two words in the new entry, it eliminates one of the two end-of-string characters.

After this merger of the two entries, it may happen that the merged entry again meets the previously mentioned condition with another adjacent entry term, so further entries containing more than two words can be formed in the same way.

As each entry in the dictionary (C string array) needs memory space for a pointer to the string as well as memory space for the string, this saves a pointer for each merged entry. The actual savings depend on how much memory space is required for a pointer in the target system. Furthermore, the size of the array `dictIndex[]` created in Step 3 is reduced by the frequency of the merged entries times the factor `sizeof(dict_index_t)`.

.

Step 3: Coding the original text

Once the dictionary has been generated, the text can be coded using the array indices of the dictionary entries.

The following is an example of coding the first text string in Listing 1:

`text[0]` = *Fuel System 1 Status*

- *Fuel* = dictionary entry `dictionary[5]`. Replace *Fuel* by the index 5.
- *System* = dictionary entry `dictionary[45]`. Replace *System* by the index 45.
- *1* = dictionary entry `dictionary[4]`. Replace *1* by the index 4 (no compression).
- *Status* = dictionary entry `dictionary[30]`. Replace *Status* by the index 30.

The space characters are deleted. The end result is the following array:

```
const dict_index_t dictIndex[] = {
   /*   0 */ 5, 45, 4, 30, /* Fuel System 1 Status */
   /*   1 */ 5, 45, 3, 30, /* Fuel System 2 Status */
   …
   /* 189 */ 16, 6, 1, 3, 2, 0, 77 /* Exhaust Gas
   Temperature Bank 2 - Sensor 8 */
};
```

As the number of indices per compressed text string is variable, a second table is needed for the pointers; for each text string in Listing 1, it contains a pointer to the first index of the text string. This allows each text string to be decompressed independently of the other text strings.

```
const start_index_t startIndex[] = {
   /*   0 */    0, /* first index of text[0] is
```

```
   dictIndex[0] */
   /*   1 */    4, /* first index of text[1] is
   dictIndex[4] */
...
   /* 188 */ 1207, /* first index of text[189] is
   dictIndex[1207] */
   /* 189 */ 1213  /* first index of non-existent
   text[190] is dictIndex[1213] */
};
```

The last entry in the array `startIndex[]` makes it easy for the decompression routine to calculate the number of indices for all text strings, including the last one. The number of indices for text string [n] is `startIndex[n+1] - startIndex[n]`.

Step 4: Adding index pairs to the dictionary

This step compresses frequently occurring word pairs (index pairs) and word sequences (index sequences). First we look for the most frequently occurring index sequence consisting of two indices. This is added to the end of the existing dictionary as a new logical entry. With the implementation shown in **Listing 3**, what actually happens is that a new two-dimensional array `pair[][2]` is created and expanded with an entry consisting of two indices. If the largest index of the dictionary was previously 126, the first index pair in `pair[0]` assumes the logical index 127. Then all index pairs from `pair[0][0]` and `pair[0][1]` (index 2 and index 0 in Listing 3) must be replaced by the single index 127. As a result, the size of the array `dictIndex[]` is reduced by the frequency of the index pair, while exactly one entry containing this index pair is added in the array pair[][2]. By applying this step to `dictIndex[]` as long as pairs with a certain minimum frequency of occurrence are found, it is possible to find not only two adjacent words but also longer word sequences. An index pair can then contain indices from one or two other index pairs, such as entry 3, index 67 or index 68 in Listing 3.

The frequency of occurrence of a pair must fulfil the following condition so that compression is actually achieved and a new pair is entered in the array `pair[][2]`: index pair frequency >2× `sizeof(dict_index_t)`.

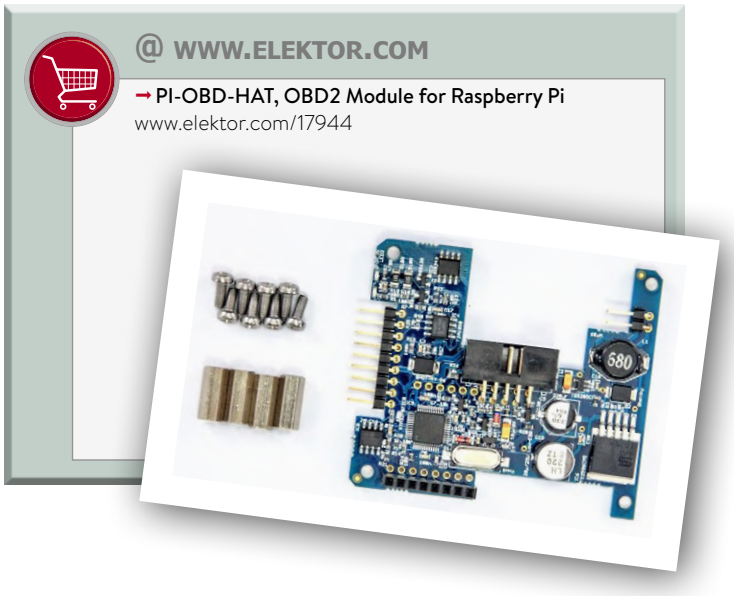
Step 5: Introducing ASCII indices

In the actual implementation there is an additional step, which cannot be completely described here due to its many special cases and conditions. By introducing ASCII indices for words in the dictionary that occur with frequency 1 or for short words with low frequency that cannot be used in `pair[][2]`, it is possible to remove words from the dictionary that do not contribute to compression. These words are entered directly in the array `dictIndex[]` with their ASCII codes and removed from the dictionary. Before this is done, all other indices corresponding to values other than the values of the ASCII codes must be shifted, for example to values greater than 127 in the case of 7-bit ASCII codes. More details about this can be found in the published source code of the compression and decompression routines for the Arduino project [3].

Follow-up

Execution of the individual steps reveals a general difficulty with compression. The memory space requirement for an individual index `sizeof(dict_index_t)` is not yet known before compression, but it is needed in the algorithm in order to decide whether dictionary entries or index pairs should actually be created for low frequencies of occurrence and short words. In the example case with compression of the PID descriptions, 8-bit indices are adequate, which simplifies the algorithm and the decompression routine. However, compression of the trouble messages is not possible with just 256 8-bit indices. There entropy coding of the indices, which code the most frequent words with 8-bit indices and all other words with 16-bit indices, is used to counter the enlargement of the array `dictIndex[]`. If the text compression achieved with Step 3 or Step 4 is sufficient, the procedure can be terminated early. In that case the decompression routine needs less memory space and runs faster than if further steps were carried out. ⏮

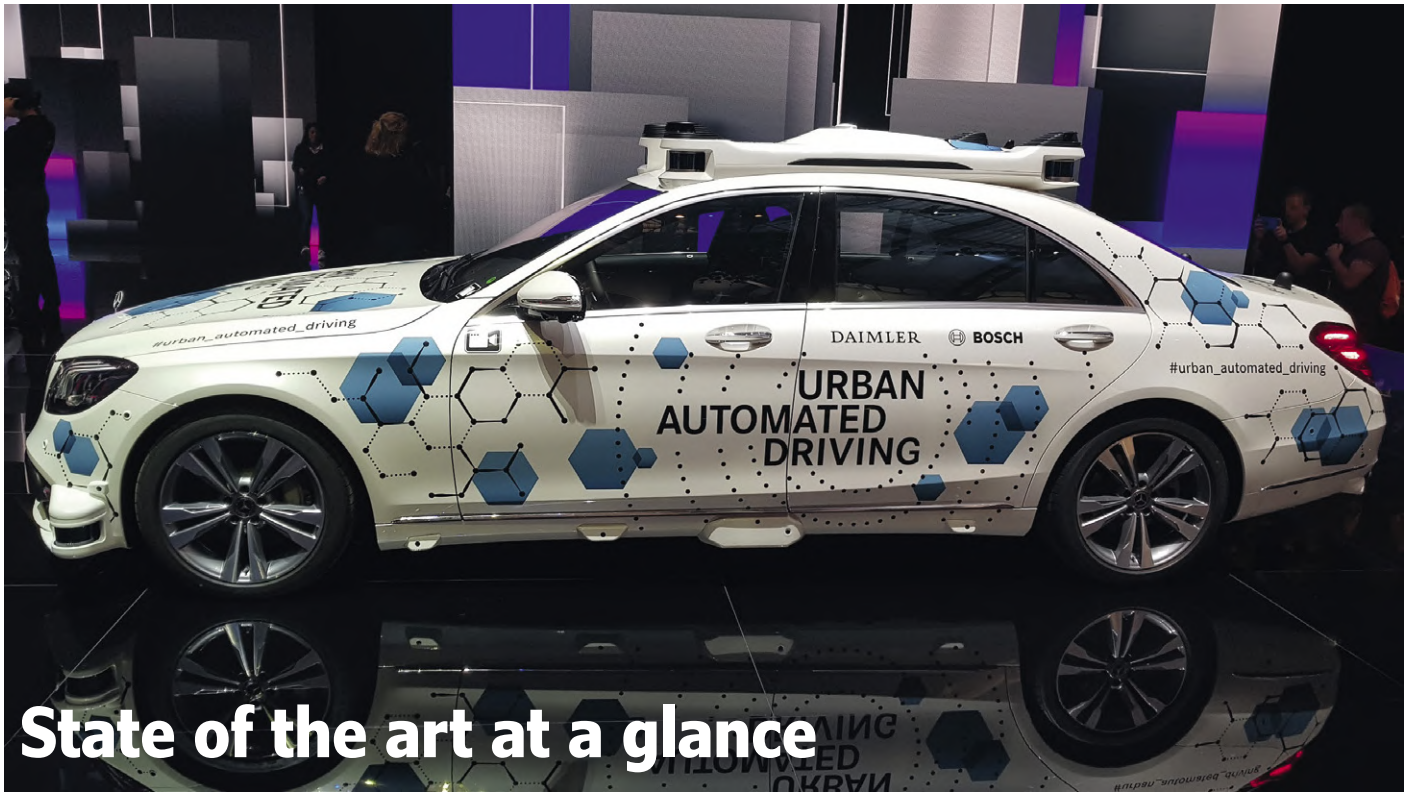
180278-02



Web Links

[1] OBD2 Analyser NG: www.elektormagazine.com/labs/firmware-update-and-emulator-for-obd2-analyser-ng-wireless-obd2
[2] OBD2 for Raspberry Pi: www.elektormagazine.com/labs/obd2-for-raspberry-pi
[3] OBD2 for Arduino: www.elektormagazine.com/labs/obd2-for-arduino
[4] Huffman coding: https://en.wikipedia.org/wiki/Huffman_coding
[5] LZ77: https://en.wikipedia.org/wiki/LZ77_and_LZ78

Focus On:
Autonomous Driving



(image: Viacheslav Gromov, IAA)

State of the art at a glance

By **Viacheslav Gromov** (Germany)

Autonomous driving is topical, and all over the place: headlines of accidents with the Tesla autopilot are reaching us from the USA, in Europe, automobile giants are uniting to remain competitive, and in China the first autonomous buses are in operation. From a technical point of view, where do we stand right now? What mobility concepts are we striving to reach, coupled to which social challenges?

Computer assistance in vehicles, as we know it today in the form of lane departure warning or voice control, has a long technological history. The beginnings are already exciting. A gyroscopic autopilot, first introduced in 1914, was designed to stabilize aircraft and is regarded as the first milestone in control automation. Later, autonomous vehicles (AV for short) were developed more in the military context (e.g. for mine detection). In the media, however, astonishing visions of mobility were also presented frequently. These include a series of remote-controlled test models and the string line vision shown in **Figure 1** [1].

The promises

Nowadays, there are the following central arguments in favour of autonomous driving: safety (human error causes about 90 % of all accidents), comfort, comprehensive mobility for all, and relief of our traffic areas with better environmental efficiency. When it comes to safety – the most important argument for many – it is assumed that autonomous vehicles also minimize the relatively frequent causes of human accidents (distraction, alcohol, ...) thanks to the C2X mentioned later, and that, in return, only a few new accident scenarios arise, for example due to incorrect calculations or hacker attacks. However, there is no clear, quantifiable evidence for this higher level of safety, which is why the factors – also discussed in the human sciences – are purely speculative in many cases.

With regard to mobility, it should be added that driving licences are no longer required and that every social group previously excluded can therefore use a vehicle. Such vehicles can also be used on demand (especially minibuses) in rural areas.

When it comes to traffic relief, studies predict an increase in road use with 21% and 80%, with AV coverage of 50% and

90%, respectively [2]. Fuel efficiency can be increased by up to 40%, not only by a regulated traffic flow with fewer braking and acceleration processes, but also by fundamentally different speeds. After some calculations, you can get from A to B faster, even at slower speeds, so the overall speed increases.

Concepts and degrees of automation

A promising (follow-up) technology is car-to-car (C2C) or car-to-infrastructure (C2I) communication, which, with other variants, is also known by the abbreviation C2X, where 'V' for vehicle is often used instead of 'C' for car. Vehicles can communicate with each other, for example to react much earlier to expected traffic jams; or interact with the infrastructure, up to and including a vision that hardly any visible traffic signs are required and that traffic light switching times will be heavily adapted or completely eliminated.

Basically, the automation levels of the vehicle are defined as shown in **Figure 2**. The Level 1 systems commonly used today, such as ABS and ESP, are partly responsible for the decreasing number of road deaths over the last few years. In current vehicles, semi-autonomous systems corresponding to Level 2 can already be integrated, such as lane departure warning or a congestion assistant. Tesla's autopilot, which made headlines, could be classified as approaching Level 3.

In addition to the automation levels, the use cases, i.e. mobility models, still have to be taken into account. Many concepts are conceivable within the above-mentioned automation levels. You could start with autonomous valet parking, for example, in which a driver drives the vehicle to the shopping mall himself, gets out and then the car automatically drives to a parking space in the vicinity. The highest form could then be Vehicle-on-Demand – depending on the price, you could then be transported by a car, alone or together with others, after pressing a call button on your smartphone and ordering the vehicle as such.

With almost all target concepts, there is no need for an individual vehicle. If the positive effects mentioned above are taken into account, this can be a great advantage of the AV. The autonomous vehicles would then not have to stand still for 23 hours a day but be much more flexible. This also has another economic reason: according to current estimates, the redundant systems installed at Level 5 would be quite expensive despite series production and would not be affordable to everyone. The large automotive groups would therefore also be mobility service providers.

Now we want to explain (very briefly and generally) the technical operation of an autonomous vehicle, so that you get an idea of the terms and keywords, which can then be the basis for your own research.

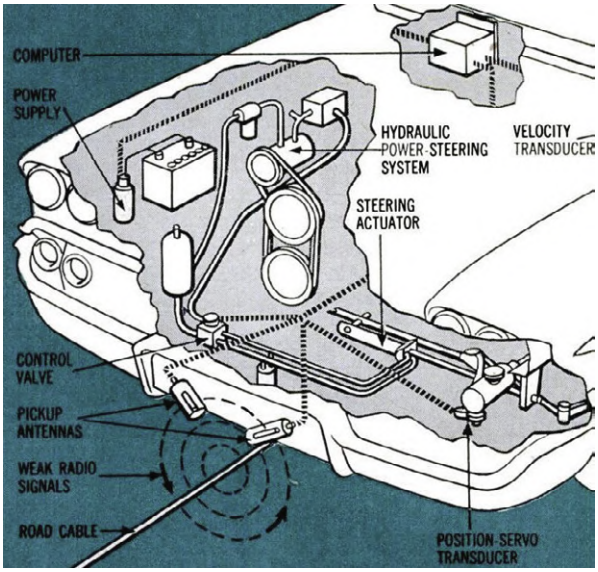


Figure 1: A prototype from General Motors, based on the 58 Chevrolet and equipped with two antennas near the ground. In 1958, the vehicle drove 'autonomously' on a test track equipped with a string guideline (source: *Popular Science* 05/1958).

A look inside

The first step (and the most critical step for humans at the wheel) is perception. Here the environment of the vehicle should be carefully observed. Camera, radar and lidar technology, whose strengths complement each other, play central roles in the autonomous vehicle. An example: while a camera is the only common way to recognize conventional colour plates and traffic lights, a radar can detect objects with very accurate dis-

SAE level	Name	Narrative Definition	Execution of Steering and Acceleration/ Deceleration	*Monitoring* of Driving Environment	Fallback Performance of *Dynamic Driving Task*	System Capability *(Driving Modes)*
***Human driver* monitors the driving environment**						
0	No Automation	the full-time performance by the *human driver* of all aspects of the *dynamic driving task*, even when enhanced by warning or intervention systems	Human driver	Human driver	Human driver	n/a
1	Driver Assistance	the *driving mode*-specific execution by a driver assistance system of either steering or acceleration/deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	Human driver and system	Human driver	Human driver	Some driving modes
2	Partial Automation	the *driving mode*-specific execution by one or more driver assistance systems of both steering and acceleration/ deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	**System**	Human driver	Human driver	Some driving modes
***Automated driving system* ("system") monitors the driving environment**						
3	Conditional Automation	the *driving mode*-specific performance by an *automated driving system* of all aspects of the dynamic driving task with the expectation that the *human driver* will respond appropriately to a *request to intervene*	System	**System**	Human driver	Some driving modes
4	High Automation	the *driving mode*-specific performance by an automated driving system of all aspects of the *dynamic driving task*, even if a *human driver* does not respond appropriately to a *request to intervene*	System	System	**System**	Some driving modes
5	Full Automation	the full-time performance by an *automated driving system* of all aspects of the *dynamic driving task* under all roadway and environmental conditions that can be managed by a *human driver*	System	System	System	**All driving modes**

Figure 2: An overview of the five automation levels including a brief definition (source: *SAE International* J3016).

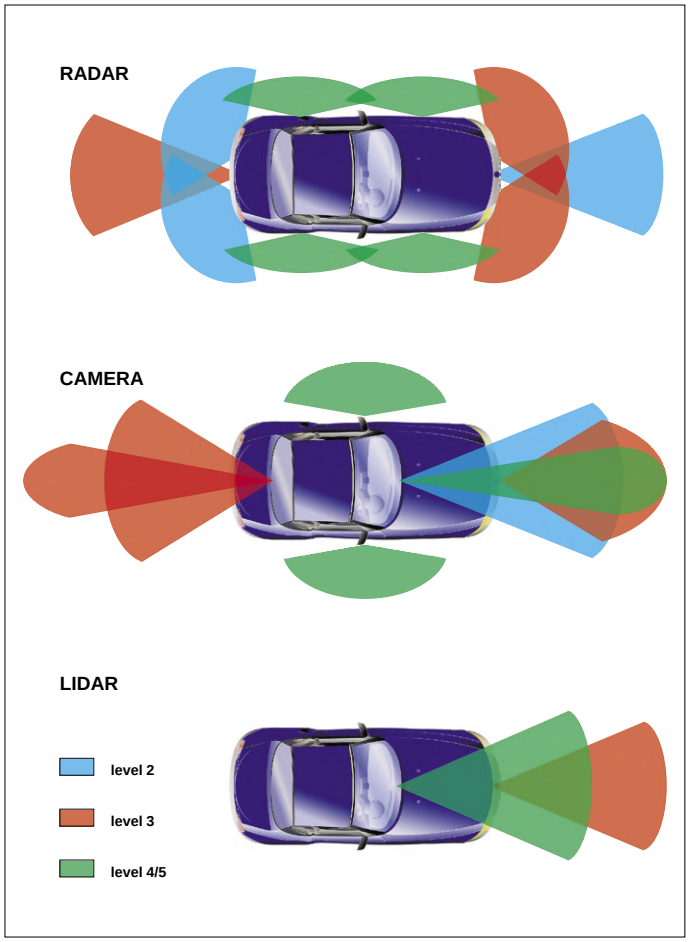


Figure 3: According to Infineon, the increase in sensors with increasing automation levels could look like this. Other sensors, such as ultrasound, are not considered here (source: Infineon).

tance readings even with very poor visibility. In order to get the most out of every situation, a sensor fusion is used, which means that several weighted sensor data are combined in a context in order to gain as many reliable insights as possible about the events.

Figure 3 shows an example of how the three main types of sensors might be used depending on the automation level, accepting that the data can fluctuate greatly depending on the manufacturer and development phase. The sensor cone of radar and lidar range from a few metres up to several hundred in the beam direction. Of course, the angular resolution and with it the total amount of data plays a decisive role. The commonly used HD cameras usually come with wide-angle focal lengths of 6 to 25 mm. The use of stereo cameras is not uncommon.

As a result of the perception, the movements of the objects in the near future (like the driving trajectory of a vehicle) should also be estimated along with all the necessary vectors (prediction), which is a very demanding task subject to assumptions. An example of the perception construction of Apollo can be seen in **Figure 4**. Apollo is an open platform for AV, created by the Chinese digital giant Baidu, in which some Western automotive giants are involved too. For more insight, a look at the corresponding Github project is interesting [3].

The goal of perception is also to classify objects correctly. Objects include road users as well as signs, traffic lights, road markings, obstacles and much more. A useful classification technique is semantic segmentation, which separates surfaces of certain classes (coloured in **Figure 5**). Semantic segmentation proves to be very useful, especially when it comes to roadway recognition and area estimation of other objects. Machine learning, especially Artificial Neural Networks (ANN), is excellently suited for the classification thanks to its pattern recognition and high robustness. In object classification, three

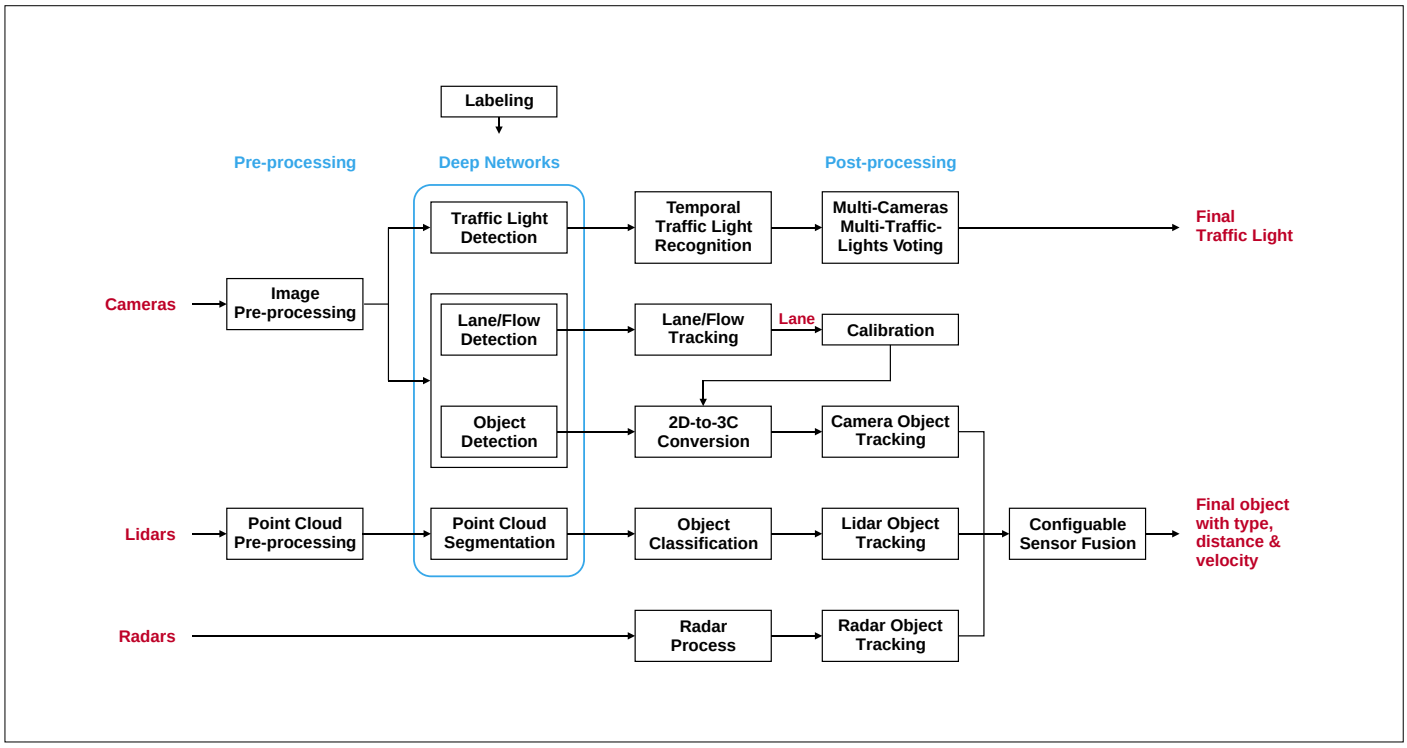


Figure 4: The structure of the Perception unit at Apollo. Object recognition with metric data and traffic light recognition are mostly carried out in parallel (source: Apollo).



Figure 5: Two common classification types: semantic segmentation on the left and object recognition on the right (image on the left courtesy MathWorks, right: symbolic image courtesy Rolf Gerstendorf).

uncertainties should be taken into account: the situation uncertainty as a result of measurement errors of the sensors, the uncertainty of existence with the question of the existence of the detected object, and the class uncertainty that affects the correctness of the assigned class. Precisely at the last point, the approximate estimation and consideration of the uncertainty is rather complicated.

Another task following the perception is the localisation of the vehicle within its environment. In order to find an answer to this "Where am I?" question to the nearest centimetre, the metric (object) data obtained from the perception is usually compared with a high-resolution map stored on board (**Figure 6**). Among other things, the comparison methods known from robotics for the measured point clouds are used. In simplified terms, this means that differences are minimized or certain typical objects (landmarks) are recognized when the originally vehicle-centred sensor data are superimposed on the map. Of course, GPS positioning data and motion data from the internal Inertial Measurement Unit (IMU) are also used. Kalman filters and many other methods (also in sensor fusion) are used to estimate the condition even in the event of inaccuracies or a lack of sensor data.

The solution with the internal HD card, however, is a double-edged sword. On one hand, it allows fast and efficient recognition, localisation and planning. Based on the information from the map, recognition algorithms for road signs or traffic lights can be used more efficiently, as they are aligned to a single region of interest (ROI) of the sensor field thanks to the information from the map. A major disadvantage is the size and timeliness of the map. With an ideal infrastructure, the map would have to be constantly loaded from the network into the internal memory within the specified radius. In order for the system to function as intended, it must be very up to date. For example, it is being considered that vehicles will update the map model resident in the cloud for others as they pass through (a partial aspect of possible continuous learning). Of course, it would be advantageous in future to neglect the internal HD card altogether, at least in its current form. However, it is not entirely clear whether this will happen at some point.

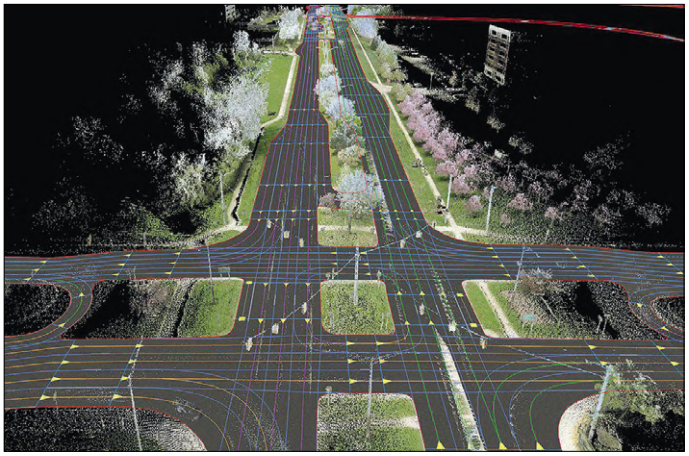


Figure 6: A partial view of Apollo's HD map of. The point clouds typical for lidar can be seen here in white, while no classified objects are shown here. (source: CB insights).

If the position and surroundings are known, the route, which usually is known ahead a few dozen metres, is planned (*planning*). The most favourable route is selected from a series of possible routes, after all others have been excluded by optimizing certain factors and taking into account physical, normative and other constraints. To make this efficient, the route options can be divided into segments, usually track sections. Naturally, the destination and the total route are the basis (total navigation). The last step before the actuators is control, i.e. the control system that executes the previously planned motion trajectory in the real world. To do this, it has to react to disturbances in real time and always take into account the physical possibilities of the vehicle (e.g. inertia). Many techniques are used, ranging from simple proportional-integral-derivative controllers (PID) known from control engineering to strongly model-based methods. The main actuators are, of course, acceleration, braking and steering axis, whereby the mechatronic technologies required for this can be used from today's existing drive-by-wire systems (DbW).

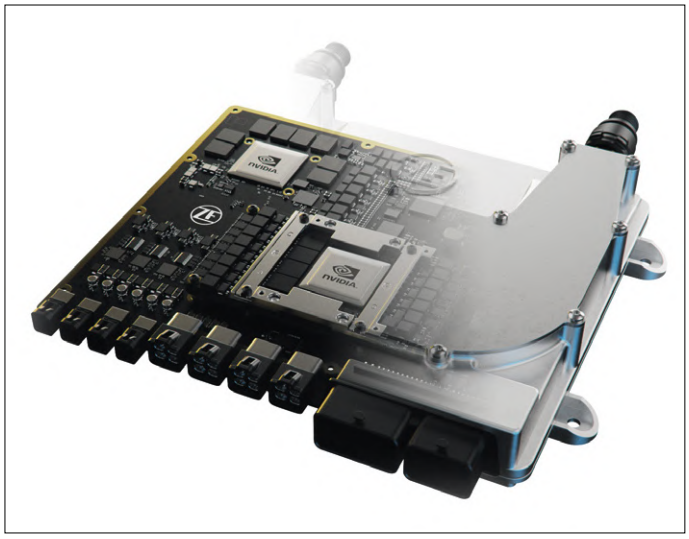


Figure 7: The ProAI RoboThink from ZF, based on NVIDIA DRIVE Xavier (or other chips if required). Waste heat is often a design challenge for systems of this type – water cooling is indicated here (source: ZF).

Depending on the manufacturer, the overall systems of such vehicles are usually divided differently between hardware and software components. There are mixed concepts with strong hardware-separated components or with large, heavily redundant central elements. This ultimately also applies to the interaction of deliberative and reactive system components. In particular, real-time capability and efficiency due to limited computing and storage capacity should be considered.

Nevertheless, there are many basic similarities. The pre-processing of sensor data during perception, for example, is often left to FPGAs or (in series production) ASICs due to high parallelization. Central processing units such as those in **Figure 7** with large GPUs, coupled with tons of parallel computing power are also usually used. The model shown here, in which up to four units are interconnected, is one of the most powerful ones on the market. A combined computing power of up to 600 tera-OPS (i.e. 600 trillion computing steps per second) can be achieved!

It is estimated that several billion miles are required for the development and approval of AV models instead of the usual test miles in the low two-digit million range, because the driver as an error-compensating component is increasingly eliminated and the system itself has to provide the required safety performance. Additionally, the system should be many times safer than today's human-car pair.

As these many miles cannot be covered on roads, much of AF's work is trained, simulated and tested at software level (software-in-loop, SiL) or on the test bench (vehicle-in-loop, ViL). At SiL, most well-known game and graphics engines are built with virtual high-resolution three-dimensional environments (**Figure 8**). Since an infinite number of driving situations could be created, methods are being developed to limit the situations to the most safety-critical and frequent ones. In principle, the highest safety requirements apply to the systems as well, as is the case with the ASIL-D class with a failure probability of less than 1/108 per hour. And there are many other new safety standards under discussion.

Finally, it must be emphasised that with algorithms, sensors and powerful semiconductors, many improvements should be expected – both in performance and price.

Challenges

There is no doubt that social acceptance is one of the biggest obstacles to this technology. According to studies, the fear of a computer error is even greater than that of a hacker attack – a risk known from everyday life.

Arguably the most explosive point however is ethics, which sooner or later must be cast in legal form. In many accidents, people do not even have time to respond in a morally sound way; the machine must be able to do it (and this is not over-moralization)! In situations of dilemma, a complex network of machine ethics can emerge, and many of the questions that come with it ultimately depend on basic norms of society (including global ones).



Figure 8: Simulation systems based on 3D engines are common in AV - here an example of AAI with some basic building blocks such as road signs or a construction site on the left side (virtual image: AAI).

Of course, the ethics commission set up for the AV in Germany banned the weighting of life according to characteristics (age, gender, ...), but considers harm minimisation to be justifiable for individuals (Rule 9) [4]. But you can go much deeper and more technical: which sensors have to be considered in such decisions? Where do you draw the line? You could go as far as the tyre condition, road conditions and beyond – parameters that are often already measured by sensors today. But the discussions are still going on. Try a simple intellectual game: If the vehicle can no longer stop and you have the choice between a collision with a motorcyclist with a helmet and a collision with one without a helmet, how should the vehicle act? The helmet wearer would have a higher chance of survival, but why should a motorcyclist wear a helmet and thus have a higher accident risk? In addition, there are a number of discussions about the passive role of the vehicle, a random variable, the determination of the 'safe condition', the scenario catalogue for registration, and countless other topics. Redundancy/safety and computing time should always be taken into account when making these decisions.

But ethics could also be abstracted in a much more general way, with the question of whether, above a certain security gain, the legislator can force citizens to use AV exclusively. It has to be said that, depending on AV interaction, mixed traffic can significantly disrupt regulated traffic, but can be technically well tolerated through adaptation. But what about freedom and driving pleasure? The ethics committee clearly rejected this constraint; old-time friends and posers can breathe a sigh of relief!

Legally, the liability risk is an exciting topic in this context. Even if product liability is to continue to apply for manufacturer's defects and even if the driver is responsible for assessing whether or not he can leave the autonomous pilot switched on in the current traffic situation, new, unforeseeable situations still arise – for example, if the first legislation worldwide on AV in Germany from 2017 applies. One example: the technology of machine learning works with universalised probability arrangements, which should react accordingly to unknown situations. But what happens if this 'black box' spots something wrong in the individual situation and acts in error?

Experts are also convinced that because of its scope and social responsibility, AV can only be an overarching project. It is not without reason that platforms such as Apollo or the well-known (internal) AV collaborations such as those of Daimler and BMW have been created. Either way, common (and explicitly political) rules and regulations are inevitable and make a significant contribution to technological success. Cross-vendor software environments such as the young, open project called OpenADx by the Eclipse Foundation in cooperation with Bosch and many other participants are a good start [5].

In some areas, many countries seem to offer better conditions for a rapid development of this technology than Germany (with the well-known test field on the A9). In the USA, there is more legal freedom to test vehicles without a driver (control/emergency stop by remote control) and in China the required training, test and simulation data is much easier to access – there, all motorways are even mapped for the AV HD.

Finally, the entire traffic concept for AVs needs to be reconsidered. This creates completely new challenges for urban and rural traffic planning.

In spite of technical and structural challenges, this technology is being driven forward worldwide as never before by current investments, subsidies and regulations. The framework conditions are also being worked on more diligently now. And it is well known that hard work pays off! ⏮

191147-03

The Author



Viacheslav Gromov is an entrepreneur, developer and freelance journalist in the field of electronics. Since his school days, he has been writing articles and reference books for various magazines and publishing houses. Thanks to the program for gifted students at the University of Freiburg, he was able to begin his studies while still at school. His areas of expertise are artificial intelligence (Edge AI), (ARM-/wireless-) MCUs and FPGAs, about which not only wrote textbooks for universities and semiconductor manufacturers, but also participated in international electronics developments from South Africa to Silicon Valley. He recently founded a company to develop local AI solutions for industrial plants and normative hybrid systems (patent pending) for autonomous driving (readers@gromov.de).

Web Links

[1] Autonomous Driving – Technical, Legal and Social Aspects, M. Maurer et al., Springer Open, 2015 (in German): https://link.springer.com/book/10.1007/978-3-662-45854-9

[2] Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations, D. J. Fagnant et al., 2015, Elsevier: www.aamva.org/WorkArea/DownloadAsset.aspx?id=6738

[3] Apollo Project: http://apollo.auto/

[4] Report of the Ethics Committee on Automated and Networked Driving, BMVI, June 2017 (in German): www.bmvi.de/SharedDocs/DE/Publikationen/DG/bericht-der-ethik-kommission.pdf

[5] OpenADx: https://openadx.eclipse.org/

Start-Up Leaders and Innovators Talk "Innovation 4.0" in Munich

By **Rachit Syag** and **Udo Bormann** (Elektor München Correspondents)

Elektor recently co-sponsored "The Top 50 Who's Who" event, a night of networking and idea sharing in Munich, Germany. The start-up founders, engineers, and innovators in attendance took advantage of the gathering to exchange business cards, enjoy drinks, and discuss the future of technology and entrepreneurship!

Top Innovators across Europe recently met at "Top 50 Who's Who: Innovation 4.0" in Munich, Germany, to discuss ways to bridge the gap between hardware, AI, IoT, and software. As a part of the Bits & Pretzels Start-Up Night, the event — which was sponsored by Sourceability, TechFouners, Brinc, Invest in Bavaria, and Elektor — enabled start-up founders, engineers, and innovators to exchange business cards, enjoy drinks, and discuss the future of technology and entrepreneurship. A fine blend of sharing and collaboration made the event special with deep insights into the future of technology and entrepreneurship!

In this first installment of *Start-Up Zone* we review some of the presentations and highlights from the evening.

Start-Up potential in Europe and beyond

Jens Gamperl, CEO Sourceability, addressed the hot topics of AI, machine learning, blockchain, autonomous driving, and smart cities. He explained that the US appears to have more options as it relates to financing, an abundance of emerging innovative tech solutions, and experiences with rapid growth. However, he believes Europe's ability to foresee the long game is its unique selling proposition (USP).

"In addition, Europe's highly educated talent pool enables global competitiveness and fuels its capability to build sustainable business models," Gamperl said.

Jewell Sparks (Global Director for innovation & strategic partnerships, Surcle.io) shared her views about both Europe and the US being great starting grounds for founders. Corporates are engaging heavily with startups in the areas of Artificial Intelligence, Machine Learning and Augmented Analytics in Europe as per Jewell's observation. She also talked about 'diversity and inclusion' being a big topic as it relates to innovation in the US. Also, venture capital firms, accelerator programmes, and organizations are holding their leadership teams accountable for both diversity and innovative technology integration and scalability, said Jewell.

Heiko Huber (Managing Director for TechFounders UnternehmerTUM Projekt GmbH) helps start-ups with networking activities. In his talk, he said the key benefit for participants at such events is to meet new people who can help with projects and business development. "This can be done by exchanging experiences, by looking at one's own ideas from a different angle (e.g. hints on a use case of a start-up idea that has not yet been considered) or by initiating a cooperation or an investment," he said.

Bernd Wunderlich (Digital Project Leader, Gartner) gave an interesting presentation on the topic of securing a new foundation for digital business (**Figure 1**). In addition to talking about digital statistics, he described the results from a survey that Gartner conducted among the DACH region CEOs. The survey covered the 10 most important future business activities, and as per the latest survey, digital initiatives ranked the highest. It was followed by the focus on activities concerned with revenue and business growth. It is interesting to note that digitalization is the top priority for the decision-makers and 49% have already changed their business model. Out of the 49%, 28% are using the said digitalization to scale the business whereas 51% of the total haven't yet started the process of digitalization.

Innovator presentations

Six companies had an opportunity to present their products and solutions at the "Top 50 Who's Who" event. The technologies

ranged from wearables to AI. Weblinks to the innovator companies' websites are given below.

Teiimo - Markus Strecker, Founder & CEO: With the aim of integrating electronics and textile, Teiimo is working in the field of conformable electronics. Mr. Strecker presented their new medical sensor shirt at the event which monitors patient's heart data in real time and talked about its benefits in the healthcare sector. Note: Teiimo won First Prize in the electronica Fast Forward 2018 competition.)

Franck.AI - Isabell Franck, Founder & Managing Director: In the era of Industry 4.0 & Artificial Intelligence, Franck.AI is determined to provide innovative software solutions. They are focused on combining machine learning and customer-specific expert knowledge to develop software solutions to optimize production and development activities.



Figure 1. Bernd Wunderlich talks about securing new digital business.

Smartfurniture - Joerg Sahlmann, Co-Founder: As the name suggests, Smartfurniture combines furniture with electronics and software to develop smart work and leisure equipment. The company is focused on four key elements when it comes to making smart furniture: design, function, quality, and price.

Swap Language - Nicholas Møller Walsted, CEO: With an innovative approach to language learning, Swap Language provides the opportunity to learn a new language with a native speaker of the language. With a simple yet impactful solution, the company helps people find a native speaker of the language they want to learn in their city and gives them the opportunity to teach their own native language in return.

Valuer.ai - Signe Andersen, Regional Manager, Germany: Using crowdsourcing and artificial intelligence, Valuer.ai helps accelerators and corporations to find startup with high potential that match their strategic innovation requirements.

Rydies - Andreas Nelskamp, Managing Director: Rydies provide micro-mobility solutions like bicycles, e-bikes, and other rideables in the age of rising traffic and people spending more time on the road and looking for parking spaces. The company focuses on providing a stress-free travel experience for short distances while battling against increasing pollution from car fumes.

Future opportunities

The "Top 50 Who's Who" event in Munich was yet another opportunity for tech innovators to gain the support they deserve. Apart from being a useful and reliable resource to the electronics engineers' community from more than 80 countries, Elektor strongly believes in promoting innovation and entrepreneurship. Through the Elektor Labs' platform and global start-up competitions like productronica Fast Forward), Elektor is continuously empowering start-up innovation. ⏮

191151-01

Web Links

[1] Teiimo: https://teiimo.com/

[2] Franck.AI: https://franck.ai/

[3] Smartfurniture: https://smartfurniture.de/

[4] Swaplanguage: www.swaplanguage.com

[5] Valuer.ai: https://valuer.ai/

[6] Rydies: www.rydies.com

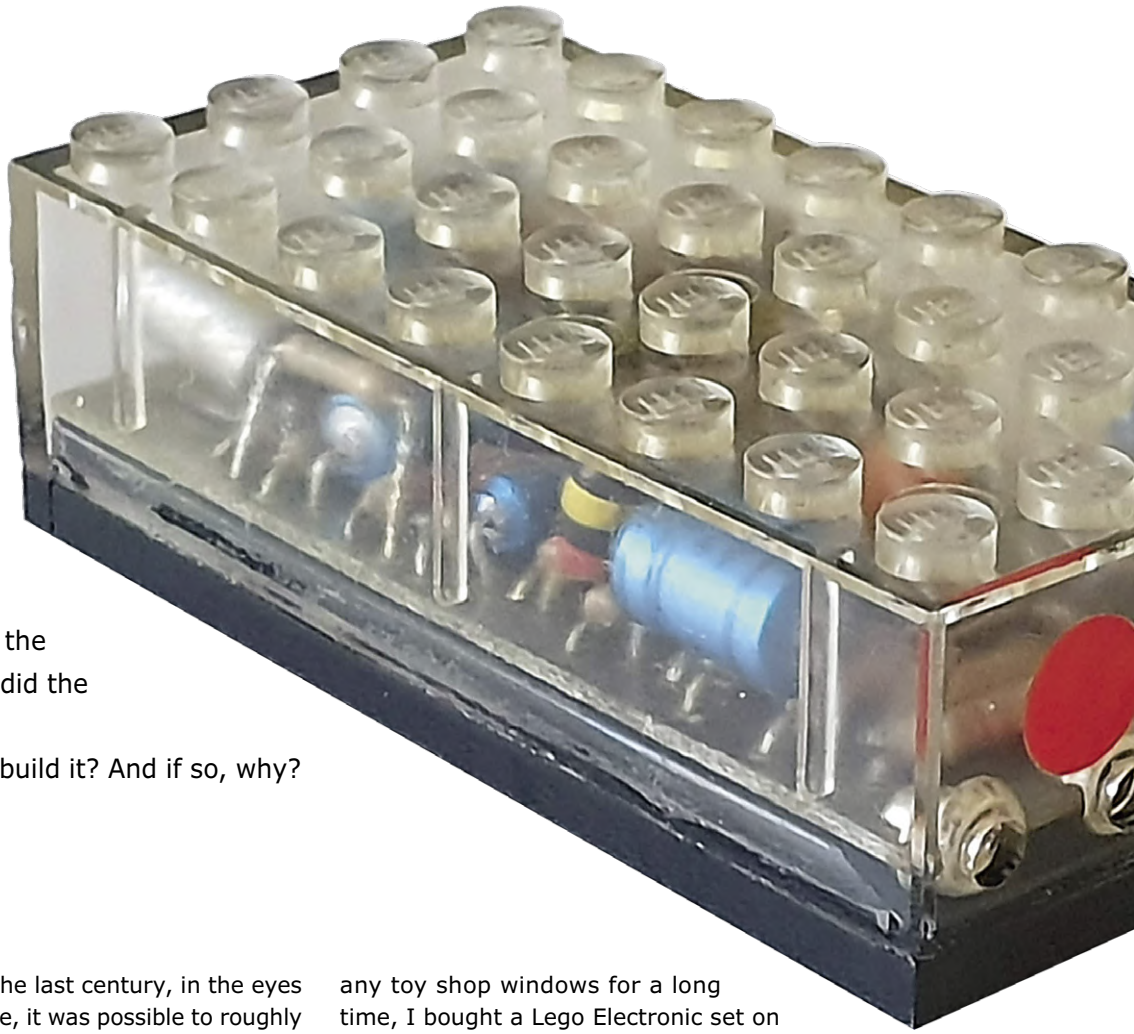
Collaborators

- Brinc - Vanessa Narvios. **www.brinc.io**
- Invest in Bavaria - Andreas Fischer. **www.invest-in-bavaria.com**
- Sourceability **/** Surcle.io - Jens Gamperl. **https://sourceability.com**; **www.surcle.io**
- Elektor - Udo Bormann & Rachit Syag. **www.elektormagazine.com**

Lego Electronic Train Set from 1968

50 years on, electronic toys still fascinate

By **Martin Kompis** (Germany)



A childhood dream and unachievable to most: A Lego Electronic Train set as it was launched commercially in the sixties. A whistle, and the locomotive started running. And these small boxes still work today. But what's inside the boxes and what design roads did the developers take at that time? And ... is it also possible to rebuild it? And if so, why?

Towards the end of the sixties of the last century, in the eyes of a child just under 10 years of age, it was possible to roughly divide mankind (i.e. the equally-aged part of mankind at that time) into two groups. On the one hand there was the vast majority not in possession of a Lego Electronic Train set, and on the other hand, the tiny minority, envied by everyone else, who called it their own. Take a guess to which group the author belonged at that time. Good guess! Not to the envied minority.
As we have all come to experience, adulthood brings a lot of discomfort with it: paying taxes, working regularly in the absence of suitable hereditary uncles, taking out the garbage, not yawning at the New Year's Reception and laughing at the right jokes, just to name a few common examples. But there are some advantages to adulthood, one of them being you can now finally buy all the toys you need so badly.
Since, in the present case, the desired object hasn't graced any toy shop windows for a long time, I bought a Lego Electronic set on the Internet. It proudly bears the type number 139A and comes with a microphone, a Lego whistle and even the original box **(Figure 1**).
The function of the system is quite simple and is probably hard to represent better than on the inside of the lid in Figure 1: if you blow once on the Lego whistle, the train starts moving. If you whistle a second time, the train stops again. But instead of whistling, you could also just clap your hands, for example. Okay, that may not sound like a sensation today in the age of smartphone-controlled and camera-studded drones for kids aged three and over (maybe six). By the end of the sixties, however, when remote controls for televisions existed but were a rare sight in this country, this was a really tempting toy that promised hours of fun.

Over the years 1968 and 1969 two different versions of Lego Electronic were available, which were sold in four different Lego sets altogether with the numbers 118, 138, 139 and 139A. The older version from 1968 can be found in the 118 and 139A sets. The latter can be seen in Figure 1 and contained an electronic 'brick', a microphone in a white plastic housing, two pairs of cables, and a whistle, which will be discussed shortly. The larger Lego set number 118 contained a complete locomotive with a tender (**Figure 2**). Just one year later, a new electronic brick arrived, which again was available either individually under number 139 or complete with locomotive and tender as set number 138, the latter with a new, smaller engine.

While the older version of the electronics is entirely composed of discrete components, the one from 1969 contains an integrated circuit with the cryptic designation '211 OM' [1]. This allows the train not only to drive forwards, but also to reverse, remotely controlled by a whistle.

A quick glance at the box I bought at the online auction shows that this is the older, all discrete-parts version. This is not only more suitable for the Retronics section, but also easier for the analysis and the replication of the circuit, since the IC is possibly a custom-made (or programmed?) component for this application, like a kind of precursor of today's FPGAs.



First there was acoustics, then electronics

The electronic block is just the second part of the whole system. The first part is the acoustic signal generator, i.e. the Lego whistle supplied (Figures 1 and 2). And it has everything. The whistle purchased by the author works perfectly together with the circuit but it's disturbingly loud, subjectively speaking. An initial measurement also shows that a sound level of around 95 dBA is reached at a mean air flow at 1 m distance. That's quite a number. I even measured 105 dBA in front of my ear, which is clearly less than 1 m away from the whistle. Both the level and the frequency of the whistle sound vary slightly with the strength of the airflow. It is around 5.7 kHz for my device. The human ear seems to be most susceptible to acoustic overload at frequencies around 3 to 6 kHz [3] and 5.7 kHz doesn't seem to be the best choice at these quite high levels. After all, a somewhat lower level is enough for reliable switching – with my device about 82 dBA is enough.

The electronics

Still, by far the most exciting and valuable part of the whole system is and remains the electronic 'brick'. It is packed with electronic components, of course all older and all wired, since SMD technology wasn't around in 1968 [2]. The whole upper side of the case is transparent, which was already part of its appeal to children with an affinity for technology. Even in the finished locomotive, the electronics are not hidden from view, but proudly and visibly displayed to everyone (Figure 2).

The connection to the motor is made via two metal studs at the underside of the electronic block, the connections to the microphone and to the battery box via two 2.5-mm jack pairs



Figure 1: The Lego Electronic set from the sixties: on the left the actual electronic block, in front the microphone and on the right in front a whistle, which definitely is high-performant...

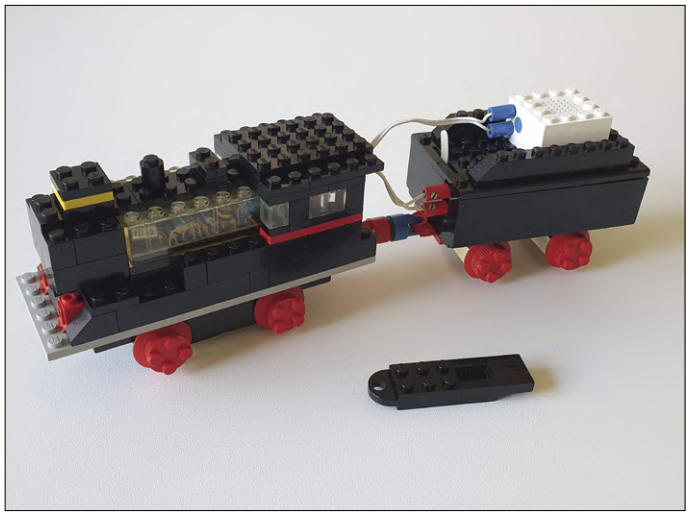


Figure 2: Replica (unfortunately no original) of locomotive no. 118. The electronic module is proudly displayed.

on the front. They are marked with a blue dot for the microphone and a red dot for the power supply (Figure 1).

This raises initial questions about the circuit. The polarity of the supply voltage is not marked anywhere. But that's of no concern really since the polarity of the operating voltage can be changed with a short flip of the lever on the battery compartment. Mechanically, all sockets are interchangeable, so you can connect the microphone where the motor belongs, the battery where the microphone belongs, and so on. And the whole thing is intended for the experimenting child's hands and should therefore be rather fault tolerant. A short test shows that the circuit only works as intended with one polarity of the supply voltage, but even with reversed (wrong?) polarity it does something quite useful: the train reverses, albeit much slower. If the motor is subjected to mechanical stress, a current of at least 80 mA flows rapidly. Inside the electronics block, however,

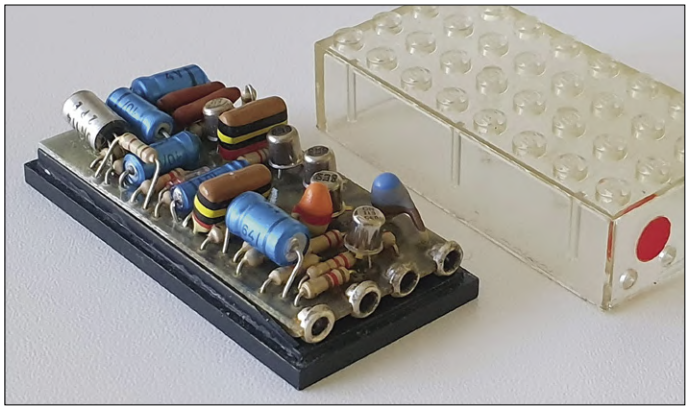


Figure 3: The Lego electronics after opening the casing. The germanium transistor is on the far left.

the transparent plastic cover means that no power transistor is visible at first glance, nor are there any diodes to protect the circuit from incorrect polarity. So how was this done back in 1968?

To answer these and other questions, the schematic would be useful. I don't find it on the Internet (at first) and so I had to open the nice case. (As it will turn out later, I simply didn't search properly: the schematic can be found on the Internet [1], even in two variants. However, both do not reflect the polarity of the transistors correctly – the circuit would not work as indicated.)

The upper part of the case is glued to the base plate and I had to wield my Swiss army knife. For this outrage against a historic Lego component I am bound to go to Lego hell (or is it called toy hell?). It is possible to remove the upper part without seriously damaging any of the plastic parts, and the electronics are now in full splendour in front of me (**Figure 3**). If you lift the small, single-sided coated board a little, you will find a yellowed note with the text 'SEP. 1968'. Very pretty.

With the aid of tracing paper, a multimeter and a sharp eye, the circuit can now be traced. The component placement proves to be a (small) challenge: Because of space, the parts are partially over another, so that a few resistors and a diode can only be seen by bending away the parts above them.

Figure 4 shows the resulting schematic. The small circuit board accommodates six transistors in metal housings. Five of them in small TO-18 packages are labelled 'ON 113'; the last one in the larger TO-1 package is labelled 'ON 114'.

The designations do not follow any system known to me and both transistor type designations are non-existent on the Internet. One measurement shows, however, that the ON 114 must be a PNP germanium transistor, while the five other transistors are NPN types and — somewhat surprisingly to me — silicon transistors.

In principle, the function of the circuit is easy to understand. Transistors T1 and T2 form an amplifier for the microphone signal. High fidelity is not a requirement here, but high amplification is. And that's why the signal at the T2 collector looks highly distorted. Essentially, only parts of the negative half-waves are amplified, and even that is not particularly cleanly. On the other hand, the signal easily reaches peak-to-peak values of about 2.5 V.

Together with capacitor C4 and resistor R9, T3 forms a switch controlling a bistable multivibrator consisting of T4 and T5. When the circuit is idle, C4 is almost charged to the operating voltage via R9 (and thus via R10 also). If there is a whistle sound, T3 briefly comes on at each half-wave and thus discharges C5 quickly. The recharging of C5 then takes place much more slowly, with the time constant of almost 50 ms determined by C5 and R9, after the acoustic signal has stopped. The bistable multivibrator is controlled via C7 and C8. If T4 is conducting, the motor does not run. If, on the other hand, T5 conducts, the C-E path of the T6 output stage transistor starts to conduct and the toy train starts to move.

Although five of the six transistors in the circuit are silicon based, the developers have chosen a germanium transistor to drive the motor. This makes sense because the voltage drop is significantly lower when transistors and germanium types are switched on than with their newer silicon counterparts. In the present circuit, the drop is an incredibly low 0.03 V. The power loss that occurs at T6 is equally low.

A question of polarity

So far so good. Nevertheless, some things remain unclear. What happens if the polarity of the operating voltage is 'wrong'? Why does the train run; and why does it run slower backwards than

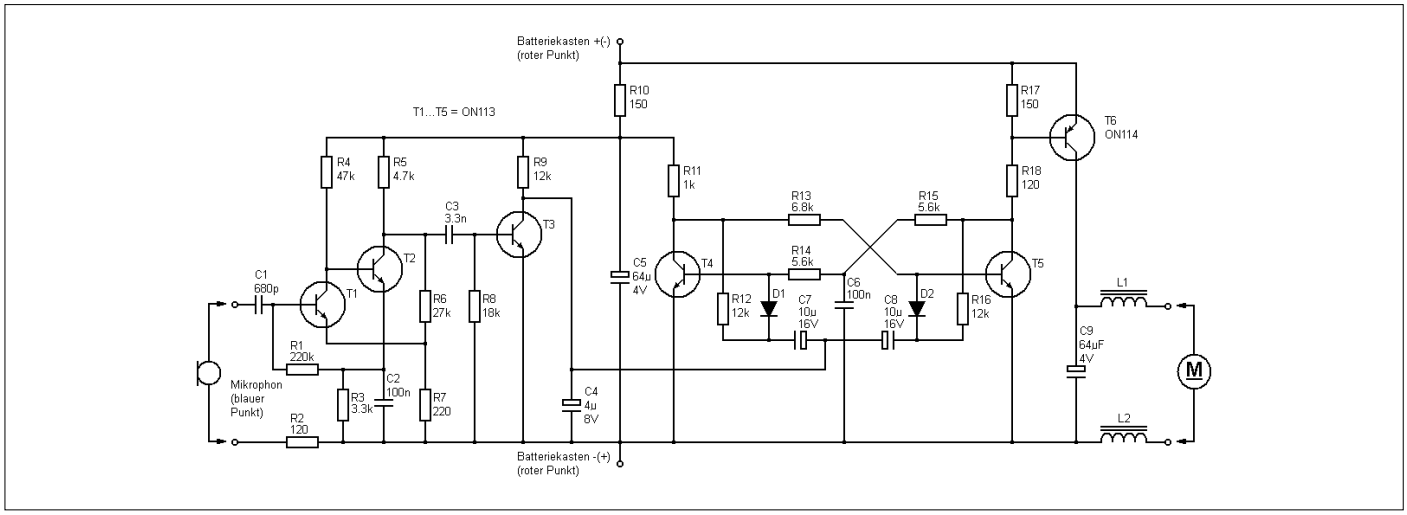


Figure 4: The circuit diagram. Transistors T1 to T5 are silicon devices, T6, good old germanium with a lower c-e drop.

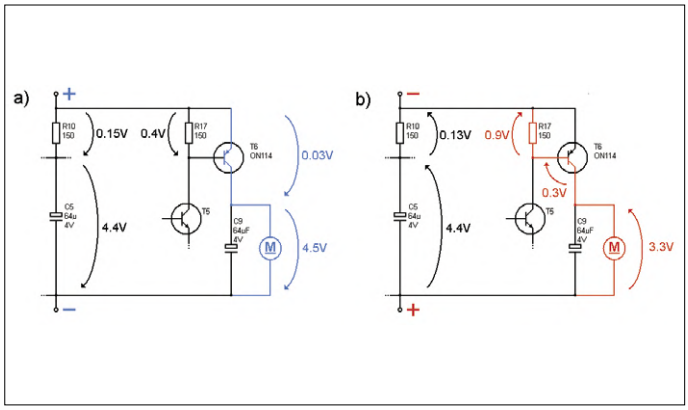


Figure 5: Some voltages (a) during forward travel and (b) during reverse polarity of the supply voltage at which the train slows down and moves in the opposite direction.

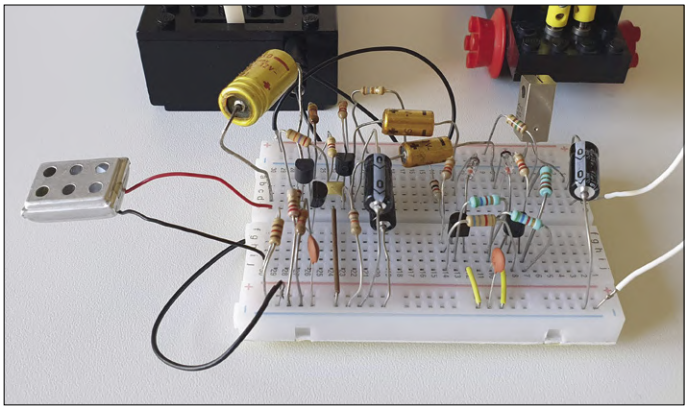


Figure 6: The replica of the Lego electronics. Rear right, the AC153K germanium transistor in its TO-1 housing with cooling block, which would not be necessary for this application.

forward? What happens to the individual circuit components if the polarity is 'wrong'? The first part of the circuit may be slightly protected by R10, but what happens to the electrolytic capacitors C5 and C9? Aren't they reverse-polarized then? **Figure 5** shows the two states and some relevant voltages during operation.

If the polarity is correct **(Figure 5a),** everything works exactly as described. In reverse gear (**Figure 5b**) on the other hand, the B-C path of T6 acts as a diode with the expected voltage drop of about 0.3 V. The current flowing through the motor must now also flow through the 150-Ω resistor R17, where another 0.9 V drop occurs. Accordingly, the motor then runs much slower, but it runs!

And the two electrolytic capacitors? Well, no unexplained miracles here. If the locomotive drives backwards slowly, the two electrolytic capacitors C5 and C9 are actually reverse-polarized, and the voltages on them are not insignificant. The developers have obviously accepted this, and it seems to work, even after more than 50 years.

The measurements on the narrow circuit board are not quite easy. In addition, it is of course tempting to rebuild the circuit to be sure that the circuit diagram is correct.

So the logical next question is: do we really have to rebuild it? As with all difficult questions in life, I consult my family first. The answer this time ranges from "Close the door" (youngsters at a "difficult age") to "Why would you do that for?" (loving wife — note: just about any age seems to be "a bit difficult") to convulsive closing of both eyes (cat, age uncertain but not a very new model). The overall resonance can therefore only be evaluated as extremely positive in comparison to earlier project ideas on my part, and I set to work immediately.

The reproduction

Figure 6 shows the replica on a small pinboard. Instead of the five ON 113 NPN silicon transistors T1 to T5, BC548C types were used and instead of the germanium transistor ON 114, a single device of a type AC153K was used, which had spent the last decades deep down in the junk box.

The gearshift works right away, just like the over 50-year-old original: one whistle, and the engine starts, a second whistle and it stops. If you flip the battery switch, the engine turns in the other direction, and much slower. Bulls eye. By the way, the microphone element was a small crystal type. Maybe the circuit is useful to someone who owns a faulty Lego Electronic set. And the replicated circuit has some entertainment value too. Have fun!

190382-03

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

Web Links

[1] Type 211 OM circuit: www.eurobricks.com/forum/index.php?/forums/topic/10989-electronic-train-118138-and-139-anyone-have-one-or-have-seen-one/&page=2

[2] SMT History (in German): https://www.all-electronics.de/eine-kleine-geschichte-der-smt/

Literature Reference

[3] Kompis M., "Audiology", Hogrefe Verlag Bern, 4th edition, 2016, ISBN: 978-3-456-85553-0, 4th chapter.

The MX3D Bridge Senses the City

How data collection can reduce the erosion of privacy

By **Tessel Renzenbrink**

The MX3D bridge is the first ever 3D printed metal bridge. Constructed by industrial robots adding millions of stainless steel welds over a period of six months, it is a unique engineering object. To test and measure the novel construction, the bridge is being fitted with a permanent sensor network. For Alec Shuldiner, Sr. Product Manager at software company Autodesk, the sensor data opens up a whole new world of possibilities. In this interview he explains how it enables us to not only look *at* the bridge but also to sense the environment *around* it. This can help us answer questions about its surroundings to improve management of public spaces. For instance, knowing how many people cross the bridge both tells us something about the crowdedness of the surrounding area and contributes to managing traffic flows. “What many people don’t realize is that data collection can actually reduce the erosion of privacy in the public space”, says Shuldiner. “But to achieve that we need to rethink how we’re designing the Internet of Things.”

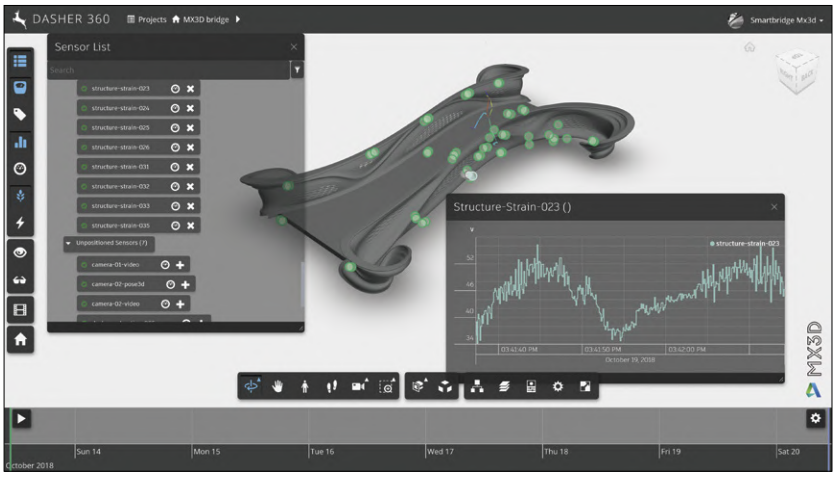


Figure 1. Dashboard showing sensor output of MX3D bridge. (Image: Courtesy Alec Shuldiner).

The MX3D project draws on the expertise of many different parties. In the lead is the Amsterdam-based company MX3D. They developed the technique for 3D printing in mid-air, dispensing with the limitations of a printer box or other support structures. The sensor network for the pedestrian bridge is designed and installed in a team effort that includes the Imperial College of London, University of Twente and Autodesk, the US company most famous for its AutoCAD application. Autodesk also provides the design software and, together with the British Alan Turing Institute, is working on machine learning algorithms to process the data. Another partner is the city of Amsterdam as MX3D’s customer. The plan is to place this novel piece of technology over one of the canals in De Wallen, the oldest neighbourhood of the city.

IoT: a window on the world

“The main argument for placing sensors”, says Shuldiner, “is that the bridge’s method of manufacturing is so cutting edge and unique that we needed to collect data just to understand how the technique works. The sensor data can tell us how the bridge actually functions. The aim is to build up a corpus of engineering knowledge to guarantee the safety of the bridge and to collect data for future designs of other objects made in similar fashion. That remains the primary interest for the project as a whole. But my personal interest is in the Internet of Things (IoT) angle. You can take the data that comes from the sensors and use it as a lens to see what is happening on and around the bridge. “The reason I am interested in sensing the urban environment is that we need to understand it much better to be able to manage it. Cities are growing more and more crowded, leading to intense issues of traffic flow. The De Wallen neighborhood, which is home to the Red Light District, is the epicentre of that problem in Amsterdam. It is one of the most crowded places anywhere on the planet. That isn’t just unpleasant, it can actually become a safety issue. Therefore, the city is



Figure 2. Sensors placed on the MX3D bridge.
(Photo: Courtesy Alec Shuldiner).



Figure 3. Alec Shuldiner in front of the MX3D bridge.
(Photo: Courtesy Alec Shuldiner).

constantly trying to figure out how to measure the crowds in order to manage them better. What we hope to achieve with the MX3D project is to have the bridge report on the number of people that are on it."

Cameras as training wheels

"We use the sensor system to get the bridge to understand how many people are on it at any point in time", Shuldiner continues. "Fundamentally there are two things that the sensors measure. One of them is the environment: what are the changes in temperature, the sound level, the light intensity and so on. The most important environmental sensors are the cameras which are recording activities on and around the bridge. The other set of sensors capture how the bridge is changing and moving in response to its environment: vibrations, frequencies, tilts, strains and displacements. The idea is to use the cameras to collect data about what is happening on the bridge and then correlate that information using machine learning to the physical changes in the bridge: the reaction of the bridge to those events. At a certain point the bridge will be able to tell us, simply from the physical response that it registers, how many people are on it. And then the goal is to get rid of the cameras. The cameras have been there for training purposes but at a certain point we won't need them any more.

Currently, the default technology to do these kinds of counts are cameras. Not just in Amsterdam but everywhere in the world. But cameras are problematic as a means of collecting data. Both in a practical and in a societal sense. Practically, they're not very precise sensors. Image data look very different when circumstances like light or weather change. Plus, it's a heavyweight data source, processing video data is computationally expensive. Besides the practical issues there is the very serious problem that cameras provide all the data you need to identify somebody with a high degree of accuracy. We used to walk around in public spaces with the assumption that we were not being spied upon. But that is not a reasonable expectation anymore. Placing cameras in the public space destroys ambient privacy.

Even if the camera is using obscuring technologies, there is also a psychological impact. In the MX3D project we skeletonize the video data: you don't see people, you only see stick figures. But the pedestrians in the area don't know that. They see a camera and naturally assume their image is being recorded in high fidelity. So even if a camera is not eroding your privacy, it is eroding your *sense* of privacy. Which is almost as bad. That's why we aim to remove the cameras after they have been used for training."

Rethinking IoT design

Shuldiner: "The use of cameras points to a larger problem I see with the Internet of Things. You have a simple counting task that is addressed with a heavyweight, suboptimal sensor that comes with negative side effects. We are not collecting the right data for the questions we have. We're not thinking ahead about which specific sensors we need. Even with MX3D, which is a cutting edge project, the sensors were an afterthought. What we should do is reverse the process when designing IoT objects. Start with the questions people want to see answered. Then determine what data is needed and assemble the appropriate set of sensors to pick up that information. And only then get to the design of the physical object that accommodates those IoT functions. For instance, leaving space for the sensors and the cabling to fit the sensor network within your object. What I would really like to see is design tools that prompt the designer to think about what questions the object should answer. And then that should lead to the automatic generation of a proposed set of sensors to be included in the design." ⏮

(191141-01)

welcome in your ONLINE STORE

EDITOR'S CHOICE



Andonstar AD407 Digital Microscope with 7" Display

With its large 7" colour LCD display, the new Andonstar AD407 model is a stand-alone microscope. In the Elektor laboratory we used to add a large screen to the predecessor called ADSM302. The AD407 eliminates the need for an external screen, freeing up space on the workbench. This can be sufficient reason to give the AD407 a clear preference if and when a new device is going to be purchased. Our budget allowing, we'd replace the existing ADSM302 USB microscope with the new AD407.



Luc Lemmens
(Elektor Labs)

www.elektor.com/ad407-hdmi-digital-microscope-with-7-lcd-screen

Elektor Bestsellers

1. Learning Python with Raspberry Pi
   www.elektor.com/19106



2. Raspberry Pi 4 B
   www.elektor.com/rpi4
3. The Ultimate Compendium of Sensor Projects
   www.elektor.com/19103
4. Andonstar AD407 Microscope
   www.elektor.com/19079
5. Raspberry Pi Zero WH
   www.elektor.com/18567
6. Mendocino Motor AR O-8
   www.elektor.com/19129
7. Elektor SDR Shield 2.0
   www.elektor.com/18515

The Ultimate Compendium of Sensor Projects

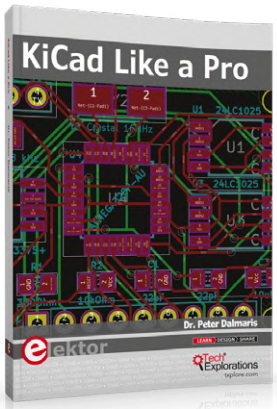


This book is about developing projects using the sensor modules with Arduino Uno, Raspberry Pi and ESP32 microcontroller development systems. More than 40 different sensor types are used in various projects described in the book. The book explains in simple terms how to implement the sensors in your project, using tested and fully working example projects.

Member Price: €31.46

www.elektor.com/19103

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.

Member Price: €35.96

www.elektor.com/18822

SDR Hands-on Book



Elektor's SDR-Shield (SKU 18515) is a versatile shortwave receiver up to 30 MHz. Using an Arduino and the appropriate software, radio stations, morse signals, SSB stations, and digital signals can be received. In this book, successful author and enthusiastic radio amateur, Burkhard Kainka describes the modern practice of software defined radio using the Elektor SDR Shield. He not only imparts a theoretical background but also explains numerous open source software tools.

Member Price: €26.96

www.elektor.com/18914

Learning Python with Raspberry Pi

NEW

This book teaches the Python programming language using the Raspberry Pi 4 computer. The book introduces the Raspberry Pi 4 hardware and then teaches Python using the topics: variables, strings, arrays, matrices, lists, dictionaries, user functions, flow of control, printing, keyboard input, graphics, GUI, object-oriented programming, and many more.

The book is aimed at beginners, students, practicing engineers, hobbyists, and anyone else wanting to learn Python programming. The book includes many example programs and case studies. All example programs and case studies have been fully tested by the author and can be proven to work.



Member Price: €31.46

www.elektor.com/19106

JOY-iT 3-in-1 Device (Oscilloscope + Signal Generator + Multimeter)



This practical and flexible 3-in-1 device combines the functions of an oscilloscope, a function generator and a multimeter. Two batteries type 18650 allow operation for about one day. The batteries are charged via a USB-C port which can also be used to operate the device during charging.

Member Price: €215.10

www.elektor.com/19157

Elektor ESP32 Smart Kit Bundle



Experimenting with the ESP32 was never easier! With the Elektor ESP32 Smart Kit Bundle, you'll absorb programming with the ESP32 IoT Microcontroller using the Arduino IDE and MicroPython programming languages. The kit consists of the highly popular ESP32 DevKitC development board, breadboard, sensors, LEDs, LCD, etc. for use in your experiments.

Member Price: €62.96

www.elektor.com/19033

Camera Projects Book



The book explains in simple terms and with tested and working example projects, how to configure and use a Raspberry Pi camera and USB based webcam in camera-based projects using a Raspberry Pi.

Member Price: €26.96

www.elektor.com/18943

Hexadoku The Original Elektorized Sudoku

Traditionally, the last page of Elektor Magazine is reserved for our puzzle with an electronics slant: welcome to Hexadoku! Find the solution in the gray boxes, submit it to us by email, and you automatically enter the prize draw for one of five Elektor book vouchers.

The Hexadoku puzzle employs numbers in the hexadecimal range 0 through F. In the diagram composed of 16 × 16 boxes, enter numbers such that **all** hexadecimal numbers 0 through F (that's 0-9 and A-F) occur once only in each row, once in each column and in each of the 4×4 boxes (marked by the thicker black lines). A number of clues are given in the puzzle and these determine the start situation.

Correct entries received enter a prize draw. All you need to do is send us **the numbers in the gray boxes**.



Solve Hexadoku and win!

Correct solutions received from the entire Elektor readership automatically enter a prize draw for five Elektor Book Vouchers worth **$60.00 / £45.00 / €50.00 each**, which should encourage all Elektor readers to participate.

Participate!

Ultimately January 27th 2020, supply your name, street address and the solution (the numbers in the gray boxes) by email to: **hexadoku@elektor.com**

Prize Winners

The solution of Hexadoku in edition 6/2019 (November & December) is: **DB072**.

The €50 / £40 / $70 book vouchers have been awarded to: Martin Bratko (Czechoslovakia), Volker Treiber (Germany), Arjen de Rijke (Netherlands), Jean-Pierre Demangeon (France) and Alexandr Papazyan (Russia).

Congratulations everyone!

E			A		3	6			0	8		F			D
B	5					D			9					A	1
8	0	6	7									9	C	E	3
9	E	A	1	6	0					5	7	4	B	D	2
2	B	7	F	9	D					3	4	0	E	6	5
C	6	4	0	1	A					D	E	3	F	7	8
5	3	8	D	E	2	4			F	0	6	1	9	C	A
3	F	1	E	7	5	C			B	9	0	A	D	2	4
6	7	0	5	2	8					4	A	C	3	F	9
4	2	C	B	D	9					1	3	6	0	8	E
A	D	9	8	3	E					C	F	5	1	B	7
7	9	F	4									8	5	1	0
D	8					9			5					4	F
0			2		B	3			D	6		E			C

3	C	9	6	7	D	1	5	B	E	2	8	0	F	4	A
0	A	7	D	2	4	B	6	1	9	C	F	3	E	8	5
5	B	4	F	E	8	9	3	D	0	7	A	1	6	C	2
8	E	1	2	F	A	C	0	3	4	6	5	B	7	9	D
4	3	5	9	1	E	6	F	0	8	D	B	2	A	7	C
6	F	A	E	D	B	0	7	2	3	4	C	8	9	5	1
1	7	8	B	3	2	4	C	E	A	5	9	F	D	0	6
C	D	2	0	5	9	8	A	F	6	1	7	E	B	3	4
9	8	B	C	4	0	E	2	7	5	A	6	D	1	F	3
D	5	3	A	6	C	F	9	4	1	E	2	7	8	B	0
7	0	E	4	A	1	3	B	8	C	F	D	5	2	6	9
F	2	6	1	8	5	7	D	9	B	0	3	4	C	A	E
A	1	D	5	9	F	2	8	C	7	3	4	6	0	E	B
B	4	0	7	C	6	A	E	5	2	8	1	9	3	D	F
E	6	F	8	B	3	5	1	A	D	9	0	C	4	2	7
2	9	C	3	0	7	D	4	6	F	B	E	A	5	1	8

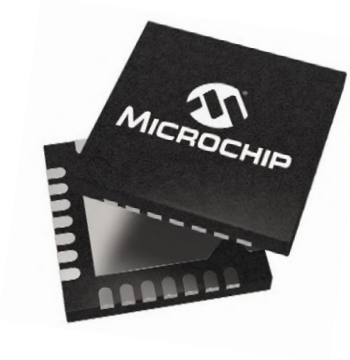
The competition is not open to employees of Elektor International Media, its subsidiaries, licensees and/or associated publishing houses.



Make the Connection

Connecting to the World, With and Without Wires

You face enough challenges in your day. Microchip understands that, so we make adding connectivity to your design easy. Whether you need a robust and reliable wired connection or the mobility and convenience of wireless, Microchip's broad portfolio will help you make the connection.



For added ease, our MCUs and MPUs are designed to be compatible with our wired and wireless devices. And we can help you get to market quickly with certified modules and production-ready protocol stacks.

Connect with Microchip and learn how to securely connect to the world around you.

SMART | CONNECTED | SECURE

Make the connection at
www.microchip.com/Connected

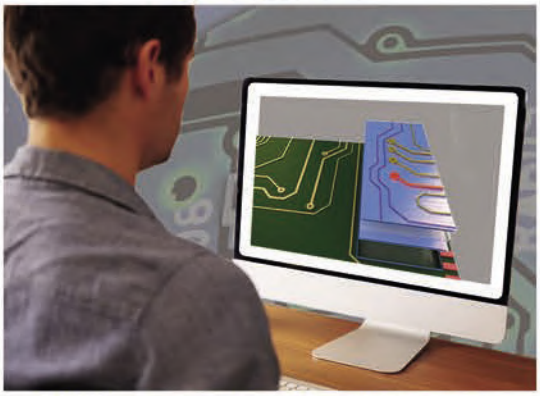
MICROCHIP

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. DS00002768A. MEC2231Eng11/18

PROTEUS DESIGN SUITE

Advanced PCB features for professional board design

DESIGN ROOMS



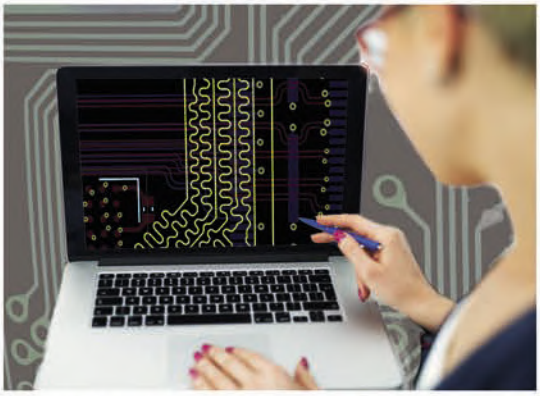
Set design rules that apply in user specified areas of the PCB.

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