Edition 5/2019 | September & October 2019 | www.elektormagazine.com

elektorlabs magazine

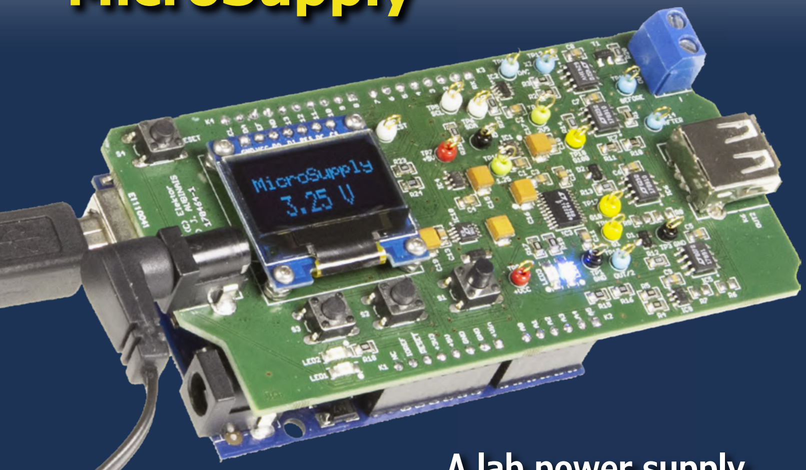
DESIGN ⊳ BUILD ⊳ SELL ELECTRONICS

Edition 5/2019 – September & October 2019

Grow Light for Horticulture Box

with state-of-the-art LED driver

MicroSupply



A lab power supply for connected devices

DDS Signal Generator

Lots of signal for little money

reichelt elektronik – The best part of your project

✓More than 110,000 products
✓Top price/performance ratio
✓High availability and reliable delivery
✓Strong brands and choice quality

DELOCK® – THE STRONG BRAND FOR IT TECHNOLOGY, PERIPHERALS AND ACCESSORIES

DELOCK® we move the world

DisplayPort Splitter

Mirror or expand the monitor signal to 3 displays in 4K resolution!

- Dual Mode capable (DP++)
- DisplayPort 1.4 Multi-Stream Transport
- Supports HDR
- Supports HDCP 1.4 and 2.2

Order no.: DELOCK 87737

TECHNOLOGY TIP £82.70 (£ 68.92)

(Delivery without monitors)

LoRa antenna, 868 MHz

DELOCK® we move the world

SMA connector

- Frequency range: 860 - 870 MHz
- ZigBee 868 MHz, Z-Wave 868 MHz
- Antenna gain: 3 dBi
- Impedance: 50 Ohm
- Efficiency: 45.91
- VSWR: 1.5

OMNIDIRECTIONAL, WITH TILTING JOINT

Order no.: DELOCK 89769

£8.65 (£ 7.21) BEST SELLER

LoRa antenna, 863-928 MHz

DELOCK® we move the world

MHF connector

- LoRa, NB-IoT, ZigBee, Z-Wave (908, 921), ISM, GSM
- Antenna gain: 1.68 dBi
- Impedance: 50 Ohm
- Adhesive mounting
- Cable type: 1.13

Order no.: DELOCK 12540

£6.51 (£ 5.43) BEST SELLER

NEWSLETTER SUBSCRIBE NOW & BENEFIT!

Always be the first to know - Top offers, interesting topics, actions and novelties

REGISTER NOW ▶ http://rch.lt/v3

Daily prices! Price as of: 1. 8. 2019

Payment Methods: VISA MasterCard Maestro PayPal

www.reichelt.co.uk

ORDER HOTLINE: +44 203 808 95 25

Elektorlabs Magazine

Edition 5/2019
Volume 45, No. 497
September & October 2019

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

www.elektor.com
www.elektormagazine.com

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

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

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

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

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

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

Ten Commandments of Electronics

1. Beware the lightning that lurketh in an undischarged capacitor, lest it cause thee to be bounced upon thy buttocks in a most ungentlemanly manner.
2. Cause thou the switch that supplies large quantities of juice to be opened and thusly tagged, so thy days may be only on this earthly vale of tears.
3. Prove to thyself that all circuits that radiateth and upon which thou worketh are grounded, less they lift thee to high frequency potential and cause thee to radiate also.
4. Take care thou useth the proper method when thou taketh the measure of high voltage circuits so that thou doth not incinerate both thee and the meter; for verily, thou hast no account number and can easily be replaced, the meter doth have one, and as a consequence, bringeth much woe unto CEO, Accounts & the Supply Department.
5. Tarry not amongst those who engage in intentional shocks, for they are not long for this world.
6. Take care thou tampereth not with interlocks and safety devices, for this will incur the wrath of thy Seniors and bringeth the fury of the Safety Officer down about thy head and shoulders.
7. Work thou not on energised equipment, for if you doth, thy buddies will surely be buying beers for thy widow and consoling her in other ways not generally accepted by thee.
8. Verily, verily I say unto thee, never service high voltage equipment alone, for electric cooking is a slothful process and thy might sizzle in thine own fat for hours on end before thy Maker sees fit to end thy misery and drag thee into His fold.
9. Trifle thou not with radioactive tubes and substances, lest thou commence to glow in the dark like a lightning bug, and thy wife be frustrated nightly and have no further use for thee except thy wage.
10. Commit thou to memory the works of the Prophets, which are written in the Instruction Books, which giveth the straight dope and which consoleth thee, and thou cannot make mistakes — yeah, well, sometimes, maybe.

(unknown author; acknowledged)

Jan Buiting, Editor-in-Chief

The Circuit

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

This Edition

Volume 46 – Edition 5/2019

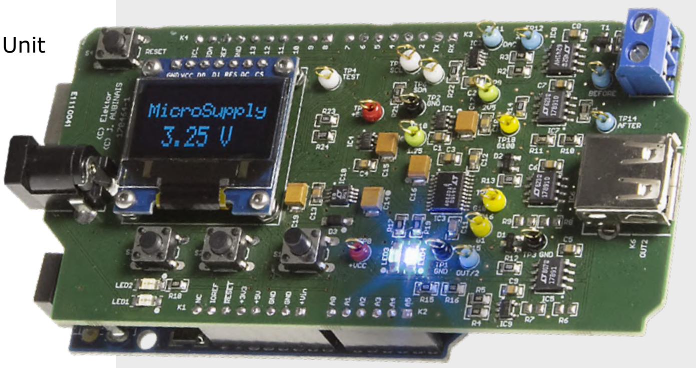
No. 497 September & October 2019

Regulars

MicroSupply

A lab power supply for connected devices

How much power does my connected device use? Does its sleep mode conform to the manufacturer's documentation? How long will the CR2032 battery last? If you're asking yourself these sorts of questions, here is a regulated power supply, dubbed MicroSupply, in the form of an Arduino shield, which measures very small currents. Thanks to a link with a PC application, you can see and log the power consumption of your device.



92

Features

Projects

USB/RS232 Breakout Board
... a.k.a. BoB

76

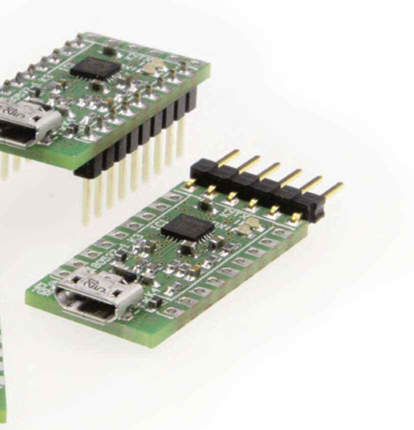
elektorlabs magazine

JOY-iT JDS6600 DDS Signal Generator

lots of signal for little money

In the past, every electronics technician in his (hobby) lab had a function generator with an XR2206, but those days are long gone. Nowadays such a generator works with a DDS chip that generates frequencies digitally. Unfortunately, a decent DDS function generator isn't really cheap — or is it? The JOY-iT DDS function generator promises a wide frequency range and many possibilities at a low price.

106



Next Editions

ElektorLabs Magazine Edition 6/2019

Build Your Own RFID Reader/Writer • TMC2160 Motor Driver Board • Elektor's First Video Games Console • 8-Channel Timer with O-LED Display • UDP Packet Communication • SV3ORA Broadband Radio Receiver • Improved Radiation Meter • FReRTOS fpr ESP32 • JesFs Flash File System for IoT SigFox Projects • RPI in Automation • DIY CPU (5) • Softstart Unit for Audio Amplifier • IoT Relay on ESP8266 • Fun with Infineon Dev Boards.

ElektorLabs Magazine edition edition 6/2019 covering November & December 2019 is published on 22 October 2019. Delivery of printed copies to Elektor Gold Members is subject to transport. Contents and article titles subject to change.

Elektor Industry Edition 3/2019

Elektor Industry issue 3/2019 is a special edition for the **Productronica 2019 trade show in Munich, Germany (November 12-15, 2019)**, and will focus on a range of innovative technologies, research, hardware and software, including IoT, 5G, sensors, CAD, SMT, PCB manufacturing, and more. Plus, you'll find fresh instalments of the magazine's regulars like Infographics and Industry Store.

Elektor Industry issue 3/2019 will be published early November 2019 to Elektor Gold members in print and to Elektor Green members as a PDF download. The edition will be available for purchase at www.elektormagazine.com.

LABS PROJECT

MQTT Sensor Hub

Take sensor readings and transmit them using an ESP32-PICO-KIT

By **Mathias Claussen** (Elektor Labs)

One of the main projects we published in a recent issue was the Monster LED Clock, featuring huge seven-segment displays. These were not just able to show the time of day: it was also possible to use them to display sensor values received over a wireless network using MQTT. In this article we look at some alternative firmware that allows the direct connection of sensors to the ESP32 in the clock. The data are sent using MQTT to a central broker, which can be implemented using a Raspberry Pi running Mosquitto and Node-RED. The firmware for this MQTT sensor hub was built with the help of software modules recycled from various projects, which saved a lot of development time.

The best thing about developing software in a modular fashion is that it is possible to reuse ready-tested parts from other projects. That saves not only development time, but also reduces the effort involved in testing and documenting the software. Mathias Claussen at Elektor Labs has been working on a second version of the Elektor ESP32-based weather station published in the January & February 2019 issue [1]. Feature requests have already been flooding in from readers using the first version, and firmware updates and other advances can be found at [2]. Among the suggestions was the ability to connect the following I²C sensors directly to the ESP32.

- BMP280 atmospheric pressure, temperature and humidity sensor [5]
- VEML6070 UV sensor [6]
- TSL2561 light sensor [7]

Since the Monster LED Clock project was based on the ESP32-PICO-KIT, it was relatively easy to integrate the required firmware to drive these sensors (as well as for the DHT11 humidity and temperature sensor used in our Air Pollution Monitor project [3]) into the code for this project. The parts of the MQTT data transport code concerned with delivering sensor values to an MQTT broker were also reused from the weather station project.
Since it is onerous to have to take the clock down from the wall every time you want to apply a firmware update, we have also implemented the facility to update the ESP32 over WLAN using an Arduino OTA library (see the **text box** 'OTA programming using an Arduino').

PROJECT DECODER

ESP32
MQTT
seven-segment display

entry level
intermediate level
➡ expert level

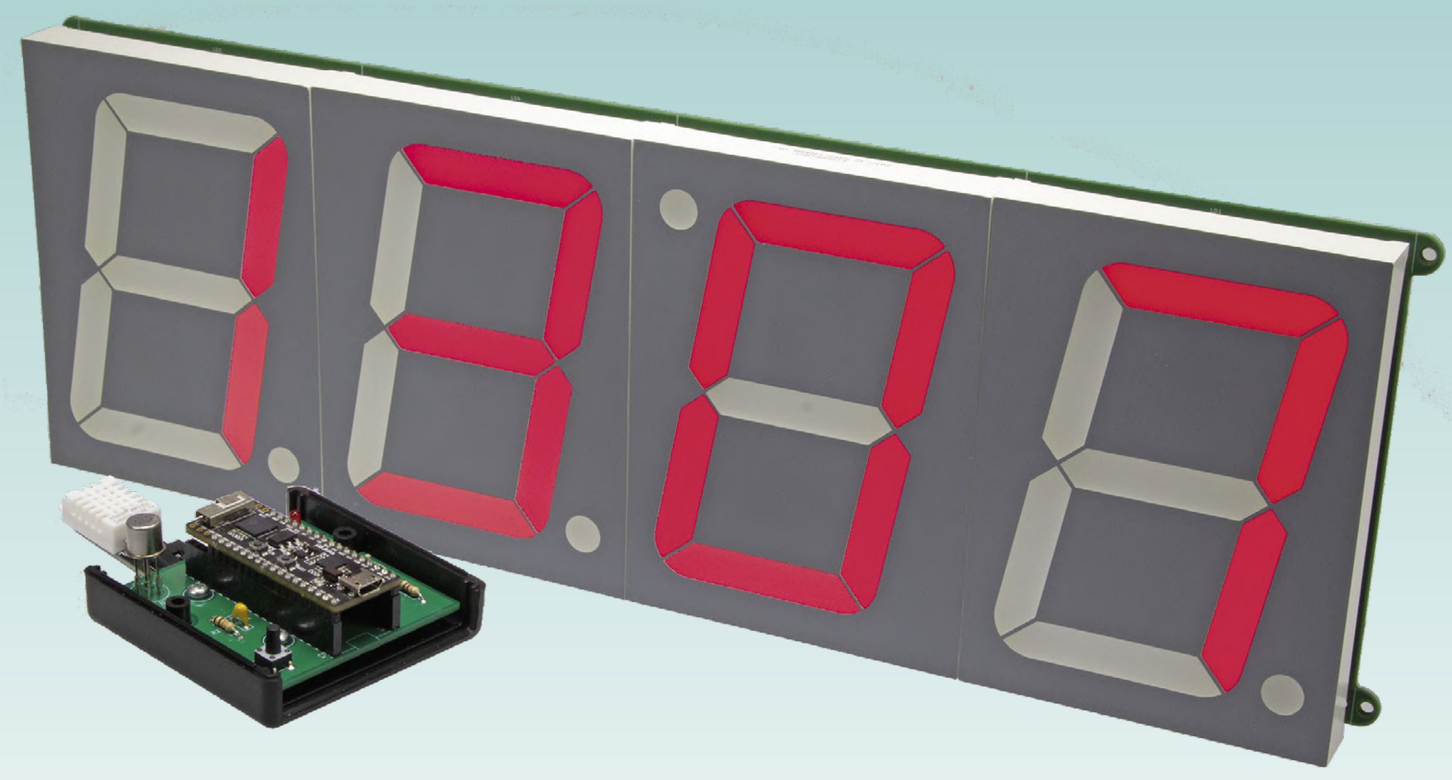
4 hours approx.

soldering iron

€100 / £90 / $120 approx.

Sensor connection

No dedicated connectors are provided to connect the sensors, but the I/O pins of the ESP32 that are not needed to drive the display are all brought out on the underside of the board (see **Figure 1**) and it is possible to solder header plugs or wires directly to these points. The I²C pins are labelled, and IO15 is used as a data line for the DHT11 or DHT22 device. A special feature of the ESP32 is its I/O matrix: whereas in the case of microcontrollers such as those in the ATmega series there is a fixed allocation of functions to pins, with the result that sometimes conflicts can arise between different functions requiring the same pins, the ESP32 arranges matters more conveniently. The I/O matrix allows the essentially arbitrary allocation of pins to functions.
It is therefore possible to use the I²C port on the clock alongside a UART by allocating unused pins to it. You could equally easily also add an audio codec connected to the I²S interface, or use an SPI bus to expand the connection possibilities without problems. Configuring the matrix can be done using the ESP32 Arduino core libraries by simply specifying in the code which pins are to be used for which purposes. An example of how to do this appears in our 'ESP32 as a



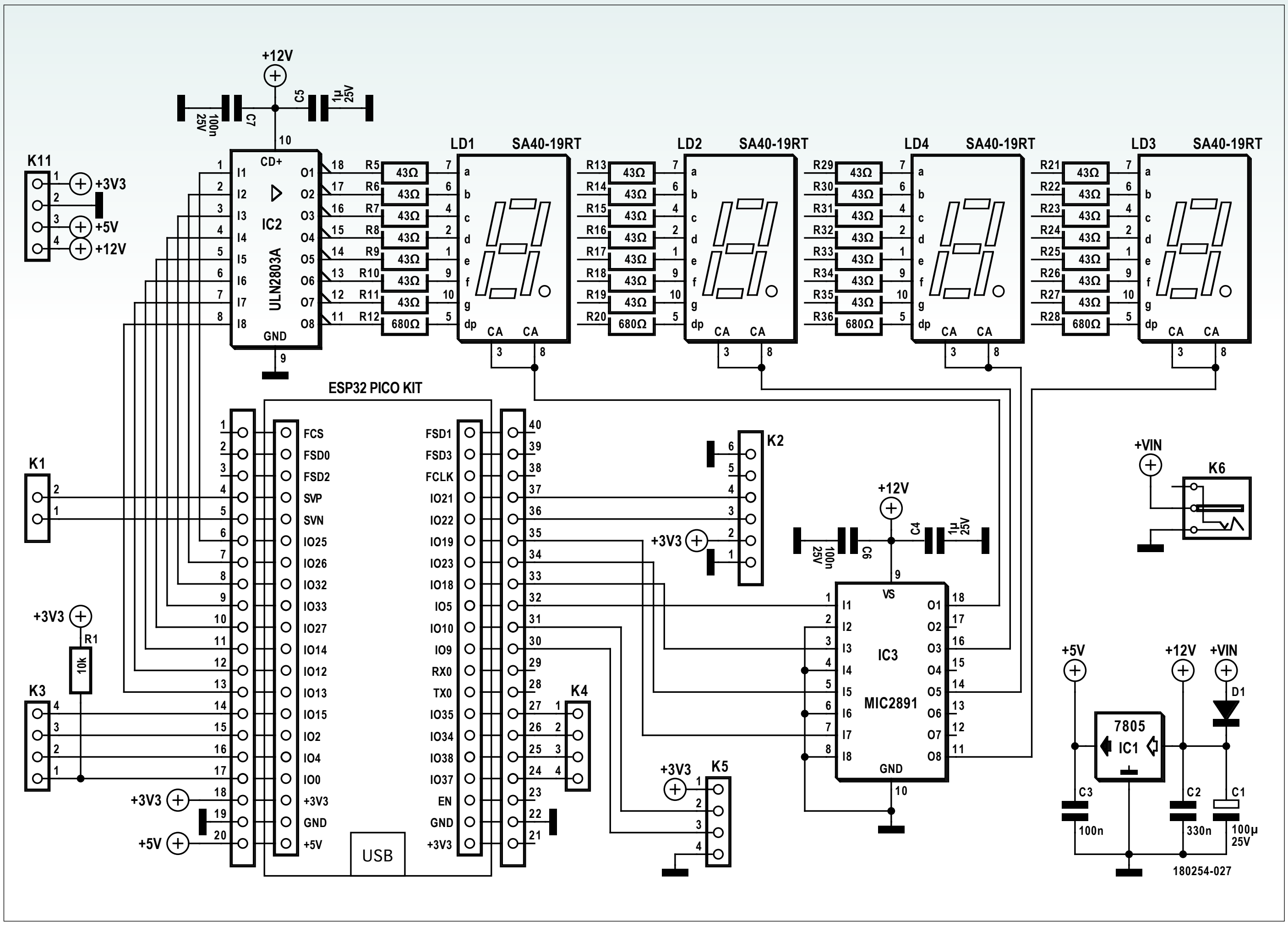
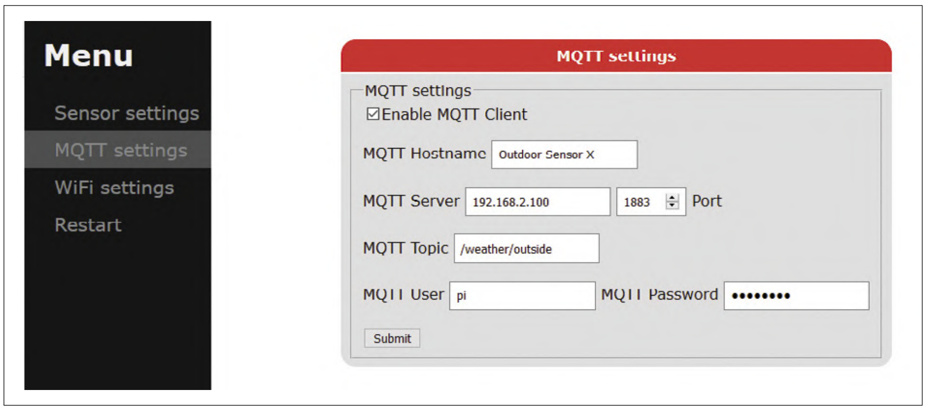


Figure 1. Circuit of the Monster LED Clock showing the available pins on the ESP32.



MQTT settings
☑Enable MQTT Client
MQTT Hostname: Outdoor Sensor X
MQTT Server: 192.168.2.100 Port: 1883
MQTT Topic: /weather/outside
MQTT User: pi MQTT Password: ••••••••

Figure 2. Configurable MQTT settings.

Time Server' project [8]. The OLED display used in that design was connected using the ESP32's GPIO pins 5 and 4 for the SDA and SCL signals. The command

```
wire.begin(5,4,100000);
```

selects these pins using the first two parameters, while the third sets the desired I²C bus frequency. The ESP32's UART can be configured in a similar way.

A disadvantage that comes with this freedom is an increase in propagation delay times and a reduction in the highest usable signal frequencies. The matrix hardware is limited to 40 MHz, and signals faster than that cannot be routed through it. And, because of increased latencies, at frequencies of over 20 MHz SPI and I²C data signals can experience delays relative to the clock large enough to cause bit errors.

MQTT delivery

We will now look at what the sensor hub will send to the new version of the Monster LED Clock using MQTT. The message takes the form of a JSON string with the following structure.

```
"{"data":
   {"temperature":31.1,
   "humidity":25.9,
   "airpressure":0,
   "PM2_5":-1,
   "PM10":-1,
   "Lux":-1,
   "UV":-1}
}"
```

This `data` object includes the measured readings. Real values are present if the sensor is recognized at start-up, and a default value is used otherwise. For `temperature`, `humidity` and `airpressure` the default value is zero, while for the other readings the default is –1. All temperatures are represented in °C.

As in the first version of the Monster LED Clock firmware, the server (MQTT broker) and the topic of the transmitted message can be set up using a configuration page served up by the ESP32 web server (see **Figure 2**). The page at [4] describes how to set up a Raspberry Pi as an MQTT broker. That site also explains how to install Node-RED [9], which makes implementing simple applications that process and display sensor data a quick job.

Software components

Having looked at the 'what' of our design, we can now turn to the 'how'. Anyone who writes software will appreciate how undesirable it is to reinvent the wheel. If you find yourself always working on the same platform (one particular microcontroller family, for example) you will almost subconsciously have built up a collection of routines, libraries and modules that you use all the time; and that is certainly the case given the large number of ESP32 projects encountered in Elektor Labs. In essence network-based projects such as the weather station or the LED clock have the same core, comprising a web server, the facility to configure the WLAN using a browser, and the other configuration routines. That means we do not need to develop these components from scratch every time.

Ticker

In the case of the clock these core components were augmented by code bor-

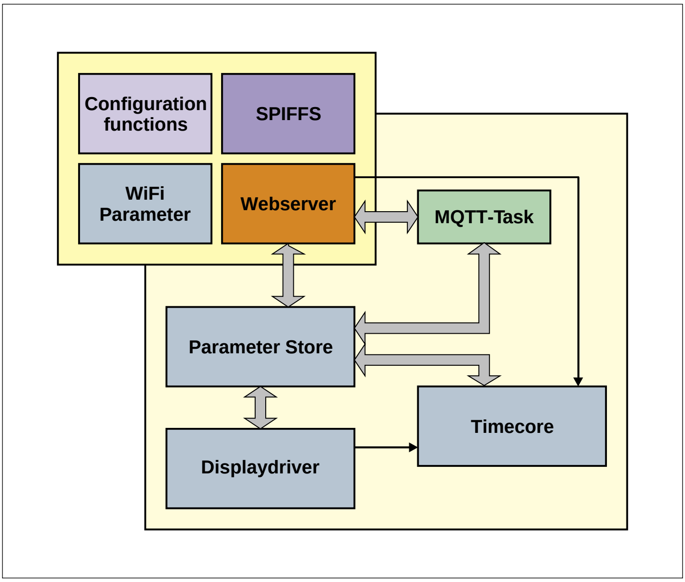


Figure 3. The software modules that are used.

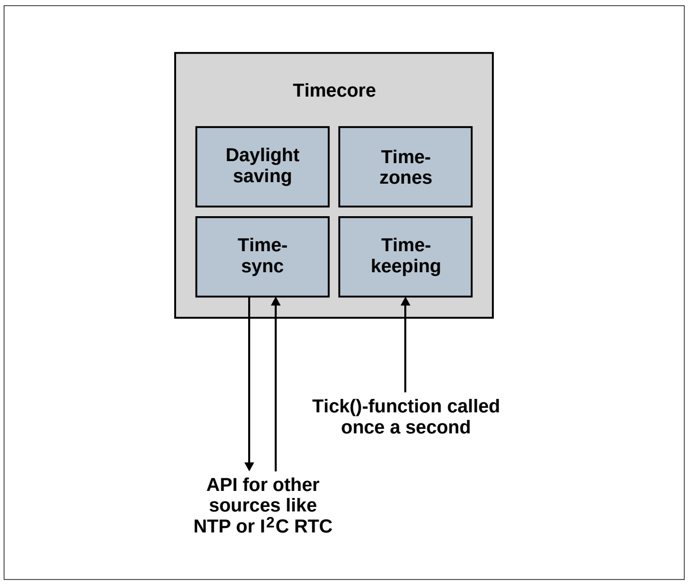

Timecore
Daylight saving
Time-zones
Time-sync
Time-keeping
Tick()-function called once a second
API for other sources like NTP or I²C RTC

Figure 4. TimeCore functions.

rowed from the ‘Pinball Clock’ project [10] to handle timekeeping, synchronization, timezones and the summertime/wintertime changeover. Internally this code is called ‘TimeCore’ and is used in various projects. **Figure 3** shows an overview of this software module.
The TimeCore library includes its own NTP client, and at run-time can also be connected to another time source such as an I²C real-time clock. The only hard requirement is that every second a certain function in the TimeCore library must be called in order to increment the internal seconds counter (see **Figure 4**).

For this purpose we use the ‘Ticker’ library, which offers a simple way to arrange for a function to be executed periodically. The relevant code is as follows.

```
#include
...
Ticker OneSecondTick;
...
OneSecondTick.attach_ms(1000, FNC);
```

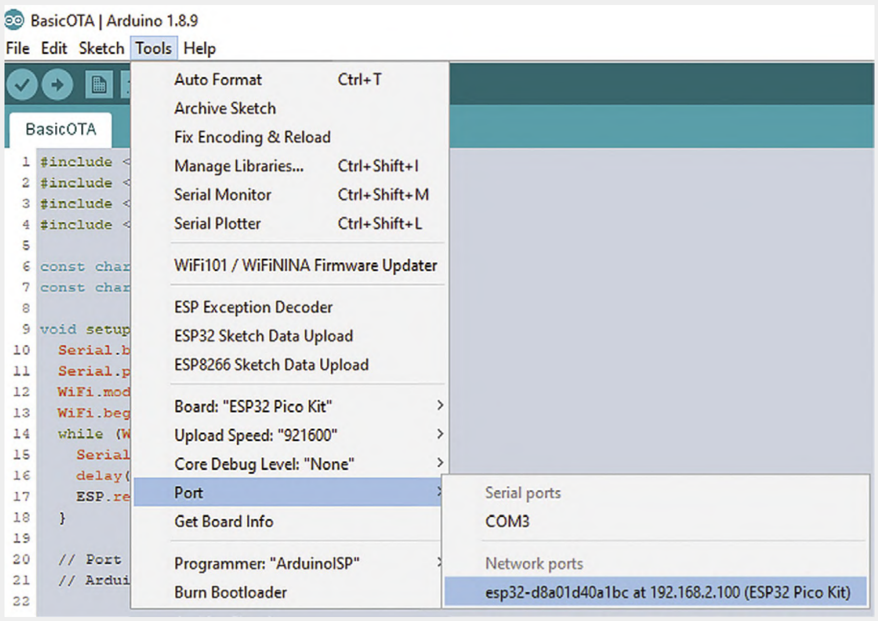
Here the ‘FNC’ is a placeholder for the function to be called periodically. This function must be declared according to the prototype `void fnc(void)`, that is, with no arguments and no return value. The first parameter to the function `attach_ms()` is the desired time between calls in milliseconds; the function can be used to call other functions at different rates for other purposes. It is worth bearing in mind that underneath, the ESP32 is running the FreeRTOS [11] operating system, but this is hidden from the programmer by the Arduino IDE. When using `Ticker` a new task is created which calls the specified function at the given time interval at a very low priority, rather than on a high-priority interrupt. That means that the call to the function can be delayed by another task running at higher priority.

Tasks

MQTT plays a particularly significant part in the code. It would be possible to pack all the MQTT-related functions into the main `loop()` function, but this makes the code harder to understand. Instead it is better to create a dedicated task: when there are no MQTT operations to carry out the task can be put to sleep until a new set of readings arrives. When the MQTT task connects to a server the process can take several seconds, and there is also the particular problem of the sever causing a timeout while attempting to create a connection. If the connection logic is placed in a separate task then in the ideal case FreeRTOS can use this spare time to carry out other tasks, for example in the main `loop()` function. In any case the task will have to know when new readings have become available. This is easy to implement in FreeRTOS as each task can receive what are called ‘notifications’ (see **Figure 5**), as long as the notifying task knows the

OTA programming using an Arduino

An ESP32 can be loaded with new firmware over its serial port. This certainly does the job, although it is a somewhat spartan approach. Much more developer-friendly is to use OTA (‘Over The Air’) programming. To set this up it is necessary first to load firmware with OTA capabilities into the flash memory. A minimal example can be found in the IDE under File -> Examples -> ArduinoOTA -> BasicOTA. If you add this component to your own firmware, and if you configure the firewall of the PC being used for development appropriately, then a new network port will be created which you can select from within the IDE (see **Figure**).
When new firmware is to be written the upload sketch writes the data to a special OTA partition of the flash memory. This is the reason that on many boards only a maximum of half the total flash memory storage provided in hardware is available for applications to use. As well as the firmware, we also need to set up the contents of the SPIFFS (SPI flash file system), which is normally used for holding pages for the web server. And now, because the OTA functionality is itself also part of the newly-uploaded firmware, it can be used to install future versions as well (at least as long as the OTA component can run without crashing).



```
#include <
#include <
#include <
#include <

const char
const char

void setup
  Serial.b
  Serial.p
  WiFi.mod
  WiFi.beg
  while (W
    Serial
    delay(
    ESP.re
  }

  // Port
  // Ardui
```

In order to make the port visible in the Arduino IDE, the ESP uses `mDNS` to send information in the background about its OTA capabilities and associated IP address to all other devices on the same network segment. At the start of an update UDP packets are exchanged and when necessary a password is requested. If no password has been configured any device on the local network can initiate an update, which is not without attached risks; but if a password is set then the SPIFFS can no longer be updated using the Arduino IDE. So when deciding how to implement OTA updates, it is necessary to carefully weigh up the convenience of password-free access against the possibility of some hacker or other nefarious prankster uploading unwanted firmware to your device: you may wish to consider additional means of preventing unwanted updates.

'handle' of the task to be notified. In the function `xTaskCreatePinnedToCore()` the sixth parameter is a pointer to a variable of type `TaskHandle_t`.

```
xTaskCreatePinnedToCore(
   MQTT_Task,
   /* Function to implement the
   task */
   "MQTT_Task",
   /* Name of the task */
   10000,
   /* Stack size in words */
   NULL,
   /* Task input parameter */
   1,
   /* Priority of the task */
   &MQTTTaskHandle,
   /* Task handle */
   0);
   /* Core to pin to */
```

If the task is successfully created the variable pointed to by the pointer will be set to the value of the new task's handle. This can be used by other functions to send notifications to the task.
Equally, a task can wait for notifications. In the MQTT task this is accomplished using the following line.

```
ulNotificationValue =
   ulTaskNotifyTake(pdTRUE,
   portMAX_DELAY);
```

The first parameter specifies that after reading all bits in the notification word will be cleared, and the second parameter gives the maximum waiting time for a notification (here infinite). The return value is a 32-bit word whose bits can be set using the function `mqttsettings_update()` in webserverfunctions.cpp as follows.

```
if(MQTTTaskHandle != NULL) {
   xTaskNotify(MQTTTaskHandle,
   0x01, eSetBits);
}
```

This code fragment first checks whether the handle is valid. Then bit 0 in in the notification word of the MQTT task is set. If the MQTT task is sleeping, then it will be woken up and its execution will proceed. This notification is used to signal to the MQTT task that the parameters that are to be transferred have changed; the task can then read these new parameter values and process them.
Further description of FreeRTOS and the inter-task communications facilities it offers is beyond the scope of this article; we are, however, planning an article for our next issue which will cover the possibilities opened up by task programming under FreeRTOS in more detail.

Modular display control

The most important job as far as the display is concerned is driving the seven-segment LEDs. Again we take a modular approach so that the code we write can be used in other projects. One point of difference in comparison to the original version is that we must now supply a hardware version identifier as a parameter. As we announced in the article describing the Monster LED Clock, a 'mini' version is on the way, and at Elektor Labs we are currently still working on the printed circuit board. For that reason the driver is designed to be ready for possible changes to the board layout, and hence will be usable not only as part of the clock project but also in your own designs. We will now look at a couple of hints and tips that will be of interest to readers wanting to use the code with their own hardware.

The function `SevenSegmentSetup(BedroomclockHW_t HW)` populates the array `DisplaySegmentPins` with the numbers of the pins connected to the LED segment lines from A through G and the decimal point. In sevensegments.cpp we find the following line.

```
volatile int DisplaySegmentPins[]= ;
```

The first seven values here are the pin numbers for the segment lines from A through G and the last entry is for the decimal point. If a different wiring of pins to segments is used then these values can be modified accordingly. The same goes for the digit common lines of the LED displays. The following line defines an array containing the pin numbers corresponding to digits 0 to 3.

```
volatile int DisplayCommonPins[]=
```

When calling the function `SevenSegmentSetup()` it is necessary to specify which hardware configuration is in use. The correct pin values for the hardware will then be written to the two arrays described above. And because we will be working with a timer interrupt the data must be transferred from the microcontroller's flash memory where they are stored into its RAM.
The reason for the above is as follows: because we want accurate timing for the multiplexing of the displays, we use one of the ESP32's timer modules. This triggers the code that sets up the segment patterns at a rate of 400 Hz. In the case of an AVR or Cortex-M series microcontroller, there is no difficulty fetching the required values from flash storage. In the case of the ESP32, however, we have to be more careful as the microcontroller does not have built-in flash, just a boot ROM and some RAM. Firmware is always stored in an external QSPI flash chip. Unfortunately the read speed of these devices is limited, and so the microcontroller has an intermediate cache memory to hold recently used (and potentially soon-to-be-used) data. The cache allows code to be executed faster than directly from the external flash. However, in an interrupt routine it can happen that the required code will not be found in the cache and must be fetched from flash. This can slow things down considerably, and if other operations (in particular writes to the flash cells) are taking place at the same time, fetching the required code will be held up even more. This is bad news for the predictability of interrupt timing. And worse, if a read operation collides directly with a write operation, an exception is thrown and the ESP32 will reset itself. We must therefore ensure that all the data that will be required by the interrupt routine are stored in RAM. We declare the function called by the timer interrupt using `IRAM_ATTR()`, which tells the compiler and linker to arrange the code so that at program start-up it is copied into RAM and executed from there. This approach completely avoids any problems associated with accesses to the external flash.

The question remains as to exactly where the interrupt is executed. The easy answer 'on the microcontroller' prompts the response 'but where on the microcontroller?'. There are two cores in the ESP32 and the basic answer is that an interrupt runs on the core on which it was set up. If you are interested in finding out more about the internal workings of the ESP32, then you can take a look at the ESP-IDF documentation [12].
We connected a type DHT11 temperature and humidity sensor to our prototype at Elektor Labs. A hitch that manifested itself with the library that we used

is that it disables interrupts briefly on the core on which the sensor data readout code is being executed. During the development of the software it appeared that every time the sensor was read, the timer interrupt handling, the multiplexing of the display, and the task for displaying the time or temperature got out of step with one another. A simple fix for this problem was to move the MQTT task from core 1 to core 0: then the interrupt code ran smoothly and there was no interference between the tasks.

Conclusion

In this article we have seen an example of a firmware upgrade demonstrating how using ready-made libraries can make software development more efficient. The key point here is the reusability of code: even if at the outset this might involve a little more effort, the work more than repays itself in making future projects easier. But even with the best libraries and software recycling practices there is sometimes no substitute for a look under the hood. The Arduino framework and the ESP-IDF manage a lot of the heavy lifting, but conceal many details that can sometimes be very important to understand. Nasty traps for the unwary and missing functions often lead to hours of fruitless debugging and are often only revealed in the fine print in the data sheet or framework documentation: assuming you are lucky enough to spot it!

If you have already built the Monster LED Clock you can install the new firmware to access many new features, transforming it from a humble clock into an MQTT sensor hub. **Figure 6** shows the printed circuit board of the clock with the DHT11 sensor connected directly to its reverse side. Because of the flexible I/O configuration it is also easy to implement further expansion: for example, you could independently add a type DS18B20 one-wire thermometer, or experiment with remote sensors using a LoRa module: the possibilities are endless!

180254-B-02

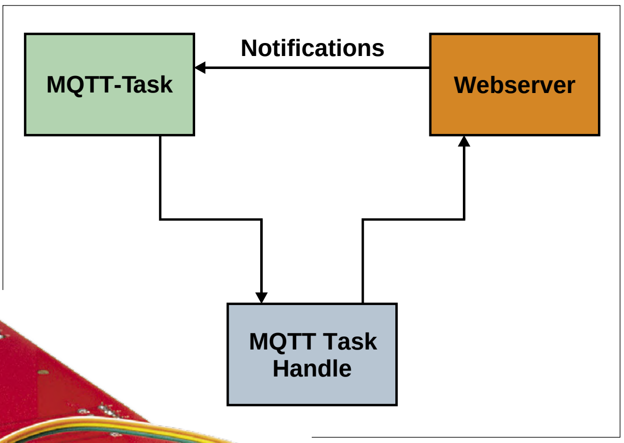


Figure 5. Communication between tasks using notifications.

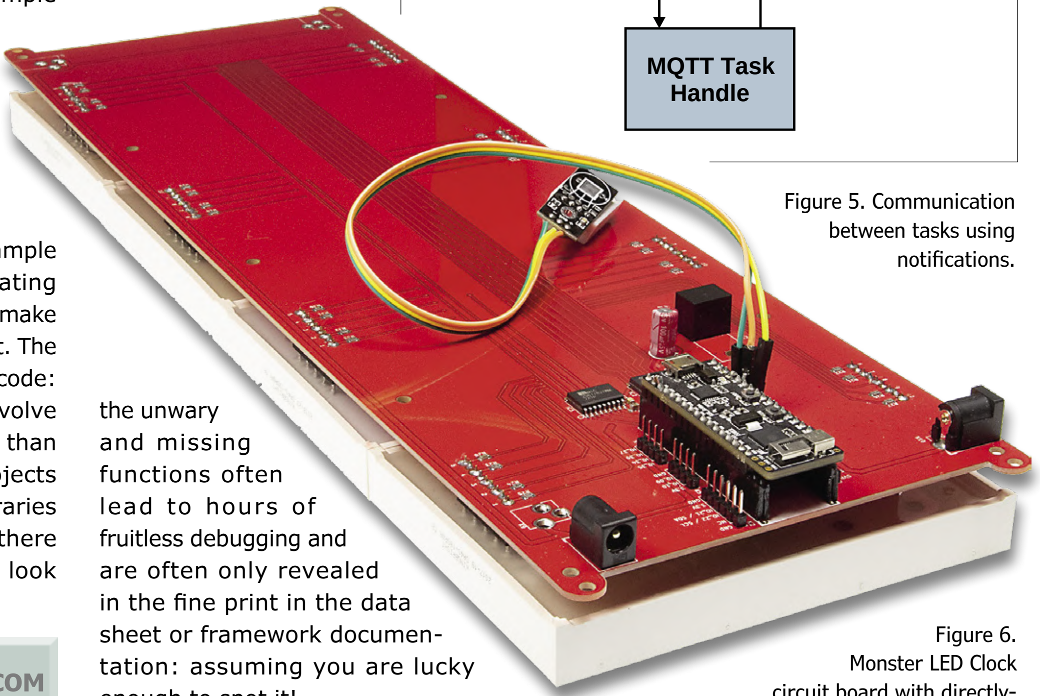


Figure 6. Monster LED Clock circuit board with directly-connected sensor.

@ WWW.ELEKTOR.COM

→ Printed circuit board for Monster LED Clock
www.elektor.com/180254-1

→ ESP32-PICO-KIT V4
www.elektor.com/18423

→ Book: 'IoT-Programmierung mit Node-RED' (in German)
www.elektor.de/18357

Web Links

[1] ESP32 Weather Station: www.elektormagazine.com/magazine/elektor-70/42351
[2] ESP32 Weather Station at Elektor Labs: www.elektormagazine.com/labs/esp32-weather-station-180468
[3] Monster LED Clock: www.elektormagazine.com/magazine/elektor-96/42659
[4] Monster LED Clock at Elektor Labs: www.elektormagazine.com/labs/bedroom-clock-with-out-side-temperature-based-on-esp32
[5] BMP280: www.bosch-sensortec.com/bst/products/all_products/bmp280
[6] VEML6070: www.vishay.com/ppg?84277
[7] TSL2561: https://ams.com/tsl2561
[8] ESP32 as a Time Server: www.elektormagazine.com/magazine/elektor-100/50916
[9] Node-RED: https://nodered.org
[10] Beat the Elektor Pinball Clock!: www.elektormagazine.com/magazine/elektor-88/42431
[11] FreeRTOS: www.freertos.org
[12] ESP-IDF: https://docs.espressif.com/projects/esp-idf/en/latest/
[13] Air Pollution Monitor: www.elektormagazine.com/magazine/elektor-88/42430

Flicker-free LED Dimmer

Featuring constant current and high efficiency

By **Joost Waegebaert, BSc.** (Belgium)

At first glance, dimming LED lamps appears rather simple. Take a PWM output from any desired microcontroller, use this signal to drive a power MOSFET, and you're done. However, in this article we take a different approach – certainly not new, but often overlooked – that gives perfect, flicker-free LED dimming with high efficiency. The circuit is built with ordinary components, which also makes it suitable for educational purposes.

The idea of dimming LEDs is not new. Countless solutions are already available, including the one presented here — which is often overlooked, even though it has significant advantages compared to the usual PWM method.

The problem with PWM

Using a pulse-width modulated (PWM) current to control the brightness of a LED is a well-known method. A PWM signal is a pulse waveform with constant frequency but variable pulse width. The wider the pulse, the longer the LED on time and the higher the average luminous output [1] of the LED.

No current flows during the off time, so the LED does not generate any light in that interval. As a result, the LED flickers. If the PWM frequency is high enough, our eyes see the 'average' luminous output, but the interval when the LED does not generate any light is still there. This can be tiring for our eyes, and it also leads to other problems.

Flickering can become noticeable when you take a picture or record a video. Even if the illumination of the object is perfect, problems can arise if the camera is not exactly synchronised with the pulsed LEDs. If a frame is recorded during the off time of the LEDs, the result will be an underexposed or dark frame. Quickly moving objects can also seem to flicker due to the pulsed light.
With PWM, the LED current is switched off and on as fast as possible. This creates high peak currents, which can easily lead

to electromagnetic interference (EMI). For this reason, the LEDs should not be connected to the PWM driver over long wires, as otherwise the wires will start acting as a source of radiated interference.

An alternative approach

It is perfectly possible to use an ordinary buck converter to generate a defined current for a LED. In the circuit shown in **Figure 1**, switch S1 periodically allows a current to flow through L1 and D2. This particular arrangement has the disadvantage that switch S1, in the form of a MOSFET in the positive supply line, is more difficult to drive than a MOSFET that switches to ground. To ensure that the MOSFET conducts current with the least possible dissipation, a voltage higher than the supply voltage is required for the gate drive, and that makes the overall circuit more complicated.

This problem can be eliminated by rearranging the components to form a floating buck converter, as shown in **Figure 2**. It is called 'floating' because the load (D2) is not connected directly to ground. An ordinary n-channel MOSFET can be used for S1 in this arrangement. When S1 is closed, the current through L1 rises. This current is measured, and S1 is opened when the current reaches a defined level. When S1 is open, the current through L1 drops. S1 is closed again before the current reaches zero. In this arrangement, the current through L1 also flows through the LED (D2). This current is not intermittent – it is basically a triangle waveform as sketched in Figure 2. The brightness of the LED is still modulated by this ripple, but the modulation is several orders of magnitude less than with conventional PWM, so the flickering is almost completely eliminated. The same is true for the EMI radiated by the connecting leads of the LED, due to the difference between a low-amplitude triangle waveform and a high-amplitude pulse waveform.

The circuit

Figure 3 shows a practical implementation of the floating buck dimmer. To clearly illustrate the underlying principle, standard components are combined here to form a working circuit.

Comparator IC1A compares the current through the switch transistor T1 to a variable reference voltage on the wiper of potentiometer R7. When the power is switched on, T1 will conduct and the current through L1 will rise and generate a voltage drop over resistor R4. Once this

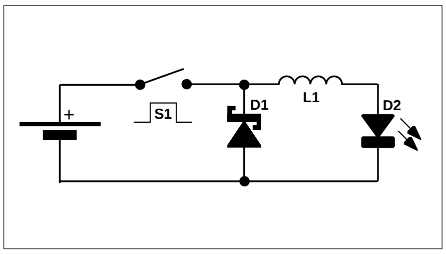


Figure 1. Basic circuit of a buck converter.

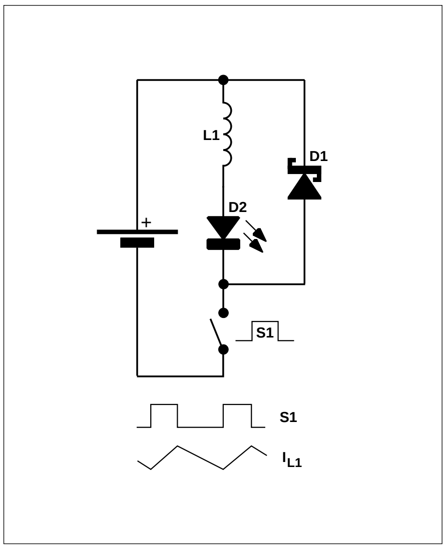


Figure 2. A floating buck converter.

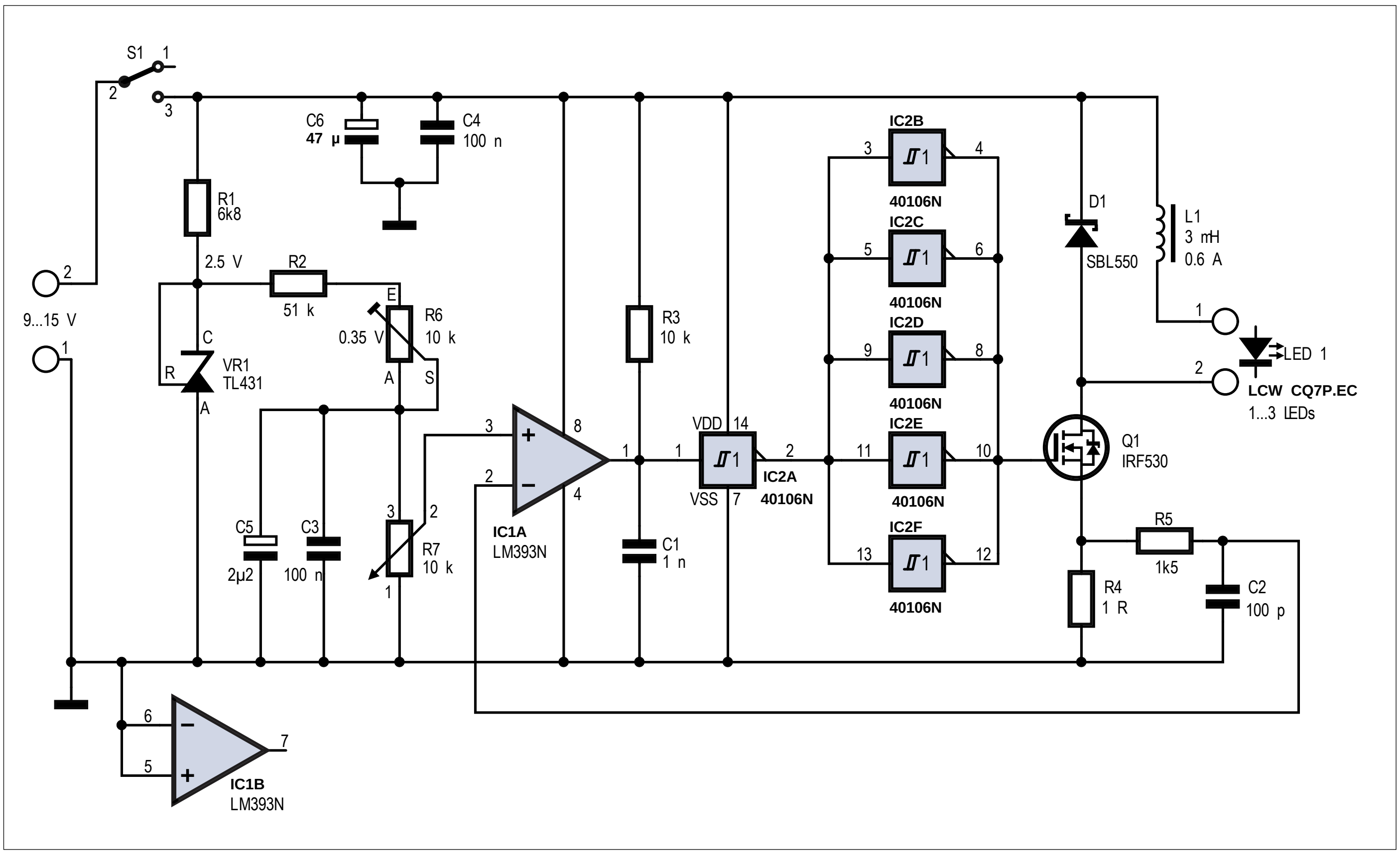


Figure 3. The full circuit diagram of the constant current LED dimmer.

A bit of math

The off time of the converter is determined by the time constant of R3/C1. The trigger level of the 40106 Schmitt trigger is approximately half the supply voltage. The duration of an RC charge cycle to approximately half the supply voltage is:

$t_{OFF} \approx 0.7 \times R3 \times C1 = 0.7 \times 104 \times 10^{-9} = 7\ \mu s$

The voltage across L1 with T1 switched on is given by:

$V_{L1\_ON} \approx V_{supply} - V_{LED} - I_{LED} \times R4 = 12 - 3.2 - 0.35 \times 1 = 8.45\ V$

The voltage across L1 with T1 switched off is given by:

$V_{L1\_OFF} \approx V_{LED} + V_{D1} = 3.2 + 0.5 = 3.7\ V$

The change in the current through L1 during the off time is:

$\Delta I_{L1} \approx V_{L1\_OFF} \times t_{OFF} / L1 = 3.7 \times 7 \times 10^{-6} / 0.003 = 8.6\ mA$

The visual flicker percentage FP [3] (assuming that the brightness (luminous output) of the LED is linearly proportional to the LED current) is:

$FP = (LO_{MAX} - LO_{MIN}) / (LO_{MAX} + LO_{MIN}) \approx (350 - 341.4) / (350 + 341.4) = 1.2\%$

If the LED is dimmed to 1/2 and 1/5 of the maximum power, this becomes:

$FP_{50} \approx (175 - 166.4) / (175 + 166.4) = 2.5\%$

$FP_{20} \approx (70 - 61.4) / (70 + 61.4) = 6.5\%$

The on time of the converter is:

$t_{ON} \approx L1 \times \Delta I_{L1} / V_{L1\_ON} = 0.003 \times 0.0086 / 8.45 = 3.1\ \mu s$

The converter frequency is:

$f = 1 / (t_{ON} + t_{OFF}) = 10^6 / (7 + 3.1) = 99\ kHz$

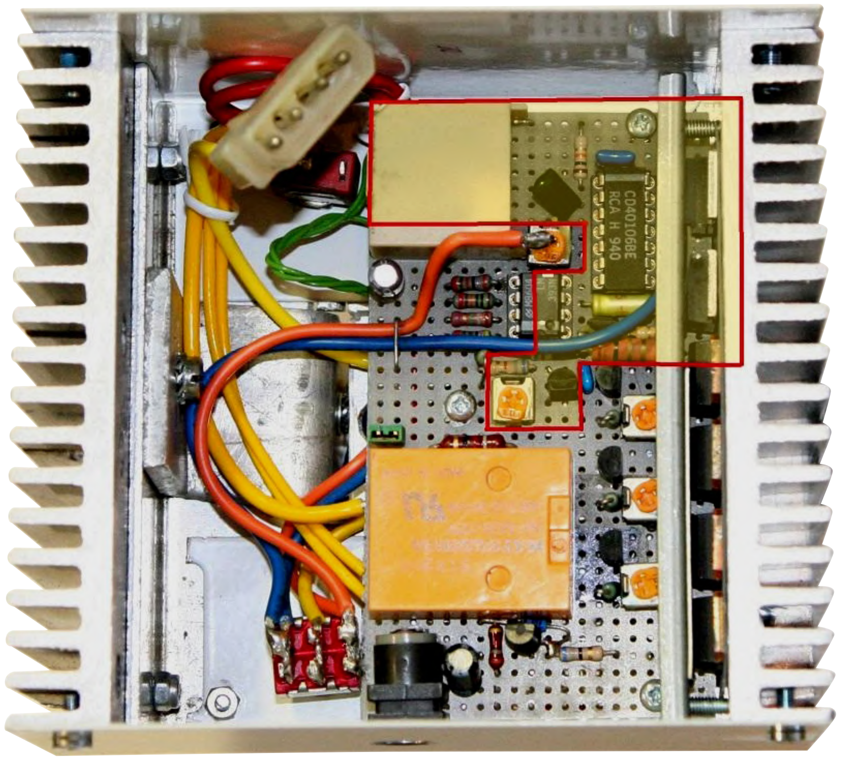


Figure 4. The section outlined in red is the dimmer built by the author as part of a circuit that powers two 1-watt LEDs from three lithium-ion cells. The heat sinks are part of an analogue solar cell charger

voltage is equal to the voltage on the wiper of R7, the open-collector output of IC1A will be driven on, causing capacitor C1 to discharge and transistor T1 to be switched off.
With T1 switched off, the voltage over R4 drops and the output of the comparator changes to the opposite state, allowing C1 to charge. When the voltage over C1 reaches the switching threshold of IC2A, T1 is switched on again and the cycle starts anew.
The voltage on the wiper of R7 corresponds to the peak current through L1. The time constant of R3/C1 results in a fixed off time and thus determines the ripple of the LED current, since the off time is the interval during which the current decreases.
Potentiometer R7 controls the LED current from approximately zero to a maximum value set by trimpot R6. For 1-watt power LEDs this should be at most 0.35 V, which results in a maximum peak current of 0.35 A because the value of R4 is 1 Ω.
Several LEDs in series can be connected to the output, as long as the combined forward voltage of the LEDs is less than the applied supply voltage (remember that this is a buck converter). If 15 V is not high enough, the junction of D1 and L1 can be connected to a separate higher supply voltage, which must be kept isolated from the control voltage.
The switching frequency is approximately 100 kHz and is determined by L1, R3, C1, and the supply voltage. Compared to the current state of the art, that is fairly low. A higher frequency would be possible with a faster comparator and a 'stiffer' gate driver, but in that case it would be better to use an integrated version of the circuit. One example is the ZXLD1350 [2], a complete one-chip controller that operates at a higher frequency. With a higher frequency, the value of L1 can also be reduced, resulting in a more compact implementation.
D1 and T1 are not critical. The types shown on the schematic diagram are simply what the author happened to have available. Any desired n-channel MOSFET can be used for T1 as longs as the source-drain resistance RDSon is less than 0.1 Ω, the maximum drain current I_{Dmax} is greater than 2 A, and the maximum drain-source voltage V_{DSmax} is at least 50 V. D1 is a Schottky diode with a maximum rated forward current IFmax of at least 2 A and a maximum rated reverse voltage V_{RRM} of at least 50 V. No

Web Links

[1] Pulse width modulation: https://en.wikipedia.org/wiki/Pulse-width_modulation

[2] ZXLD1350 data sheet: www.diodes.com/assets/Datasheets/ZXLD1350.pdf

[3] Flicker percentage: www.energy.gov/sites/prod/files/2015/05/f22/miller%2Blehman_flicker_lightfair2015.pd f

heat sinks are necessary. If you build this circuit, make sure that all ground connections are made to a single point at the ground terminal of R4.

The result

The flicker percentage (FP) with this approach is less than 1.5% (see the **inset**). A 60-W incandescent lamp has an FP of approximately 6.5%. The FP of this circuit increases when the LED is dimmed, because the off time is constant and the current consumption of the LED remains the same. In this regard, the ZXLD1350 performs better than this simple circuit because the IC actively measures the LED current and maintains the ripple of the LED current at a constant percentage of the set current. Consequently, the ripple decreases with decreasing LED current, so the FP remains nearly the same.

Conclusion

This circuit controls the brightness of an LED with a constant current, and only ordinary components are required to build the circuit. The flicker effect is nearly absent at every brightness setting, and the connecting leads of the LED fixture generate only a small amount of EMI. **Figure 4** shows the author's prototype.

(190062-02)

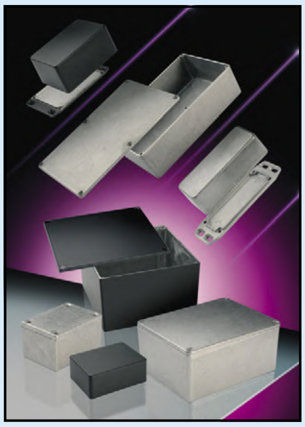
Advertisement

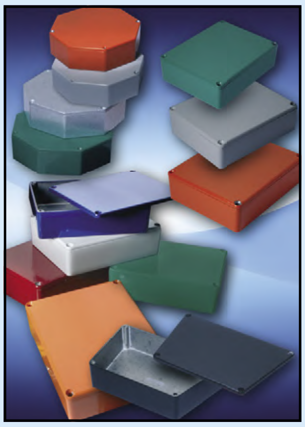
HAMMOND MANUFACTURING®

Die-cast enclosures standard and painted

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

01256 812812
sales@hammond-electronics.co.uk





productronica
fast forward
the start-up platform
powered by Elektor

COMPETE TO
LAUNCH YOUR STARTUP
@
PRODUCTRONICA 2019

p-ffwd 2019 – Participate Now!
November 12-15, 2019
Munich

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

Productronica Fast Forward is brought to you by productronica elektor INNOVATION STARTUP TRADE

Platinum Sponsor: DISTRELEC Distribution with a difference

Silver Sponsor: kurtz ersa Weller almit

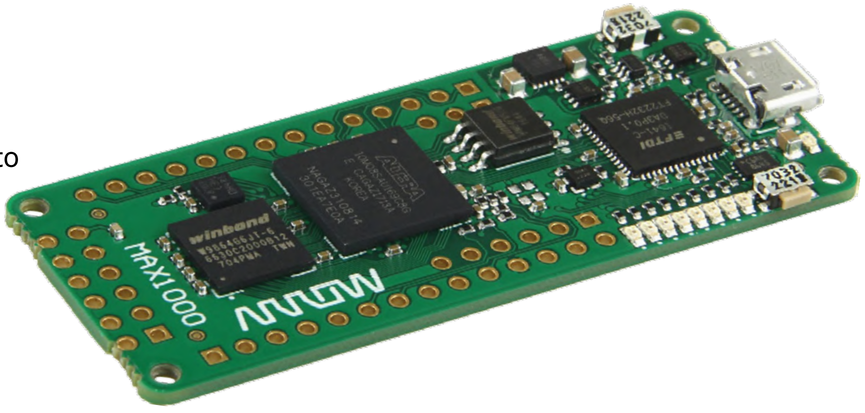
Bronce Sponsor: BERNSTEIN TOOLS FOR ELECTRONICS eLABS PROJECT

The SCCC Project (4)

Homebrew soft-core processor and C compiler

By **Martin Oßmann** (Germany) and **Mathias Claußen** (Elektor Labs)

In this installment we will look at how to output analogue signals with the help of sigma-delta DACs (digital-to-analogue converters) and interrupts.



In the previous installment of this series [3] we saw what peripheral modules are available to our homebrew CPU: these include a pulsewidth modulator (PWM), an analogue-to-digital converter (ADC), a GPIO interface and, last but not least, a user UART. These modules are already assigned input and output channels in the Verilog source code. We also saw how the designer can use additional Verilog commands to connect these peripherals to the pins of the FPGA, and from there control and read from external components such as an accelerometer. Finally we looked at how the peripherals are addressed from the C program running on our processor. Before we proceed to further experiments in this direction, it would be a good idea to take a look back over that article.

Experiment 6

In this sixth experiment in our course we will find out how to use sigma-delta DACs and interrupts. First to the DACs. The MAX10 FPGA itself has no DAC outputs; we could, as in many microcontrollers, realize a DAC function using PWM. However, we will instead look at an alternative DAC that works on the sigma-delta principle.
In the top level module we instantiate two modules of type `firstOrderSigmaDeltaDacV01`, whose outputs are connected to the pins PMOD1 and PMOD2 of the FPGA. The input data buses are eight bits wide and connected to channels 19 and 20, where they can be accessed using an `OUTA` instruction (the peripheral modules mentioned in the introduction already occupy channels 0 to 7 and 18).

```
reg[8-1:0] aDacSignal ;
reg[8-1:0] bDacSignal ;
firstOrderSigmaDeltaDac1v01 #(.MSBI(7)) DACa
   (clk_100, aDacSignal, PMOD1) ;
firstOrderSigmaDeltaDac1v01 #(.MSBI(7)) DACb
   (clk_100, bDacSignal, PMOD2) ;

always @(posedge sCclk) begin
  if(outAstrobe) begin
    if( outAchannel==0) begin   ... end
     else if( outAchannel==19) begin aDacSignal
   <=outA[8-1:0] ;   end
    else if( outAchannel==20) begin  bDacSignal
   <=outA[8-1:0] ;  end
    end
  end
```

The sigma-delta code itself is as follows.

```
module firstOrderSigmaDeltaDac1v01  #(parameter
   MSBI=7)
 (input            clk ,
  input  [MSBI:0]  DACsigIn ,
  output reg       DACout
  );

reg [MSBI:0]   DACin;
   // DAC input (with an offset of 2**MSBI)
reg [MSBI+2:0] DeltaAdder; // output of delta adder
reg [MSBI+2:0] SigmaAdder; // output of sigma adder
reg [MSBI+2:0] SigmaLatch;
   // latched output of sigma adder
reg [MSBI+2:0] DeltaB;     // B input of delta adder
```

```
always @(SigmaLatch)             DeltaB = <<
   (MSBI+1);
always @(DACin or DeltaB)        DeltaAdder = DACin
   + DeltaB;
always @(DeltaAdder or SigmaLatch) SigmaAdder =
   DeltaAdder + SigmaLatch;

always @(posedge clk) begin
  DACin <= DACsigIn ;
  SigmaLatch <= SigmaAdder;
  DACout <= SigmaLatch[MSBI+2];
  end

endmodule
```

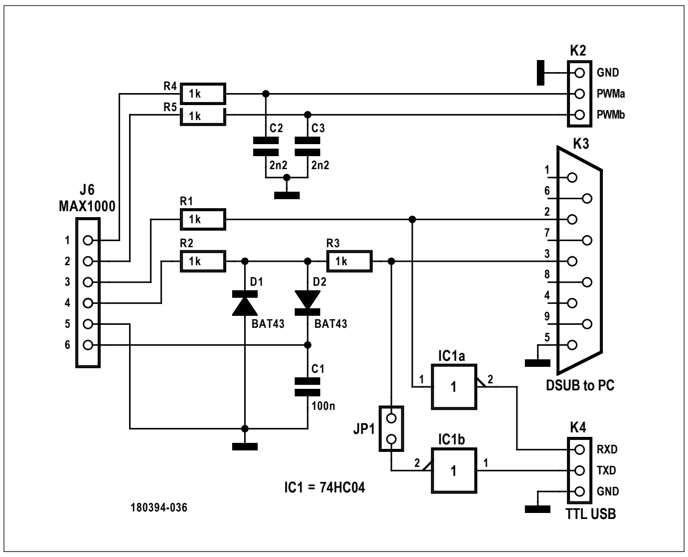


Figure 1. Signal outputs.

In comparison to a PWM DAC a sigma-delta DAC generates less noise at lower frequencies. The external hardware that needs to be connected is shown in **Figure 1**. In the C code we once again use `OUTA` instructions wrapped in a suitable function. This code sits in a loop waiting until the variable `mail` is set to 1 by the interrupt routine. Then this variable is reset to zero, and the parameters `x` and `y` to the function are output on channels 19 and 20 respectively.

```
#asm
SigmaDeltaDacA      EQU  19
SigmaDeltaDacB      EQU  20
#endasm


outDACs(int x, int y) {
  while(mail==0){ }
  mail=0 ;
  x ;
#asm
  OUTA   SigmaDeltaDacA
#endasm
  y ;
#asm
  OUTA   SigmaDeltaDacB
#endasm
  }
```

The routine `outDACs` is short enough that we can list what the compiler makes of it in its entirety.

```
outDACs        EQU $
// start of the outDACs function
// parameter x is at address 2 relative to the stack
   pointer
// parameter y is at address 1 relative to the stack
   pointer
// the return address is at address 0 relative to the
   stack pointer

__LBL00000     EQU $       ; whileStackBase=0
                LDPUSH      @ __mail
// copy value of mail to the stack
                LDI         0  ; hex=00000000
// load R0 with zero
                POP         R1
// load R1 with the value of mail from the stack
                CMPEQ
// compare for equality
                JFALSE      __LBL00001
// if not equal, break out of loop
                JMP         __LBL00000
// continue in loop
__LBL00001 EQU $
                LDI         0  ; hex=00000000
// load R0 with zero
                ST          @ __mail
// store zero from R0 to mail
                GETLCL      2
// local variable x is at address 2 relative to the
   stack pointer zsp=0
                OUTA        SigmaDeltaDacA
// send value to DACa
                GETLCL      1
// load local variable y into R0
                OUTA        SigmaDeltaDacB
// send value to DACb
                RET
// return from function
```

The `LDPUSH` and `GETLCL` instructions have been generated here by the optimizer using the following templates. From a sequence of the form

```
    LD          @ __mail
    PUSH        R0
```

the instruction

```
    LDPUSH      @ __mail
```

is generated, and likewise from a sequence of the form

```
    LDI         2
    IADD        SP
    LDIND       @R0
```

the instruction

```
    GETLCL      2
```

is generated. These optimizations make the code shorter and faster. Next we will look at interrupts.

Interrupts

The sCCCP CPU currently supports four interrupts, of which just two (the RTC interrupt and the UART RX ready interrupt) are used. Interrupt processing is handled in the CPU during the fetch phase. Under normal circumstances the current instruction is fetched in this phase, but if there is a pending interrupt (`selectedIRQ != 0`) then the behaviour is different.

First the program counter is saved on the stack. Then the program counter is set to eight times the interrupt number: thus for IRQ1 it is set to 8, for interrupt 2 it is set to 16, and so on. The action of writing the old program counter value to the stack is overlapped with the next fetch phase, which follows immediately on from the current interrupt fetch phase. The Verilog code for the CPU expresses this as follows.

```
if(selectedIRQ) begin
  writeAddress <= sCregSP-1'b1 ;
  writeValue   <= sCregPC ;
  writePending <= 1 ;
  sCregSP <= sCregSP-1'b1 ;
  sCregPC <= selectedIRQ << 3 ;
  sCstate <= sCstateFetch ;

  if(selectedIRQ==1) begin irq0reg<=0 ; end
  ...
  end
```

In order to arrange for a C function to be called under interrupt control, we follow the example here from the RTC code.

```
#asm
RTCintVector       EQU   8

  ORG       RTCintVector
  PUSH      R0
  PUSH      R1
  CALL      RTCinterrupt
  POP       R1
  POP       R0
  RET
#endasm
...

int mail ;

RTCinterrupt(){
  mail=1 ;
  }
```

In the interrupt routine itself all that we do is set the variable `mail` to 1. Then we just have to take care of the following: in the main code the register `RTCtimerTOP` is set to 50, which means that every 50 clock cycles an interrupt will be triggered. We can now enable the RTC interrupt using `setCPUflags(RTCintEna)`. Then we call the function `doSamples()`, which sends a repeating sequence of 32 values to the two DAC channels.

```
#asm
CPUflagsChannel    EQU   6
RTCtimerTopChannel EQU   7
#endasm

doSamples(){
// written by SinTabGen1v01.lua
// 128+127*sine(Xi)
while(1){
  outDACs(  128 ,    0 ) ;
  outDACs(  153 ,    8 ) ;
  .....
  outDACs(  103 ,  248 ) ;
  }
}

#define RTCintEna 1

main(){
  setRTCtimerTop(50) ;
  setCPUflags( RTCintEna ) ;
  doSamples() ;
  }
```

This experiment is again relatively easy to try out. First synthesize the Quartus project ‘experiment6.qpf’ and load the resulting bitstream into the FPGA. Then compile the C code in ‘experiment6.c’ and load it into the CPU using the Processing uploader. You should now see a sinewave on PWM output 1 and a sawtooth signal on output 2.

Experiment 7: sampling oscilloscope

Our experiments so far have tended to be in the way of demonstrations. Next we will look at an experiment that is closer to being a real application. We will use the analogue-to-digital converter, which is provided in the MAX10 chip, to record data and send them to the PC over a serial interface. The PC can then display the data, much like an oscilloscope. And, of course, it is possible to use an FFT function on the PC to implement a spectrum analysis function.
In order to build this application we will work at three different levels. At the FPGA level we will be expanding our CPU so that it

is capable of storing 8192 samples in a buffer in the FPGA with the ADC running at its maximum sample rate of 1 megasample per second. The contents of the buffer will then be read out by a program running on the CPU, and the data will be sent to the PC. And at the third level we will arrange for the PC to display the values and calculate the spectrum.

Reading the values into the FPGA goes as follows. At the heart of the process is the write pointer register `traceWritePtr`, which in the idle state has the value 8191. If the value in this register is less than 8191, then data values will be read in until it again equals 8191. To initiate the reading process we simply set the pointer to zero, and then 8192 values will be written consecutively to the buffer `traceBuffer`. The write pointer is set using an `OUTA` instruction. It is also possible to read the value in the write pointer over channel 32, which lets us determine when a complete buffer of values has been read in.

The read pointer `traceReadPtr` is used to read data from the buffer. This can be set using an `OUTA` instruction over channel 33. The value in the buffer at the address pointed to by this pointer can be read using an `INPA` instruction over channel 33.

```
parameter traceLength = 8192 ;
parameter traceLengthWidth = 13 ;
reg [12-1:0] traceBuffer[0:traceLength-1] ;
reg [traceLengthWidth-1:0] traceWritePtr ;
reg [traceLengthWidth-1:0] traceReadPtr ;
reg trigger ;

// End Of Conversion pulse eocPulse kommt vom ADC

always @(posedge sCclk) begin
   if ( (outAstrobe) & ( outAchannel==32) ) begin
   trigger<=1 ; end
   if (eocPulse) begin
      if(trigger) begin
         traceWritePtr<=0 ;
         trigger<=0 ;
         end
      if(traceWritePtr != traceLength-1 ) begin
        traceBuffer[traceWritePtr] <= ADCval ;
        traceWritePtr <= traceWritePtr+1 ;
      end
     end
  end

always @(posedge sCclk) begin
  if(outAstrobe) begin
    if( outAchannel==33) begin traceReadPtr<=outA ;
      end
    end
  end

assign inpA=
  (inpAchannel==0) ?  { 32'h12341234   } :
  .....
  (inpAchannel==32) ? traceWritePtr :
  (inpAchannel==33) ? traceBuffer[traceReadPtr] :
  32'h12341111 ;
```

In total, that is less than forty lines of Verilog. It is often the case that when processing signals that the functionality required in the FPGA can be implemented in a short piece of code.

The next step is to send the values from the buffer to the PC. Programming that in Verilog at the FPGA level would be somewhat arduous; it is simpler to use our soft-core CPU for this purpose. The serial protocol between the MAX1000 board and the PC is organized as follows. When the PC sends an 'x', the FPGA will start the sampling process and continually read from the ADC until 8192 values are sitting in `traceBuffer`. Then the sCCCP CPU will fetch the data from the buffer in the FPGA and send them serially at 115200 bits/s to the PC. At the end a checksum will be sent. The code example 'experiment7.c' also shows you how the sCCCP CPU reads the data. The code is as follows.

```
doCommand(){
  int c ;
  c=getCharPolling() ;
  if(c=='x') { execute() ; return ; }
  putChar('?') ;
  }

execute(){
  int data,k,i,chkSum ;
  traceTrigger() ;
// reset traceWritePtr=0
  while( getTraceWritePtr() !=traceLength-1) { }
// wait until end reached
  chkSum=0 ;
  putChar('[') ;
// open frame
  for(k=0 ; k<frameLength ; k++){
// loop over the values
      setTraceReadPtr(k) ;
      data=getTraceData() ;
// get traceData[k]
      word12Out(data) ;
// output 3 nibbles in hex (12 bits)
      chkSum=chkSum+data ;
// update checksum
      putChar('.') ;
// . after every value
      if( (k&0x1F)==0x1F ){ crlf() ; }
// increase readability
    }
  word12Out(chkSum) ;
// send chkSum to PC
  putChar(']') ;
// close frame
  crlf() ;
  }
```

The analogue input is connected as shown in **Figure 2**. First of all fit jumper JP1 and leave the analogue input 'Analog in' unconnected. This arrangement will send the PWM signal to the ADC. Later we will remove JP1 and connect the AC-coupled input 'Analog in' to the signal of interest.

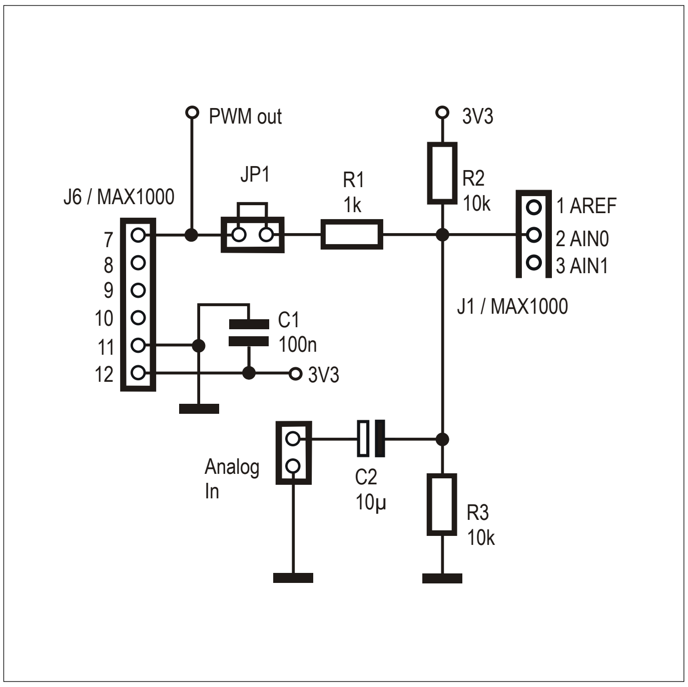


Figure 2. Circuit of the analogue input AIN0.

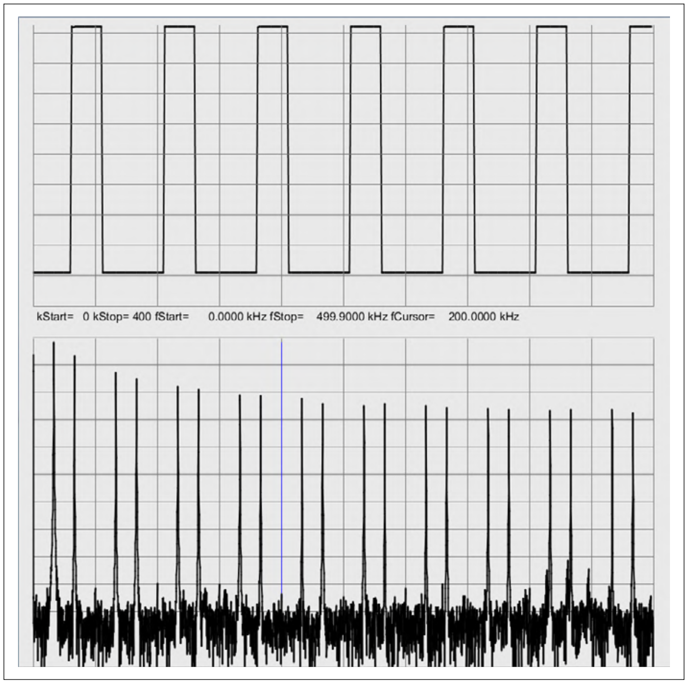

kStart= 0 kStop= 400 fStart= 0.0000 kHz fStop= 499.9000 kHz fCursor= 200.0000 kHz

Figure 3. The upper part of the display shows the sampled data, while the lower part shows the corresponding spectrum.

The PWM signal is generated using the following code.

```
setPWMtop(2000) ;
setPWMperiod(6000-1) ;
```

The period of the PWM signal is 6000 times the period of its 100 MHz clock, or 60 µs. The 'on time' of the signal is 2000 cycles, or 20 µs. Now we need a program to run on the PC side to fetch the sampled data values and display them. For that we will use the Processing sketch 'C:/sCCCP/experiment7/Processing/ShowGraph1/ShowGraph1/ShowGraph1.pde'. Before running it, it is necessary to adjust the third line of the code ('`String serialPort="COM2"`') appropriately to match the serial port that you are using to connect to the user UART of the MAX1000. You can test whether the correct code is running on the MAX1000 with the help of a terminal emulator program. If you send an 'x' (for 'execute') to the MAX1000 you should be rewarded with a long stream of data of the form [804.842.....]

The Processing sketch uses these values to generate a display like the example shown in **Figure 3**. The upper part of the screen shows the sample data, while the lower part shows its spectrum.

This application is not far from emulating the function of a DSO or a spectrum analyser, the main thing missing being a trigger function to initiate the recording process. It would be possible to add a complete GUI to the code, and of course a considerably faster ADC could be used in order to allow higher-frequency signals to be displayed. And if instead of recording the samples coming from ADC we recorded the states of a group of digital inputs, we have the makings of a logic analyser.

In the next and final part of this series we will pimp out our hardware still further: in experiment 8 we will receive reports from the DDH47 weather service transmitter, which operates on 147.3 kHz. The results will be displayed on a VGA monitor. ⏮

180394-D-02

Web Links

[1] The SCCC Project (1), ElektorLabs magazine 2/2019: www.elektormagazine.com/magazine/elektor-88/42444

[2] The SCCC Project (2), ElektorLabs magazine 3/2019: www.elektormagazine.com/magazine/elektor-96/42664

[3] The SCCC Project (3), ElektorLabs magazine 4/2019: www.elektormagazine.com/magazine/elektor-100/50909

[4] Project page accompanying this article: www.elektormagazine.com/180394-D-02

Horticulture Box

A grow light using 'horticulture' LEDs from Würth Elektronik

By **Luc Lemmens and Mathias Claußen** (Elektor labs), with contributions by **Würth Elektronik**

The use of and research into, artificial electric light sources to stimulate plant growth dates to the second half of the 19th century. It closely followed the development of human electric lighting, which can be divided into three main paths of development: incandescent, open-arc and enclosed gas discharge lamps. And of course, our cherished LEDs are now being used for horticultural purposes too. Here we present a plant-friendly LED light source that leaves little to be desired.

The LED theory behind grow lights is more complex than you would imagine and exceeds the scope of this article. Fortunately, useful coverage of the subject may be found in application notes ANO002, ANO003, ANO004 on the Würth Elektronik website [1]. Here we can limit ourselves to mentioning the most important advantages of LEDs over their more conventional predecessors as far as horticultural use is concerned.
Quintessentially, it's not just the intensity but also the spectrum of the applied light which really matters for the growth and development of plants. Here are some typical bandwidths and their known effects:

- Red light (630–660 nm) is the main driver of photosynthesis and is an important effector for the growth of stems. This wavelength can also regulate flowering, dormancy and seed germination.
- Blue light (400–520 nm) is another key wavelength for photosynthesis but must be carefully controlled and mixed with other frequencies as overexposure at this wavelength may inhibit growth. This wavelength has also been linked to the regulation of chlorophyll concentration, lateral bud growth and leaf thickness.
- Far red light (720–740 nm), which is in the IR spectrum, affects germination and can reduce the flowering time of plants but also increase stem length as part of an effect called 'shade avoidance response'.
- Green light (500–600 nm) was once disregarded as largely insignificant to plant development. However, recent investigations have revealed that plants shaded by others are particularly responsive to this wavelength as part of the "shade avoidance response".
- UV light (280–400 nm) is still highly experimental in the cultivation of plants. Although this wavelength is mutagenic, some plants (e.g. lettuce, tomato) are much more resistant to these wavelengths. UV light may be responsible for the generation of certain protective molecules like antioxidants and phenols, which are important for human nutrition.

As opposed to older light sources, LEDs can be manufactured to emit very specific bandwidths of light. That allows us to tune the spectrum of a grow light (containing different colours of LEDs) to the specific needs of a plant. Würth Elektronik offers the "WL-SMDC SMD Mono-colour Ceramic LED Waterclear" range of LEDs. This range has been expanded to include wavelengths of 450 nm (deep blue), 660 nm (hyper red) and 730 nm (far red), which have been selected to match the absorption spectra of photosynthetic pigments. These three LED variants are contained in our grow light, together with some cool white (6000 K) LEDs.
For this purpose, the ESP32 microcontroller in the Horticulture Box has a dedicated web interface permitting the intensity level of each of the four channels to be controlled independently, or the overall intensity of the four channels, via Wi-Fi using a computer, smartphone or tablet. This means you can easily customize and experiment with different light recipes in your grow box.
There are many more advantages when using LEDs in grow lights, including size, efficiency, and durability.

PROJECT DECODER

LED | PWM | ESP32 | grow light | horticulture

entry level
➡ intermediate level
expert level

2 hours approx.

Arduino IDE (optional)

€175 / £165 / $195 approx.

Driver board

Most of the circuitry was gratefully borrowed from a Würth Elektronik Reference Design covering a four-channel PWM LED driver. Designers Luc Lemmens and Matthias Claussen at the Elektor Labs replaced a PIC microcontroller plus Bluetooth module from the original design

Quick Features

- 4 colour channels: red, far red, blue, white
- Würth Elektronik WL-SMDC high brightness LEDs
- 300 mA / 350 mA / 450 mA LED current limiting
- 4 PWM drivers
- ESP32 controlled
- Fully adjustable colour blend and intensity
- Timer to mimic day/night cycle
- 55 cm radius illuminated surface
- Wi-Fi controlled (web interface / MQTT)
- Operates from any 24 V, 50 W power source
- Driver board and LED board available preassembled

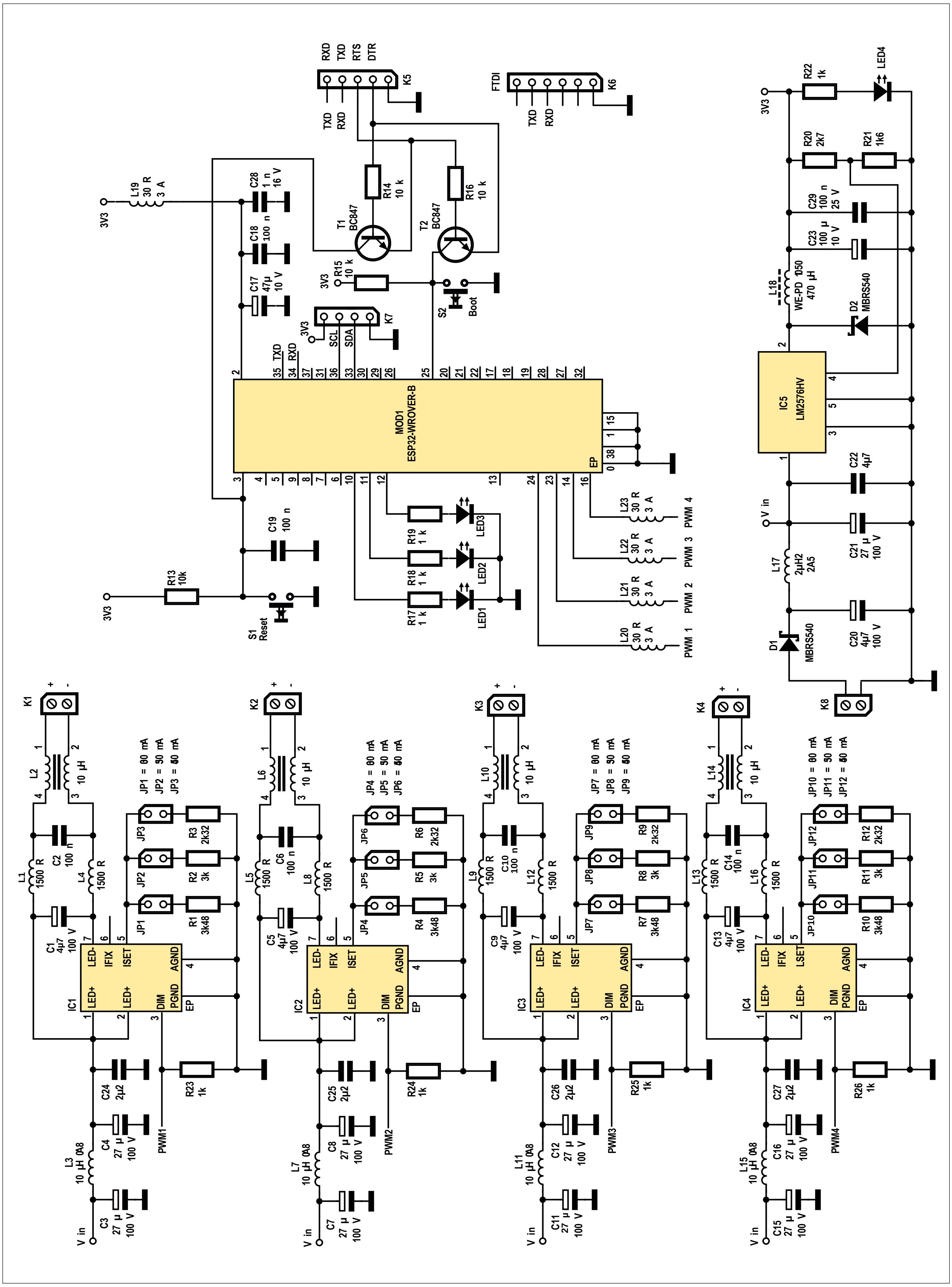


Figure 1: Schematic of the Horticulture Box driver board. Here the ESP-32 module rules the roost.

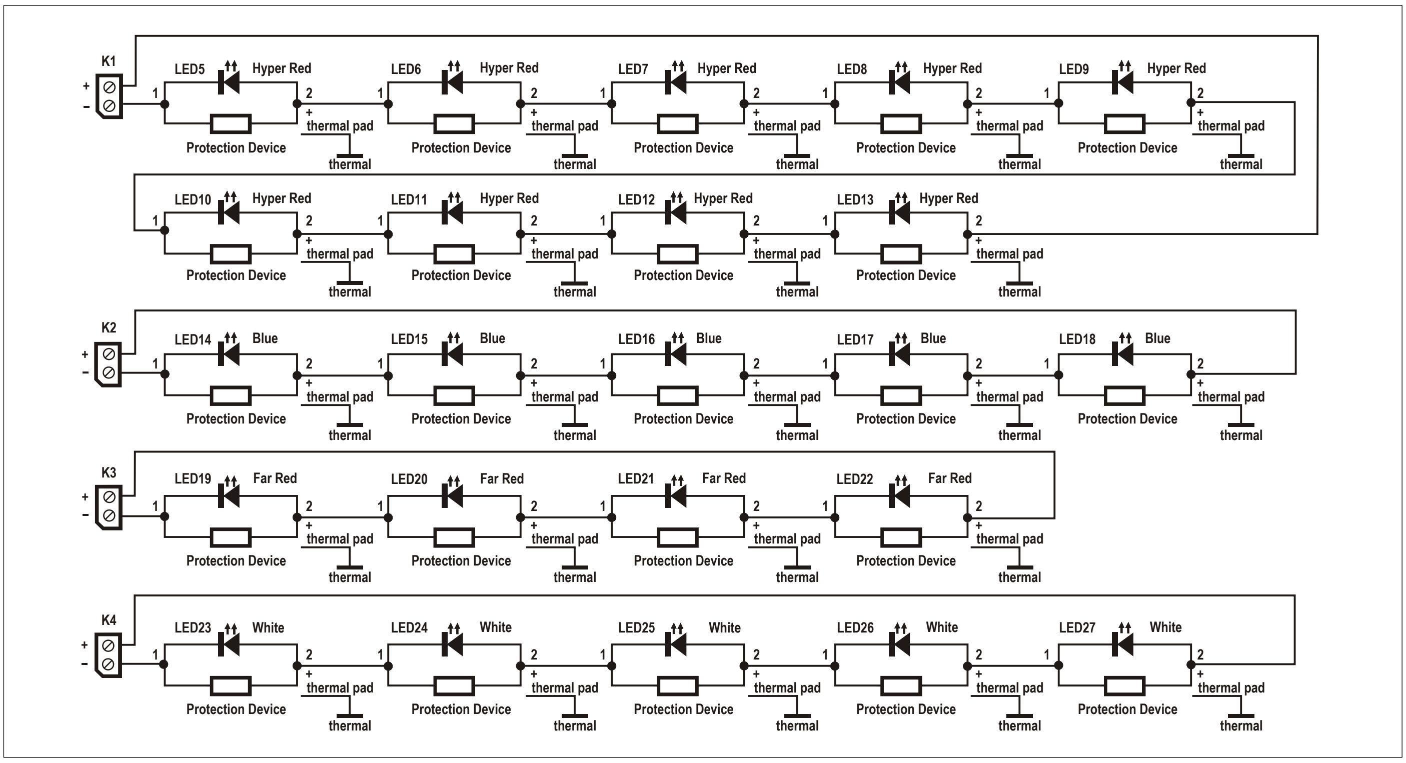


Figure 2: Schematic of the LED board.

with Elektor's popular ESP32 module type ESP32-WROVER-B. Luc decided to use a LM2576 DC/DC converter to supply the ESP32 instead of the Würth Elektronik power module from the original design. The results of the rework operation are seen in the circuit diagram presented in **Figure 1**. The schematic includes all inductors and capacitors necessary to pass the EMC limits of the norm EN55015 and CISPR32 even during dimming. In the modified design the Elektor/Würth Elektronik part of the four driver stages passed the pre-compliance tests without any problem. We virtually copied both the schematic and the layout of the PCB here. The ESP32 on the other hand, presented serious problems with 80 MHz and higher harmonics towering in the spectrum. Not surprisingly, 80 MHz happens to be the bus clock frequency of the ESP32. A partial redesign of the circuit board was called for, involving some rerouting of the ground plane and some EMI chip bead ferrites added to tackle this issue. For the rest, the schematic is rather straightforward. The ESP32-WROVER module is of course the 'brains' of the driver board, generating the PWM signals controlling the intensity. The micro takes care of the user interface as well as the connection to the local Wi-Fi network. The LM2576 step-down converter (IC5) provides the 3.3-V supply voltage to the ESP32 module (MOD1); LED4 (blue) indicates that power is switched on. The input voltage of the grow light can be as high as 45 V. However, to limit the power dissipation on the PCB with the LEDs on, we settled on limiting this voltage to 24 V.

The ESP's UART can be accessed either via K5 or K6 for serial communication, or function in the programming of the ESP when it is in bootloader mode. The connections to DTR and RTS on K5 allow the Arduino programming environment to automatically switch the ESP32 to bootloader mode via T1 and T2. However, you will need a USB to UART interface that provides these two handshake signals, like the Elektor FT232R USB/Serial Bridge/BoB or the FT231X BoB. You can also use a standard 3.3-V FTDI cable and hook it up to K6, but in that case — as you may know from other ESP-based projects in Elektor — you'll need to employ pushbuttons S1 and S2 to manually reset and switch the ESP into its bootloader mode.

When you buy the ready-assembled Horticulture Box circuit boards from the Elektor Store you don't need to bother about firmware programming, while a firmware update can be done 'OTA' (over-the-air). That's right, no cable or interface is needed to perform an update.

SDA and SCL level I^{2}C bus signals with 3.3 V swing may be connected on K7 for future developments. Pull-up resistors may be needed if you want to connect a device to this bus. You may also need bidirectional 5 V-to-3.3 V level shifters, keeping in mind that the I/O pins on the ESP32 are not 5-V tolerant!

The ESP supplies the PWM signals at the DIM inputs of LED drivers IC1–IC4. The output current is limited to either 300 mA, 350 mA or 450 mA by closing one of three jumpers associated with a driver.

LED1 lights up green when the ESP32 is connected to a Wi-Fi network and switched to Station Mode. It flashes when the ESP32 is in Access Point mode. LED2 (red) lights when an OTA firmware update is being performed. Yellow LED3, finally, flags an MQTT connection being made.

LED board

This may appear the least complicated part of this project, but appearances are deceptive from a look at the simple schematic in **Figure 2**. First, we had to decide on the LED colours and how many of each to include in our grow light. Elektor Labs having zilch experience in this field, they relied on the expertise and knowledge of

Figure 3. Mean Well 24 V / 2.2 A switch mode power supply type LRS-50-24.

PCB Heat Management

As the performance and miniaturization of power LEDs continue to advance, the heat dissipation of LED applications is becoming increasingly important. A thermal management concept has to be developed to keep the thermal load of the components within the permitted limits. The goal must be to keep the temperature of the LED die as low as possible.
Vertical heat conduction by means of so-called thermovias directly below the cooling surface of the LED can help. So can horizontal heat conduction in large copper areas both on top and bottom layers.
In Würth Elektronik's Heatsink technology, the heat spread in this way is conveyed over a large area to a glued aluminium heatsink using a transfer adhesive. The special advantage compared to the IMS technology is the possibility to build multilayer circuits.
Thermovias are holes specially placed in the printed circuit board for heat transfer. Ideally, these holes should be located directly below the heat source, allowing direct heat dissipation. The hole diameter is typically 0.30 mm and the wall thickness of the copper sleeve in the hole is at least 25 µm. Larger diameters are possible for thermovias.
Since thermovias are filled and closed with a copper 'lid', solder flow is prevented, and components can be soldered onto the pads without problems.
Good approximations of the thermal resistance and temperatures prevailing in and on the printed circuit board can be carried out by means of simulations or the creation of an FEM model.
As well as by vertical heat conduction, the heat will be distributed laterally across a larger surface on the PCB. A large area is an important prerequisite for high vertical heat transfer due to thermally less conductive materials, e.g. insulation layers between PCB and heat sink, or between PCB and the environment.
Common methods of cooling by heat spreading include distributing the heat in copper layers and securing the PCB to a heat sink.
Heat sinks made of aluminium have a proven track record in PCB practice. A common method is pressing with a FR4 prepreg or the use of transfer adhesive. The advantage of using the adhesive compared to a prepreg is the dynamics during soldering. The different coefficients of expansion of the aluminium and the printed circuit board can be better compensated.
With the technology used by Würth Elektronik, the heat sink is cold bonded to the printed circuit board under vacuum. This bond by means of a transfer adhesive is called TWINflex technology.
Ideally, the reverse side of the printed circuit board is produced with a full-surface copper layer. The heat is distributed on this level and is then transferred to the heat sink by the transfer adhesive. The thermal resistance of the application is reduced by the larger area available. The aim is to achieve a minimum PCB thickness with maximum heat sink surface area.

Würth Elektronik to get a 'blend' that will work for a relatively small horticulture box. The resulting LED board should be enough to cover a circular area with a radius of approx. 55 cm. The outer zone is less illuminated than the centre.
Second: high-intensity LEDs do heat up. Würth Elektronik has the technology and facilities to develop and produce PCBs that keep LEDs as cool as possible in order to slow down ageing. The method essentially involves gluing a thin PCB to an aluminium heatsink — more on this in the **PCB Heat Management** text box. Otherwise a perfect solution, the one drawback is higher cost compared to a simple PCB.

To keep this project affordable, we decided to design a thin (0.5 mm... phew!) PCB with large copper planes on both sides. The LEDs in the WL-SMDC series have electrically neutral thermal pads that can be soldered to the copper plane at one side of the PCB. This side in turn is connected to the copper plane at the other side through thermal vias. By default, the jumpers on the driver board are set to 300 mA current flow on all four channels. This value permits the grow light to work without a heat sink but might be a bit low for some plants.

Building and setting up the grow light

The preassembled boards from the Elektor Store effectively limit your contribution to setting the jumpers, interconnecting the boards and connecting a 24 V / 2.5 A DC power supply. There are numerous ready-built and affordable supplies that can do the job. We did the prototyping with a benchtop PSU for sure, while EMC pre-compliance tests were performed using an Egston type BI60-240250-E2 power supply. Mean Well has a very elegant and small 24 V / 2.2 A SMPS type LRS-50-24 that works great too (**Figure 3**).
If you decide to solder the PCBs yourself, it is highly recommended to have solder paste and a hot-air soldering station as a minimum. A small soldering iron may work too, but make sure to place the components in the right order as taller components may block access to pads for the lower ones. SMD parts land first. Do not start with the through-hole components until after a thorough check of the SMD solder work!

Firmware programming and updating

The preassembled driver board from the Elektor Store comes with the ESP32 module ready programmed and up and running. Only if you want to change the system firmware, you need to install the Arduino IDE on your computer, with support for the ESP32 core. This process is thoroughly explained at [2].

You'll also need to download the firmware for this project from the Elektor Labs website [3]. The comment lines at the beginning of the main sketch called 'Firmware.INO' contains a list of the Arduino libraries required to be able to compile the source code.

If you have a USB UART with RTS and DTR handshake signals, connect it to K5. The Arduino IDE is now able to upload the firmware and sketch data automatically. If you have a UART without these handshake signals (e.g. an FDTI cable connected to K6), you need to operate pushbuttons S1 and S2 to switch the ESP32 into bootloader mode before the IDE can transfer data: keep S2 depressed while at the same time pressing and releasing S1, then release S2.

In the default firmware, Arduino OTA is enabled, meaning that you can flash the system across Wi-Fi without a USB connection, or pressing buttons. Connect to the grow light in Access Point mode or let the device connect to your local network. Start the Arduino IDE, open the Tools menu and select the 'ESP32 WROVER Module'. In the submenu 'Ports' you will see a HC-LED-XX-XX-XX as a Port to select. This is the ESP32 on the Horticulture Box. If selected, you can press the Upload button to get the code transferred to the chip. If everything worked well after the upload, the chip will reboot and the new firmware is in place. But be warned, if you modify the firmware and break the Arduino OTA, you'll need to attach a cable to flash the system from

COMPONENT LIST, DRIVER BOARD

Resistors

R1,R4,R7,R10 = 3.48kΩ, thick film, 1%, 0.125W, 150V
R2,R5,R8,R11 = 3kΩ, thick film, 1%, 0.125W, 150V
R3,R6,R9,R12 = 2.32kΩ, thick film, 1%, 0.1W, 150V
R13,R14,R15,R16 = 10kΩ, thick film, 5%, 0.1W, 150V
R17,R18,R19,R22,R23,R24,R25,R26 = 1kΩ, thick film, 5%, 0.1W, 150V
R20 = 2.7kΩ, thick film, 5%, 0.1W, 150V
R21 = 1.6kΩ, thick film, 1%, 0.125W, 150V

Inductors

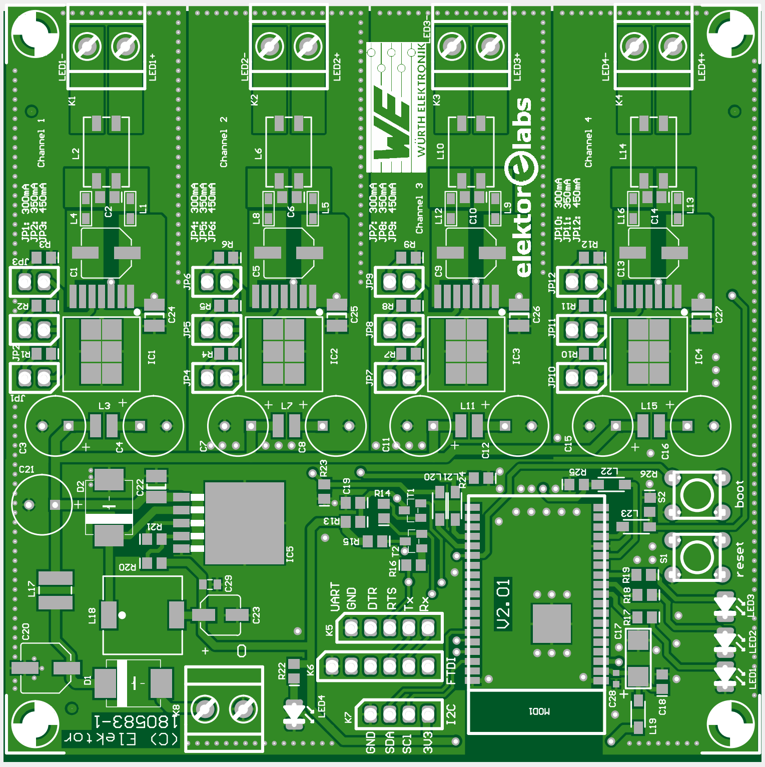
L1,L4,L5,L8,L9,L12,L13,L16 = EMI suppression ferrite, 1500Ω@100MHz, 0805, WE-CBF
L2,L6,L10,L14 = SMD common mode line filter 10µH, 1.6A, WE-SL2
L3,L7,L11,L15 = SMD power inductor, 10µH, 800mA, WE-PD2, size 3521
L17 = SMD power inductor 2.2µH, 2.5A, WE-PD2, size 4532
L18 = power inductor (SMD), 470µH, 600mA, WE-PD 1050
L19,L20,L21,L22,L23 = ferrite bead, 31Ω, 3A, size 1206

Capacitors

C1,C5,C9,C13,C20 = 4.7µF, 100V, 7.7 x 6.3mm
C2,C6,C10,C14,C18,C19,C29 = 100nF, 100V, X7R, 0805
C3,C4,C7,C8,C11,C12,C15,C16,C21 = 27µF, 100V, 20%, 8mm, radial
C17 = 47µF, 10V, 2312
C22 = 4.7µF, 50V, X7R, 1210
C23 = 100µF, 10 V, 5.5 x 5.5mm
C24,C25,C26,C27 = 2.2µF, 100V, X7R, 1210
C28 = 1nF, 16V, X7R, 0603

Semiconductors

D1,D2 = MBRS540, 40V, Vf=550 mV @ If=5A
LED1 = LED, green, 3mm
LED2 = LED, red, 3mm
LED3 = LED, yellow, 3mm
LED4 = LED, blue, 3mm
T1,T2 = BC847C, 45V, 100mA, 250mW, hfe=400
IC1,IC2,IC3,IC4 = MagI3C LED Step Down High Current, Würth 172946001
IC5 = LM2576HVS-ADJ, step-down regulator, 4-60V, 3A
MOD1 = ESP-32-WROVER-B



60% of true size

Miscellaneous

S1,S2 = switch, tactile, 12V, 5 mA, 6x6mm
K1,K2,K3,K4,K8 = PCB terminal block, 0.2" pitch, 2-way, 630V
K5 = pinheader, 5-pin, 0.1" pitch, vertical
K6 = pinheader, 6-pin, 0.1" pitch, vertical
K7 = pinheader, 4-pin, 0.1" pitch, vertical
JP1,JP2,JP3,JP4,JP5,JP6,JP7,JP8,JP9 = jumper, 1x2, 0.1" pitch, vertical
4 pcs. jumper, 2-way, 0.1" pitch
PCB 180583-1 V2.01

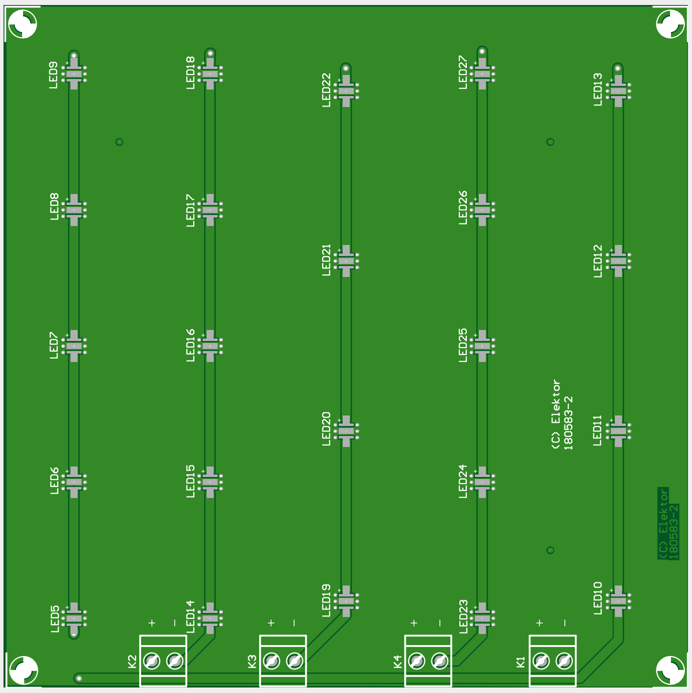
COMPONENT LIST, LED BOARD

Semiconductors

LED5,LED6,LED7,LED8,LED9,LED10,LED11,LED12,LED13 = high brightness LED, WL-SMDC series, red, 660nm, 125°, 700mA
LED14,LED15,LED16,LED17,LED18 = high brightness LED, WL-SMDC series, blue, 450 nm, 125 °, 700 mA
LED19,LED20,LED21,LED22 = high brightness LED, WL-SMDC series, red, 730nm, 125°, 700mA
LED23,LED24,LED25,LED26,LED27 = high brightness LED, WL-SWTC series, white, 120°, 121lm, 6000K, 700mA

Miscellaneous

K1,K2,K3,K4 = PCB terminal block, 0.2" pitch, 2-way, 630V
PCB 180583-2 V1.1



60% of true size

Caution, Eye Damage!

To avoid permanent damage to your eyes, never look directly at the horticulture light source or other high brightness LEDs when illuminated. The intensity of the LED light is very high, even at reduced effective power.
There is another thing to consider when the brightness of LEDs is controlled by PWM as is the case in this project. In contrast to incandescent lamps, for example, the intensity of the light emitted by an LED can vary very rapidly. An LED that appears to be off can actually pulse extremely brightly for a very short time, invisible to the human eye. It is not certain how harmful this is to the eyes, but it will certainly not be healthy.
Perhaps the safest approach is to never look directly at high intensity LEDs even if they appear to be switched off.

scratch.
The uploading of ESP sketch data can also be performed OTA, just click 'ESP32 Sketch Data Upload' in the Tools menu.

The software bits

In this project hardware and software development ran cheerfully in parallel, causing a nice exchange of information during the design process, especially when it came to the I/O pins used. Some of the pins got swapped during the design to make routing of the PCB a tad easier, and if it is just a line of code that needs to be changed, doing so can save a lot of hassle in the board layouting.

Back to what's being used for the software, those of you having absorbed the most recent issues of Elektor may have noticed that we have done quite a few ESP32-based projects. As a developer you don't start from scratch every time; instead you start to build a set of 'components' you can connect like pieces of a puzzle. For the Horticulture Box software, the basic components are like those found in the 'ESP32 Bedroom Clock' [4] or the 'ESP32 Weather Station' [5], namely a webserver and basic routines to configure the connection to a wireless network.

For the Horticulture Box some additional modules were included. First, we have 'ledc', a PWM unit with some nice additions to drive LEDs. This is easy to set up by telling it the desired PWM frequency and resolution. For the Würth Elektronik LED driver is based on the dimming ratio . There is also a minimum on-time you need to observe. As with every PWM unit you are keen to know the resulting resolution at a given PWM frequency. The PWM has 40 MHz as its input frequency and we can calculate the maximum resolution in bits from:

$\log_2 (40 \times 10^6 / 250) = 17.2$

This leads to 17 usable bits of resolution for the PWM system. As we have some minimum on-time for the PWM, we only use 8 bits in the code. To initialize the PWM unit you can use this line of code:

```
ledcSetup(CHANNEL,
   PWM_FREQ ,PWM_RES );
```

This will set up one of eight available LED channels. Unlike many other systems, this ledc channel is not bound to a special pin on the ESP32. Instead we can use any possible output pin. To set this up we need the following line:

```
ledcAttachPin(GPIO_PIN, CHANNEL);
```

With these steps you are ready to control the PWM pins. To set a value you can call:

```
ledcWrite(CHANNEL,  VALUE);
```

You also can read back the current value from a given ledc channel using:

```
ledcRead(CHANNEL)
```

These are the basics to control the ‘ledc’ unit. Inside the software module we have the four physical LED channels and a fifth one being pure software for overall brightness control of the channels. In the web interface it’s called ‘intensity’. If you change the value of this channel it doesn’t get set immediately. Instead, every 100 ms a function is called from a timer and checks if the set value is equal to the one currently used. If not, it is incremented or decremented at a given maximum step size. This results in some fading you see if you change the intensity level. This is applied to all channels at the same time; at startup the intensity will slowly increase.

The timekeeping module from the ESP32 Bedroom Clock is also added with a few modifications. Some plants need a day and night cycle, so we built in a function that can turn off the LEDs for a given period of the day.

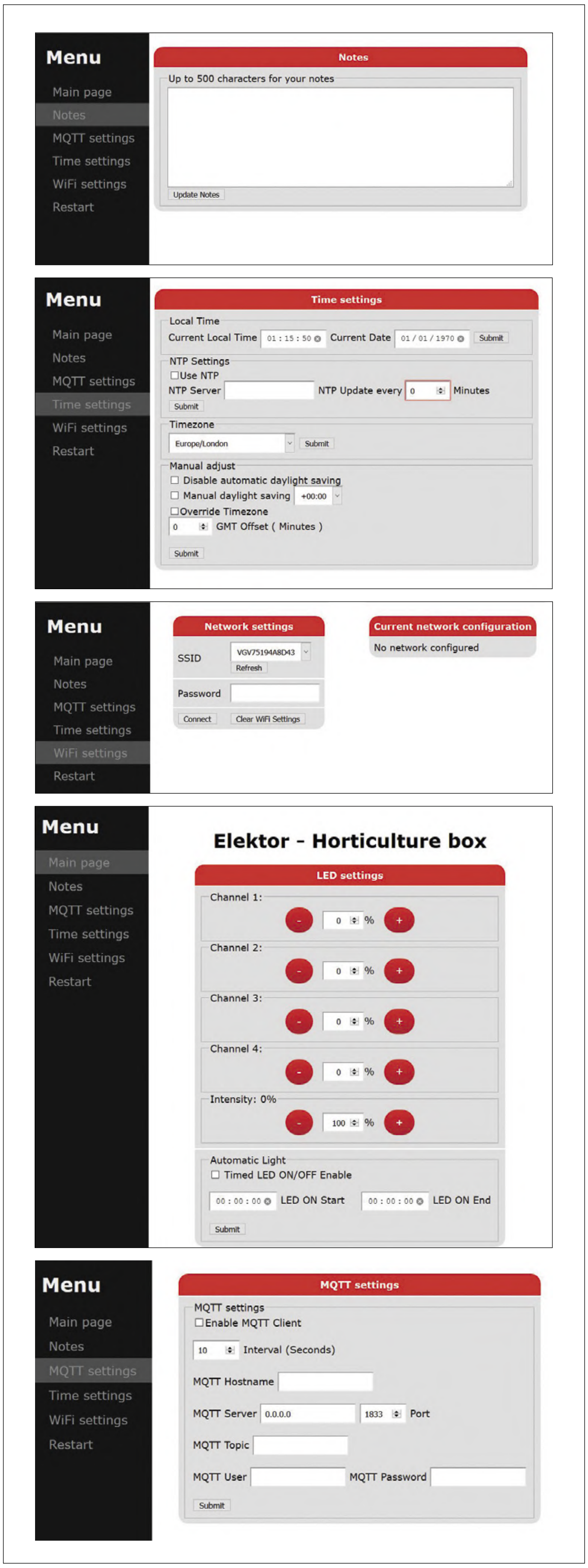
A web interface 4 sure

As you can see from the screenshots collected in **Figure 4**, the UI (user interface) is a rehash from earlier Elektor projects. The main page has five fields to change the PWM settings. Channels 1 to 4 are the physical channels of the grow light, while ‘Intensity’ controls the overall luminosity. Plus or minus will change a value by 1%, alternatively you can enter a value between 0% to 100%. In the background communication a change of values is carried out using web sockets. If more than one person changes settings all others will get the new values pushed to their browser. This is done in the code with a separate web socket server that sends the messages as JSON while also processing it as JSON.
There is also a part to enable or disable the timed mode for the LEDs.

Within ‘Time settings’ the internal clock can be set manually or select an NTP server to have the system time adjusted automatically.

In Wi-Fi Settings you can set the credentials of your local wireless network. This allows you to control the grow light with any other device connected to this network.

The MQTT part may be new to many of you. This got borrowed from the Weather Station and provides a way to control the LED settings and the automatic on/off time from an MQTT broker. This is currently done with the PubSubClient library and ArduinoJson for data exchange. To send new settings to the system you need to create a JSON string that looks like this:



Notes
Up to 500 characters for your notes

Local Time
Current Local Time 01 : 15 : 50 Current Date 01 / 01 / 1970
NTP Settings
☐ Use NTP
NTP Server NTP Update every 0 Minutes
Timezone
Europe/London
Manual adjust
☐ Disable automatic daylight saving
☐ Manual daylight saving +00:00
☐ Override Timezone
0 GMT Offset (Minutes)

SSID VGV75194A8D43
Password
No network configured

Elektor - Horticulture box
Channel 1: 0 %
Channel 2: 0 %
Channel 3: 0 %
Channel 4: 0 %
Intensity: 0% 100 %
Automatic Light
☐ Timed LED ON/OFF Enable
00 : 00 : 00 LED ON Start 00 : 00 : 00 LED ON End

MQTT settings
☐ Enable MQTT Client
10 Interval (Seconds)
MQTT Hostname
MQTT Server 0.0.0.0 1833 Port
MQTT Topic
MQTT User MQTT Password

Figure 4: Screenshots of the User Interface developed for the Horticulture Box: main page, MQTT page, notes, time settings, and Wi-Fi. The UI is basically a rehash of a version developed for two other Elektor projects.

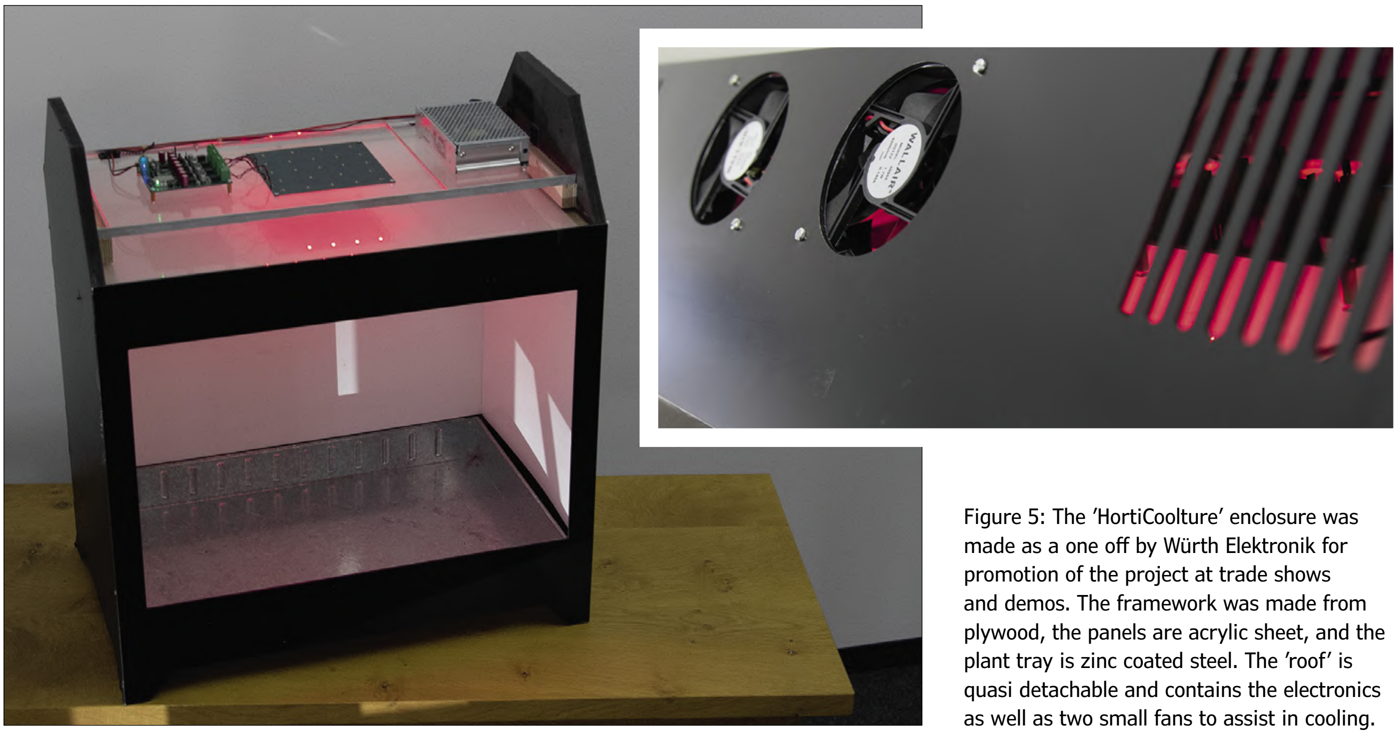


Figure 5: The 'HortiCoolture' enclosure was made as a one off by Würth Elektronik for promotion of the project at trade shows and demos. The framework was made from plywood, the panels are acrylic sheet, and the plant tray is zinc coated steel. The 'roof' is quasi detachable and contains the electronics as well as two small fans to assist in cooling.

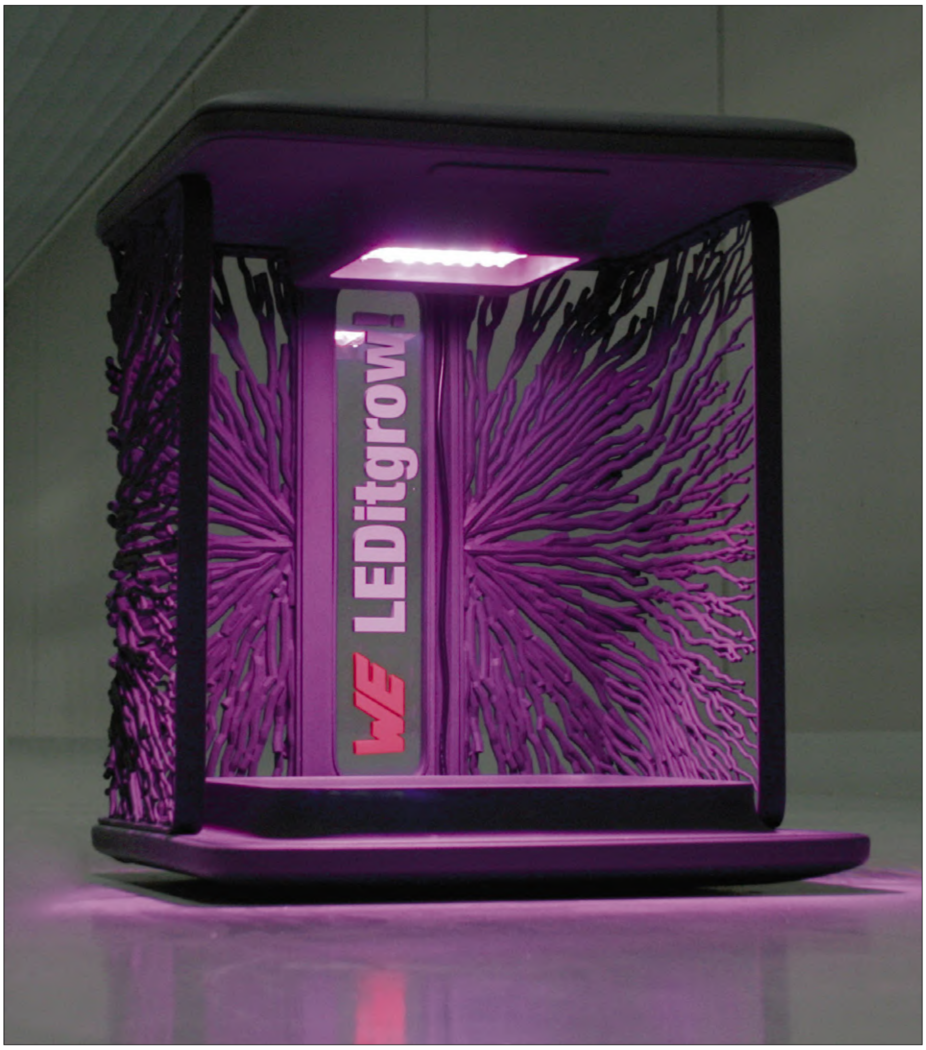


Figure 6. The latest version of the Horticulture Box comes out of a 3D printer. (product and image: Würth Elektronik)

```
{
    "light": {
        "ch0": 6400,
        "ch1": 0,
        "ch2": 0,
        "ch3": 0,
        "intense": 32000
    },
    "timer": {
        "enable": false,
        "start": {
            "hour": 0,
            "minute": 0,
            "second": 0
        },
        "end": {
            "hour": 0,
            "minute": 0,
            "second": 0
        }
    }
}
```

16-bit unsigned values are allowed for 'ch0' to 'intensity'. Larger numbers are automatically limited to 65535, for the minimum the limit is 0. This allows, say, NodeRed to take over the control of the lighting to furnish the plants not only with a day light cycle but possibly also over the day to provide different colour blends. Not all fields have to be available to change the values. For example, to change the value of 'ch0' only:

Web Links

[1] Würth Horticulture LEDs Application Notes: https://katalog.we-online.de/en/led/WL-SMDC_HORTICULTURE

[2] Arduino IDE for ESP32: www.elektormagazine.com/labs/esp32-getting-started

[3] Horticulture Box firmware download: www.elektormagazine.com/180583-01

[4] ESP32 Bedroom Clock: www.elektormagazine.com/labs/bedroom-clock-with-out-side-temperature-based-on-esp32

[5] ESP32 Weather Station: www.elektormagazine.com/magazine/elektor-70/42351

```
{
    "light": {
        "ch0": 6400
   }
}
```

This only sets the first channel to 6400. To make this possible, the code for processing the MQTT messages checks whether individual objects are contained. If we use ArduinoJson the code is as follows:

```
JsonObject light = doc["light"];
 JsonVariant light_ch0_var = light["ch0"];
    If (false == light_ch0_var.isNull() ){
    ………
```

In the IF statement we can check the presence of a value for a certain channel. If it's found then it will be processed, else we simply ignore it. If you need to process JSON yourself it is better to check if the values you need are really inside the JSON, or you may end up using unintended defaults.

LED it grow

Our grow light is installed in a case that can be used as a *paludarium*, but any other mechanical construction that holds the LED PCB above your cherished plants will do fine. **Figure 5** shows views of Würth Elektronik's "HortiCoolture" enclosure, custom developed for promotional activities. It has two small fans in the top part to assist in cooling the LED board. **Figure 6** shows a later version designed and made by Würth Elektronik from 3D printed parts.
Custom building such a nice looking enclosure maybe beyond your reach but as always there are cheaper alternatives.

Consider:

- 'picking' an aquarium or herbarium at your local thrift shop;
- visiting Ikea;
- dropping by at aquarium & pond store.

Dutch readers in particular should consider their 'Grow Shops' as potential buyers of the project rather than parts suppliers. Although there are a number of scientific papers on optimal light settings for specific plants you are encouraged to find your own 'light recipes'. ⏮

180583-01

@ WWW.ELEKTOR.COM

→ Horticulture Box driver and LED board, preassembled, including programmed ESP-32-W-ROVER-B module
www.elektor.com/180583-71

→ Horticulture Box Control/Driver board, bare PCB 180593-1 v.2.01
www.elektor.com/180583-1

→ Horticulture Box LED board, bare PCB 180593-2 v1.1
www.elektor.com/180583-2

Anzeige

Electronic components
and printed circuit boards
for a successful growth!

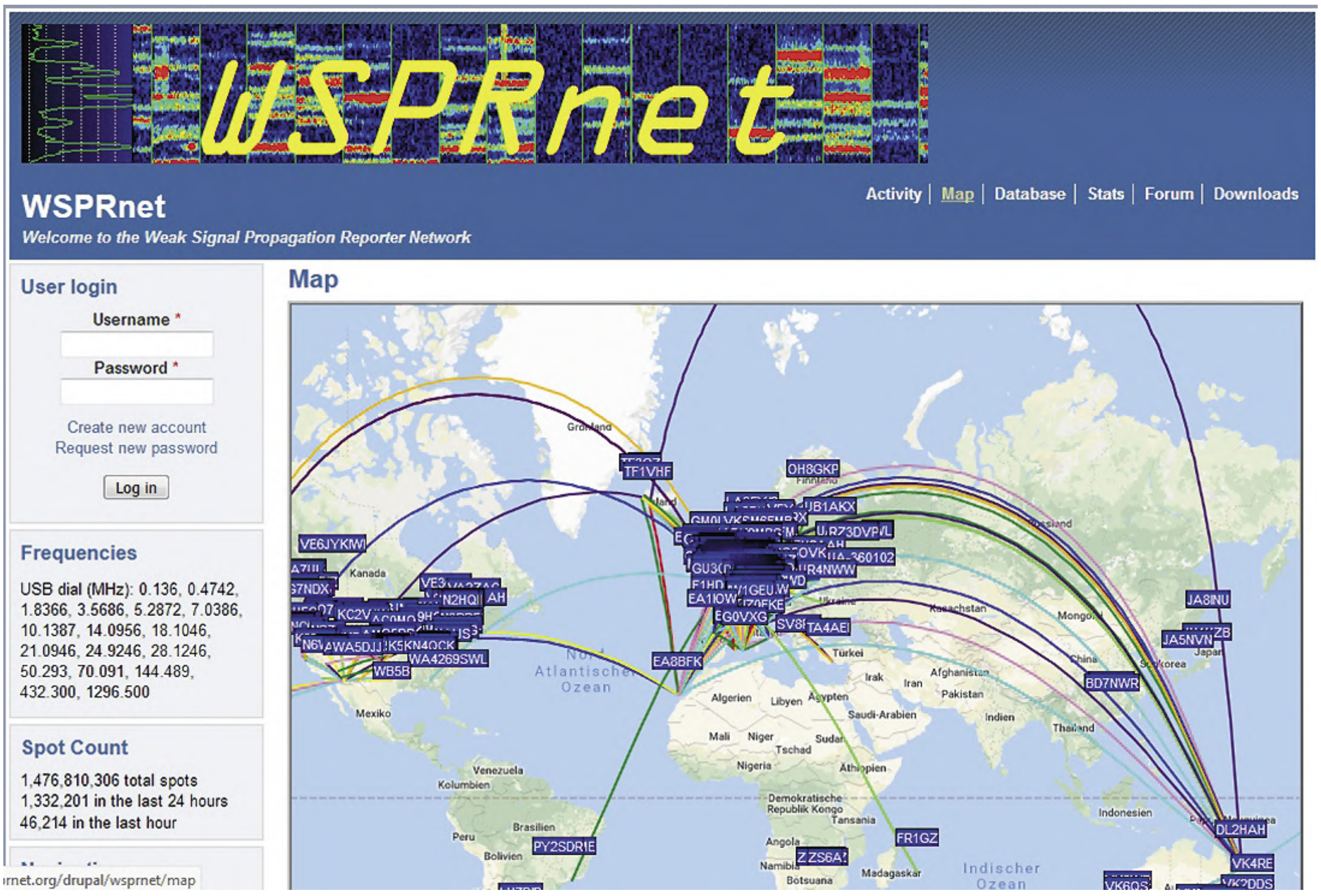
www.wedirekt.com
www.we-online.com

HOMELAB PROJECT

Transmitting WSPR Reports

Reach thousands of miles with the Elektor SDR Shield

By **Burkhard Kainka** (Germany)



Frequencies

USB dial (MHz): 0.136, 0.4742, 1.8366, 3.5686, 5.2872, 7.0386, 10.1387, 14.0956, 18.1046, 21.0946, 24.9246, 28.1246, 50.293, 70.091, 144.489, 432.300, 1296.500

Spot Count

1,476,810,306 total spots
1,332,201 in the last 24 hours
46,214 in the last hour

The WSPR (Weak Signal Propagation Reporter, pronounced 'whisper') radio transmission technique was developed to enable considerable distances to be spanned using low power and minimum bandwidth. This article shows how you too can transmit WSPR reports using the Elektor SDR Shield and free software.

There are always plenty of active WSPR stations, which list their reception results on the Wsprnet website [1]. In no time at all you will gain a good idea where signals can be received. 600 miles with just 10 mW is entirely feasible. Using 100 mW, you can cover even larger distances without difficulty.
The program you need is WSPR 2.0 [2] by Joe Tailor, who developed WSPR. The transmitted signal is essentially a constant tone of 1.5 kHz, which is usually transmitted with an SSB transceiver. If you listen carefully, you will notice very small differences again and again. In fact, WSPR uses a 4FSK signal with four frequencies that are 1.46 Hz apart. Each frequency remains stable for about one second. In consequence, the receiver software provides extremely narrowband filtering and thus achieves a good signal-to-noise ratio.
We have already covered WSPR reception technology with the Elektor SDR Shield in Elektor [3]. Now it's time to discuss transmitting techniques. Included in this, we cover the Arduino software with the WSPR encoder, as well as a small RF output

stage and antenna matching. The transmission tests described here are legal in this form only for licensed radio amateurs. For readers who are also interested but don't have an amateur radio licence, you can nevertheless carry out small (low-power) experiments receiving your own signal without connecting a transmitting antenna.

Controlling the SDR Shield

The SDR Shield is normally used as a shortwave receiver. However, the onboard Si5351 PLL generator also provides two outputs that can be used either for test and measurement functions or for driving a transmitter. You can also generate a WSPR signal using these. This and other kinds of experiment are described in *The SDR Hands-On Book* [4].
In his work on the SDR book and for the first articles about the SDR Shield, the author used the older Etherkit Si5351 library by Jason Milldrum in its Version 1.1.2. Today new users will encounter the newer version 2.1.4, however. Combining the newer library version and the original source code might give you error messages, because some structures have changed slightly.

The initialisation process now requires an extra parameter 0, whereby the additional 0 stands for the fact that, at this point, you do not want to specify the quartz frequency more precisely (in terms of deviating from 25 MHz).

```
si5351.init(SI5351_CRYSTAL_LOAD_8PF, 0, 0);
```

The actual calibration must follow after the initialisation, where `...INPUT_XO` indicates that it refers to the crystal oscillator:

```
si5351.set_correction(162100, SI5351_PLL_INPUT_XO);
```

The frequency output now manages without the previous parameter PLL_FIXED, because the library automatically determines the PLL settings:

```
si5351.set_freq(freq*100ULL, SI5351_CLK0);
```

All sample programs from the book have been revised in the meantime, including the applications already presented in our first series of articles on the SDR Shield. The software archive for the book [4] now contains both the original and the updated versions.

Listing 1: Initialisation with si5351vfo2_1.ino.

```
{
  Serial.begin(9600);
  Serial.println("Si5351 Clockgen"); Serial.
println("");
  si5351.init(SI5351_CRYSTAL_LOAD_8PF, 0, 0);
  si5351.set_correction(162100,
SI5351_PLL_INPUT_XO);
  Serial.println(10100);
  si5351.set_freq(4040000000ULL,  SI5351_CLK1);
  si5351.output_enable(SI5351_CLK1, 1);
  freq = 10100;
  lcd.begin(16, 2);
  lcd.print(freq);
}
```

To begin, we need only the tuning program si5351vfo2_1 (**Listing 1**). The VFO frequency of 7038.6 kHz is set here (**Figure 1**), in order that WSPR signals in the 40-m band are mixed down to 1.5 kHz. Output A can be switched on to check the correct setting. A signal at 7040.1 kHz then couples weakly into the input and in WSPR 2.0 can be seen exactly in the middle of the 200 Hz-wide reception range (**Figure 2**). However, the calibration of the receiver frequency is generally not sufficiently precise.
You can see that the carrier is being received, confirming that some weak capacitive coupling exists. Now only the modulation is missing, i.e. the selective switching between the four adjacent frequencies of the 4FSK signal.

The JTEncode library

Another piece of Jason Milldrum's handiwork is the JTEncode library [5] for producing WSPR and other digital signals with the Si5351. We used his sample program to develop a first program for the SDR Shield. For this purpose, the receiver is locked to the WSPR frequency in the 20-m band (VFO = 14095.600 kHz). The transmitter operates in the same band (14097.100 kHz) and starts whenever Pin 12 of the Arduino is taken to GND. You will then be able to hear your own signal at 1.5 kHz.
For use with the program Si5351WSPR1.ino (**Listing 2**) we invented the fantasy radio callsign of EL2SDR, which should be

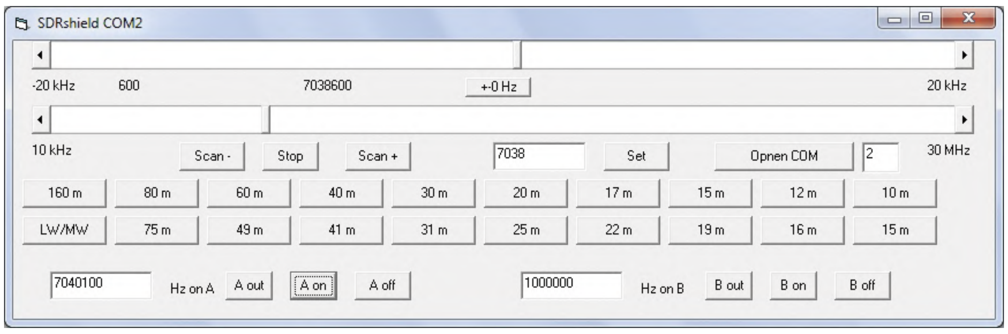


Figure 1: Setting the VFO frequency.

Listing 2: Transmitting WSPR signals (excerpt from Si5351WSPR1.ino).

```
// Simple JT65/JT9/WSPR/FSQ beacon for Arduino, with the Etherkit
// Si5351A Breakout Board, by Jason Milldrum NT7S.

#include
#include
#include
#include
#include "Wire.h"
#define BUTTON                    12
#define LED_PIN                   13

…
char message[] = "EL2SDR JO31";
char call[] = "EL2SDR";  //das Rufzeichen
char loc[] = "JO31";    //der Lacotor wie z.B. JO31";
uint8_t dbm = 10;       //10 mW, 30 für 1W, 37 für 5 W
uint8_t tx_buffer[255];
…

void encode()
{
  uint8_t i;
  // Clear out the old transmit buffer
 memset(tx_buffer, 0, 255);
 jtencode.wspr_encode(call, loc, dbm, tx_buffer);
  // Reset the tone to the base frequency and turn on the output
  si5351.output_enable(SI5351_CLK0, 1);
  digitalWrite(LED_PIN, HIGH);
  for(i = 0; i < symbol_count; i++)
  {
      si5351.set_freq((freq * 100) + (tx_buffer[i] * tone_spacing), SI5351_CLK0);
      Serial.print (tx_buffer[i]);  Serial.print (",");
      proceed = false;
      while(!proceed);
  }
  // Turn off the output
  si5351.output_enable(SI5351_CLK0, 0);
  digitalWrite(LED_PIN, LOW);
}


void loop()
{
  if(digitalRead(BUTTON) == LOW)
  {
    delay(50);   // delay to debounce
    if (digitalRead(BUTTON) == LOW)
    {
      encode();
      delay(50); //delay to avoid extra triggers
    }
  }
}
```

used only for making indoor experiments without any antenna. In addition, we need to declare a notional 'European locator code' for the transmitter (JO31) and transmitter power (10 dBm). With this data the encode function creates a transmit buffer with the WSPR symbols 0, 1, 2 and 3, which denote the four closely spaced frequencies separated by 1.46 Hz.

```
jtencode.wspr_encode(call, loc, dbm, tx_buffer);
```

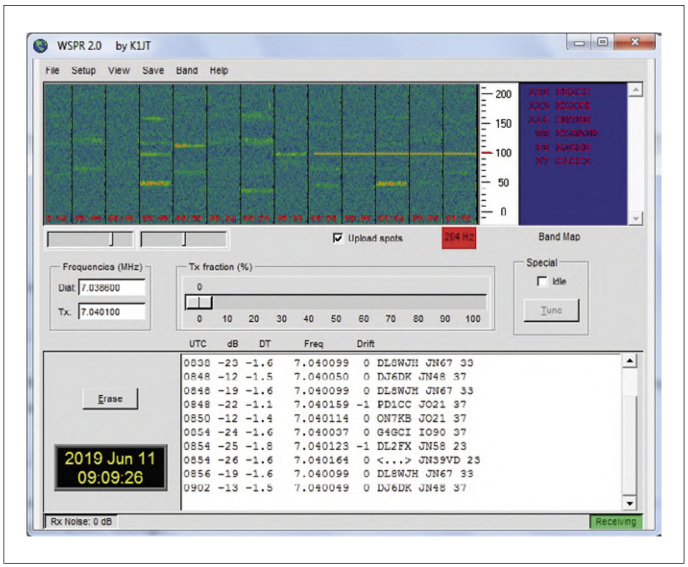
A typical buffer consist looks like this:

```
1,3,0,2,2,2,2,2,1,0,2,0,1,3,1,0 …
```

This then controls the frequency of the FSK signal during the actual transmission process (Listing 2).
A transmission cycle lasts two minutes and always starts on an even minute. For successful operation, the timing must be adhered to precisely. Two seconds after the start of an even minute, your own transmitter is started by a button push on Pin 12.
The receiver can be equipped with an antenna and can receive actual (real) WSPR stations. The transmitter output remains independent and is not connected to an antenna. The signal couples capacitively, with adequate strength, from output A to the antenna input and is received correctly by WSPR 2.0. **Figure 3** shows the result of this experiment. My own signal was received with 12 dB above the noise. In addition, a station from Italy was decoded with -26 dB, which was using a transmit power of 37 dBm (5 watts).

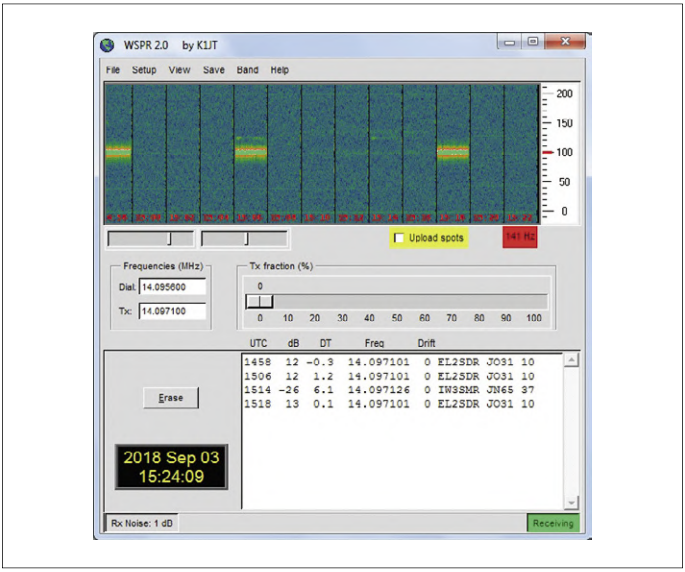
Calibration and timed starting

For WSPR operation on all bands, extended firmware si5351vfo2_WSPR.ino and a corresponding PC application program SDRshield_WSPR have been written. The exact frequency is crucial. For this reason, the author has integrated a calibration function. A shortwave broadcast transmitter is chosen as the frequency standard. Using the slider control, you tune the receiver precisely to the frequency of the selected broadcast transmitter. The carrier should be exactly in the 5 kHz grid in the spectrum of the software SDR# [6]. You can check the setting with other radio transmitters, which in most cases operate very precisely. Using Save you can transfer the



```
UTC   dB   DT     Freq     Drift
0838 -23 -1.6   7.040099  0 DL8WJH JN67 33
0848 -12 -1.5   7.040050  0 DJ6DK JN48 37
0848 -19 -1.6   7.040099  0 DL8WJH JN67 33
0848 -22 -1.1   7.040159 -1 PD1CC JO21 37
0850 -12 -1.4   7.040114  0 ON7KB JO21 37
0854 -24 -1.6   7.040037  0 G4GCI IO90 37
0854 -25 -1.8   7.040123 -1 DL2FX JN58 23
0854 -26 -1.6   7.040164  0 <...> JN39VD 23
0856 -19 -1.6   7.040099  0 DL8WJH JN67 33
0902 -13 -1.5   7.040049  0 DJ6DK JN48 37
```

Figure 2: WSPR signals and the auxiliary carrier A.



```
UTC   dB   DT     Freq     Drift
1458  12 -0.3  14.097101  0 EL2SDR JO31 10
1506  12  1.2  14.097101  0 EL2SDR JO31 10
1514 -26  6.1  14.097126  0 IW3SMR JN65 37
1518  13  0.1  14.097101  0 EL2SDR JO31 10
```

Figure 3: Receiving your own signal.

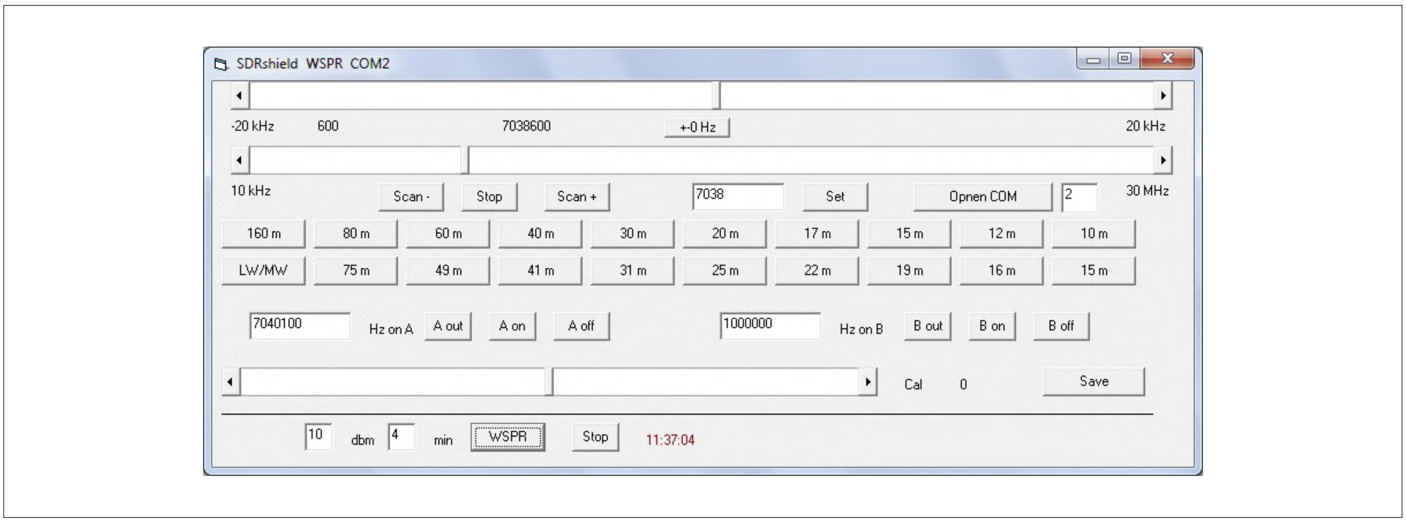
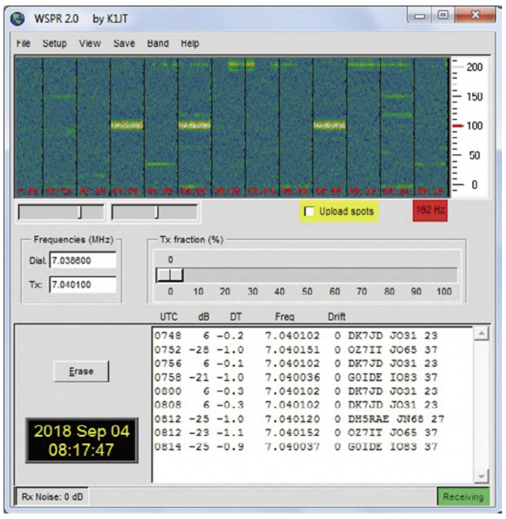


Figure 4: The WSPR frequency of 7038.600 kHz in the 40-m band.

Listing 3: Transmitter control.

```
    if (ch == 119) {       // w, Start wspr
      dbm = number;
      encode();
    }
   …
    if (ch == 102) {        //f
      si5351.set_freq(freq*400ULL, SI5351_CLK1);
      wsprfreq = freq;
    }

void encode()
{
…
      si5351.set_freq((wsprfreq * 100)+ 150000 + (tx_buffer[i] * tone_spacing),
                  SI5351_CLK0);
…
```



```
UTC   dB   DT     Freq     Drift
0748   6 -0.2   7.040102   0 DK7JD JO31 23
0752 -28 -1.0   7.040151   0 OZ7IT JO65 37
0756   6 -0.1   7.040102   0 DK7JD JO31 23
0758 -21 -1.0   7.040036   0 GOIDE IO83 37
0800   6 -0.3   7.040102   0 DK7JD JO31 23
0808   6 -0.3   7.040102   0 DK7JD JO31 23
0812 -25 -1.0   7.040120   0 DH5RAE JN68 27
0812 -23 -1.1   7.040152   0 OZ7IT JO65 37
0814 -25 -0.9   7.040037   0 GOIDE IO83 37
```

Figure 5: Receiving your own signal.

correction factor to the Arduino.
The Visual BASIC application program was modified in such a way that the shortcut keys for the amateur radio bands no longer relate to the beginning of the band but to the corresponding WSPR frequency. The VFO is set to 7038.600 kHz by a click on 40 m. The VFO is then set to 7038.600 kHz.
A click on the WSPR button activates the transmit function (**Figure 4**). The transmitter will then start on the next even minute and after a preset interval of 2 (default), 4, 6, 8 or 10 minutes, depending on the input. The time is also displayed in red during the active phases.
The current receive frequency is assigned to the transmit frequency in the Arduino sketch. The w command starts the transmit process, so that you are no longer reliant on a start pulse at Pin 12. The single parameter of the current transmission power in dBm is also transferred across. During transmission, 1500 Hz and the respective modulation frequency are added to the frequency in wsprfreq (**Listing 3**).

The first test on the 40-m band was again carried out without a transmitting antenna, without connection to output A. **Figure 5** shows some of the stations captured on the receive antenna and in among them our own signals that were generated with the author's own callsign DK7JD. Although a wire-free connection exists between transmitter and receiver, its range of just 2 cm is minimal.

A 200-mW transmit amplifier

Figure 6 illustrates a small transmit amplifier using a BS170 MOSFET. This provides up to 200 mW (23 dBm) output with an operating voltage of 5 V. If you wish, you can reduce the voltage to 3.3 V, in order to produce only 100 mW (20 dBm). A series resistor of 100 Ω further reduces the power by 10 dB, so that we now deliver 10 mW (10 dBm) or correspondingly 20 mW (13 dBm) to the antenna. The output power is now variable across broad limits:

- 5 V: 200 mW
- 3.3 V: 100 mW
- 5 V via 100 Ω: 20 mW
- 3.3 V via 100 Ω: 10 mW

The lowpass filter at the output of the transmitter is optimised for operation on the 30-m and 20-m bands. The 0.5 µH filter chokes are wound as air-cored coils with ten turns of 5 mm diameter. **Figure 7** illustrates the compact dimensions of the transmitter.
The lowpass filter was designed for a typical operating frequency of 10 MHz. For verification purposes, the SDR Shield was tested with a four-pole measurement program.
The test result **(Figure 8**) shows an upper cut-off frequency of 18 MHz. Thus, the 20-m band at 14 MHz can also be used and, with restrictions, the 17-m band at 18 MHz. The lower usable limit of the transmit amplifier is about 2 MHz. However, sufficient harmonic attenuation can be expected only from 10 MHz onwards, allowing a suitable antenna to be connected directly. The lower cut-off frequency at around 2 MHz is due to the relatively low inductance of 3.3 µH in the drain circuit of the amplifier.

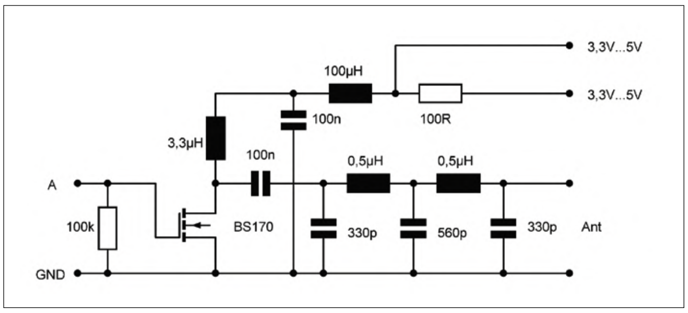


Figure 6: 200-mW output stage using a BS170 FET.

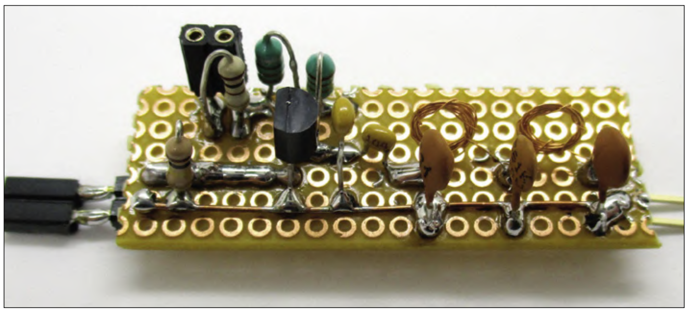


Figure 7: FET output stage with filter.

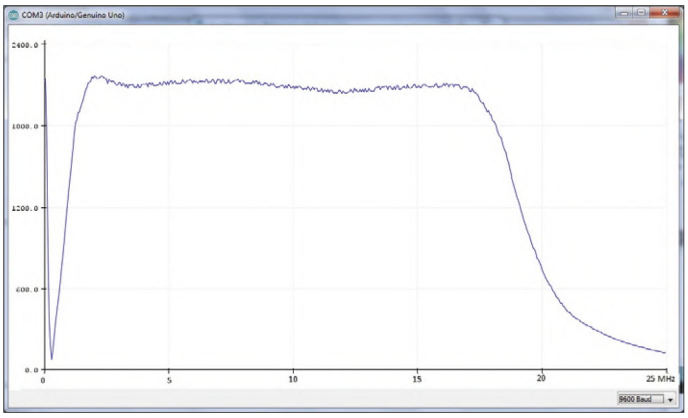


Figure 8: Frequency response of the output stage.

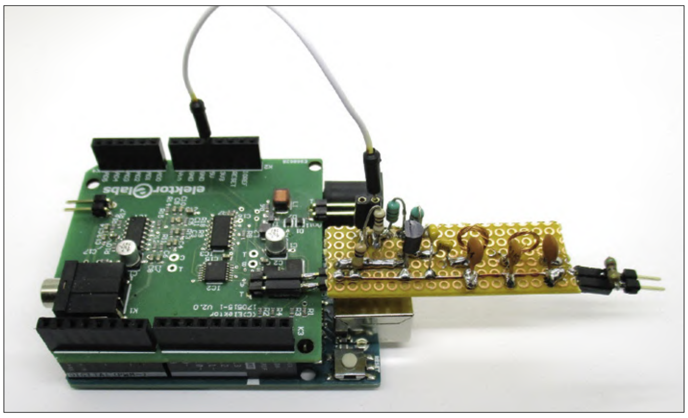


Figure 9: Operation with 5 V and a terminating resistor.

Examination with the oscilloscope demonstrates the effectiveness of the lowpass filter and the achievable output power. For this purpose, the output stage was operated with 5 V and provided with a terminating resistor of 51 ohms **(Figure 9)**. The load resistor has a voltage of 8 Vpp, i.e. an amplitude of 4 V. From this the power can be calculated:

$P = U^2 / 2R = 4^2 / 100 = 160$ mW

Tests have shown that even more power can be achieved with a slightly higher operating voltage up to 7 V. The oscillograms in **Figures 10 to 12** show the increasingly better filtering up to the antenna output.

All in all, every band from 160 m to 20 m can be used with the amplifier. However, you do need to provide additional harmonic suppression. This can be done relatively easily with an antenna adapter in the form of a Pi filter.

Antenna matching

Using an antenna matching device, you can also use antennas of the 'wrong' length and, as such, are therefore not in resonance. Only an optimal dipole antenna has a real base resistance close to 50 Ω. In all other cases you are battling with a higher impedance and an additional capacitive or inductive reactive component.

With a Pi filter **(Figures 13 and 14)** you match almost any antenna. It uses two variable capacitors and a coil with taps, the inductance of which can be changed with a shorting jumper. Depending on the setting, the voltage at the output can be made higher or lower than at the input. Therefore, both low-impedance and high-impedance antennas can be matched. The Pi filter transforms the impedance of the antenna and compensates for any remaining reactance.

At the same time, the Pi filter is a lowpass filter and attenuates

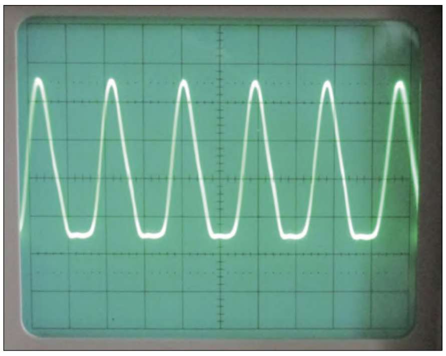


Figure 10: Signal at the drain of the FET.

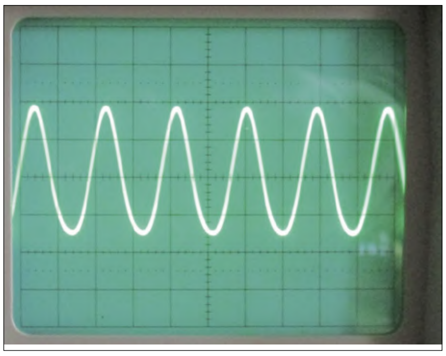


Figure 11: At the centre of the filter.

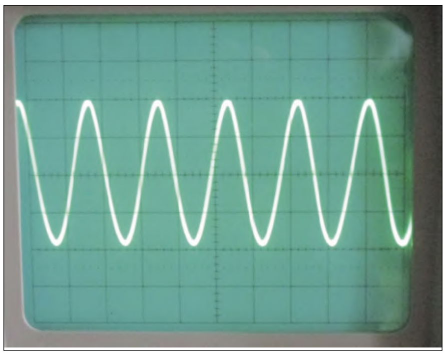


Figure 12. At the terminating resistor.

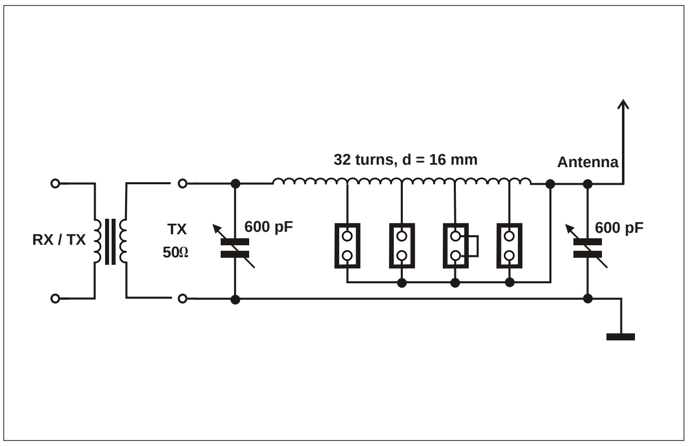


Figure 13. Pi filter with transducer.

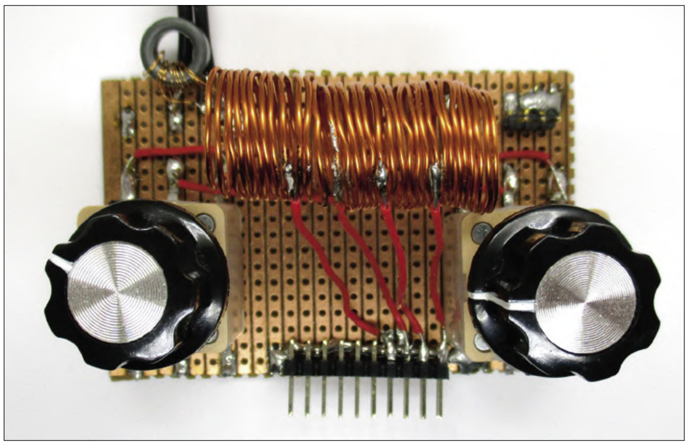


Figure 14. Construction using an air-cored coil and variable trimmer capacitors (film dielectric type).

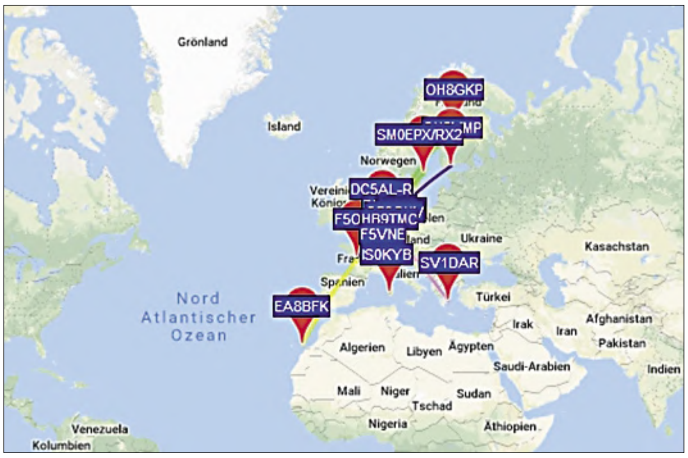
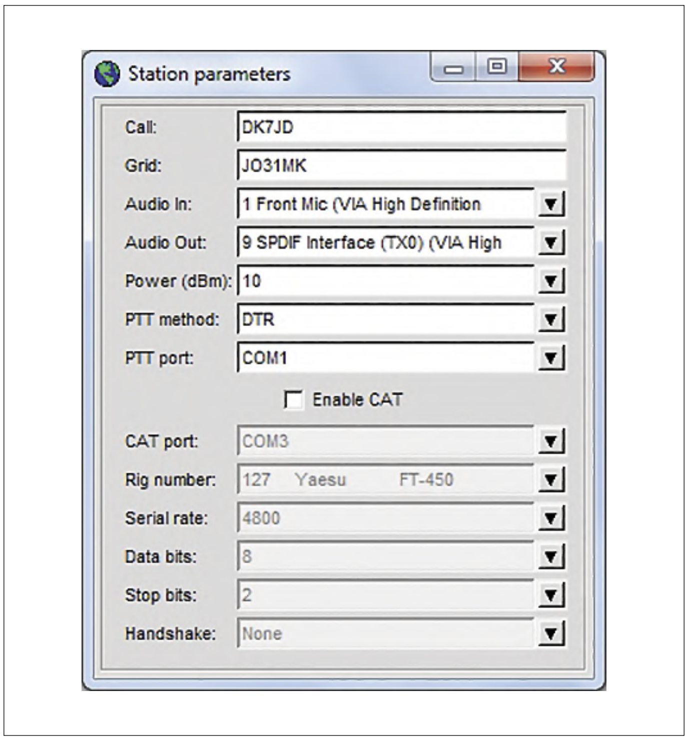


Figure 15. WSPR reception locations using 200 mW.



Station parameters

Call:	DK7JD
Grid:	JO31MK
Audio In:	1 Front Mic (VIA High Definition
Audio Out:	9 SPDIF Interface (TX0) (VIA High
Power (dBm):	10
PTT method:	DTR
PTT port:	COM1
	☐ Enable CAT
CAT port:	COM3
Rig number:	127 Yaesu FT-450
Serial rate:	4800
Data bits:	8
Stop bits:	2
Handshake:	None

Figure 16. Software settings.

harmonics of the transmitted signal. For optimum filtering, the highest possible quality factor or Q must be achieved. However, this is not possible at every frequency if the Pi filter, with its impedance of about 50 Ω, is connected directly to the output stage. Therefore, an additional broadband transformer with a turns ratio of 1:4 was used. The output voltage is stepped up for an optimal match to the Pi filter. In this way, the WSPR transmitter can also be used on 80 m, 60 m and 40 m. An oscilloscope is used as a tuning aid to adjust the antenna signal for the highest voltage. At the same time, you are able to detect gross errors in the filtering. In this case you can see a significant deviation from the sinusoidal wave shape, indicating excessively high harmonic content.

With the 200 mW available from this small WSPR transmitter, it was possible to transmit successfully on all bands from 80 m to 20 m. The author was able to reach almost the whole of Europe and cover distances of up to 3000 km **(Figure 15)**. In other experiments, stations in the USA, Canada and Australia were also reached with similarly low power level. To achieve this, the transmitter should be active on the appropriate frequency for only as long as necessary. The general fluctuations of the propagation conditions then lead to a successful connection briefly every now and then. An experiment with only 10 mW on 40 m achieved distances up to 600 km.

Web Links

[1] Weak Signal Propagation Reporter Network: http://wsprnet.org/

[2] WSPR 2.0 Software: https://physics.princeton.edu/pulsar/K1JT/wspr.html

[3] Elektor SDR-Shield 2.0 (2), Elektor magazine 5/2018: https://www.elektormagazine.com/180284-02

[4] SDR Hands-On Book with Software Archive: http://www.elektor.com/sdr-hands-on-book

[5] JTEncod Library: https://github.com/etherkit/JTEncode

[6] SDR# Software: https://airspy.com

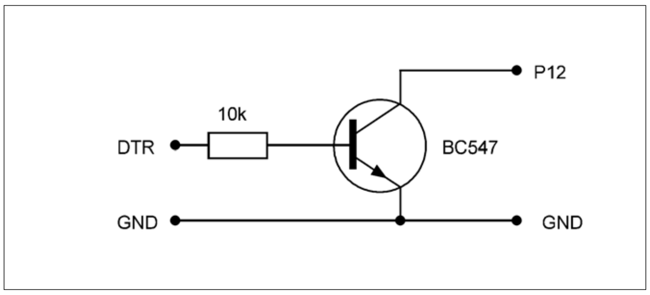
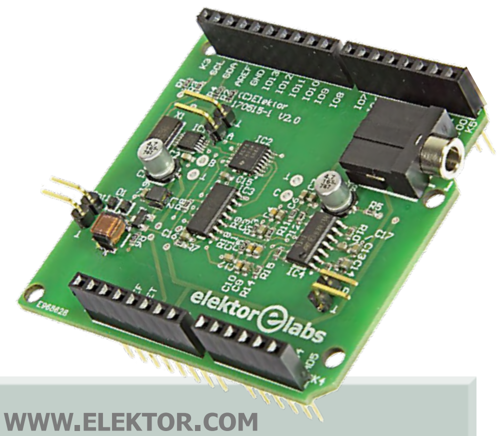


Figure 17. Start interface.

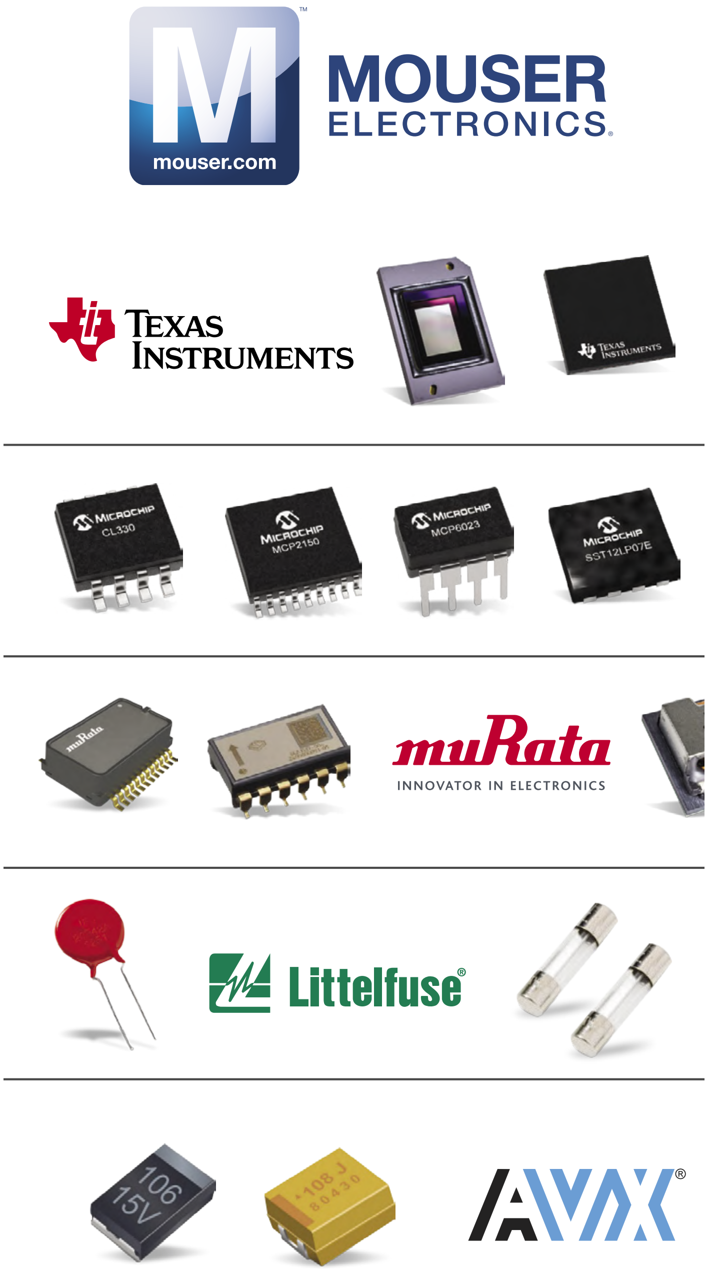
CAT control

The WSPR 2.0 program is preconfigured for CAT (Computer-Aided Transceiver) control of numerous amateur radio devices that have an appropriate interface. The corresponding frequency is set and the transceiver is switched to transmit. The software then generates the modulation in the range 1400 to 1600 Hz and controls the microphone input of the SSB transmitter. All necessary settings and parameters are entered in the Setup/Station-Parameters menu **(Figure 16)**.
Fully automatic control of the SDR shield is not possible, because it is not an SSB transmitter. But automatic starting can also be achieved using the PTT method. The DTR setting using the COM port COM1 causes the DTR output at the serial interface COM1 to be incremented in each transmission phase in order to switch on a transmitter. This signal can be employed to initiate the transmitter. All that's necessary is an inverter stage **(Figure 17)**, which controls the start input P12 on the Arduino.
With this hookup, the WSPR software controls the transmission phases. Your own signal is no longer received then. In addition, the time intervals between the transmission phases can be set, also dispersed randomly. This prevents remote stations from constantly transmitting simultaneously over a longer period of time and therefore being unable to receive each other.

180709-02



Advertisement



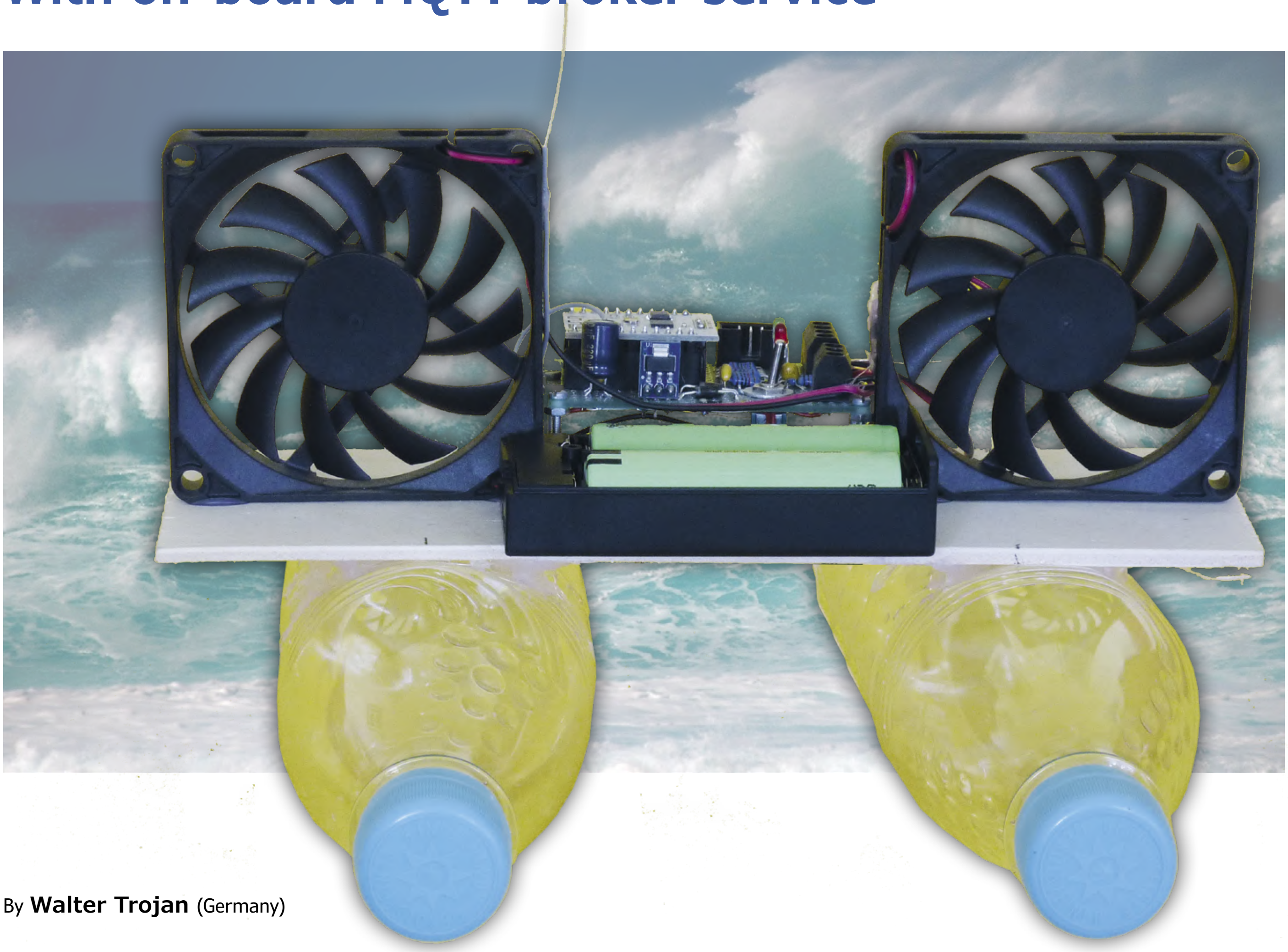
One source for your entire BOM

Newest and widest selection of electronic components in stock

HOMELAB PROJECT

RC Catamaran Uses IoT Protocol

With on-board MQTT broker service



By **Walter Trojan** (Germany)

Often the easiest way to get acquainted with new technology such as IoT hardware and software is to apply it in a practical way to build something useful; that way learning gets to be fun. In this project the author builds a remotely controlled model for his grandchildren; they get lots of fun out of using the finished product and also helped him build the boat. It uses some empty drinks bottles, a couple of old PC fans and some small controller boards such as the ESP8266 to handle the control functions, set up a Wi-Fi network and even provide an MQTT broker service!

stts"Grandpa, your inventions are really cool, when can you make something new for us?" This request by my grandchildren interrupted my musings on voice control, artificial intelligence and the like, and brought me back to earth. Not easy to come up with something original, I have already built robots, traffic lights and other things for them. A rummage through my box of discarded electronic parts often gives me some inspiration. This time I found a couple of PC fans and a remote controller for a toy RC helicopter. Now a vague idea was beginning to take shape in my head, I needed some sort of toy the children would be able to help construct… how about a model boat powered and steered by the two fans? I was also keen to use the opportunity to learn something new and build on my knowledge. I had just been reading something about an MQTT broker running on the ESP8266, maybe I could try that out. Yep, you read that correctly: An MQTT broker running on the small ESP8266 with its very limited main memory. Is that even possible? Let's see…

Quick Features

- Communication between catamaran and remote control via MQTT Protocol for exchange of commands and status information
- Own WLAN for the independence from the place of use and Dedicated MQTT-Broker
- Comfortable control with one (converted) console
- Monitoring of catamaran and console battery voltages
- Using the Arduino IDE as an uncomplicated development environment for the software
- Simple and cost-effective mechanical construction

The mechanics? They're a breeze

As far as the mechanics go the catamaran construction really is just about as simple as it gets: Two medium-sized PET bottles, hot-glued to a small square of plywood does the job. Thrust is provided by two PC fans, which are again attached using hot glue. Power comes from two 18650 LiPo batteries, each with a capacity of 3400 mAh; that supplies enough energy for a reasonable period of fun and games. Next to the battery box in the middle, the main board is fitted with an ESP8266. The cover picture shows the whole package and proves that you really can build technically demanding toys with very little outlay indeed. The craft is actually quite stable and moves through the water fairly quickly powered and steered by the two fans (no reverse).

Catamaran control using the versatile ESP8266

In this project, an ESP8266 Wi-Fi module is used to control the catamaran while its bigger brother the ESP32 module is used in the remote control handset to read the control paddle positions and send the control signals to the catamaran. Both of these processor modules are produced by the Chinese semiconductor company Espressif; and are equipped with one or two 32-bit processor cores offering a range of peripheral features. As both these MCUs have been used in previous Elektor projects already I will not go into a detailed description and only refer to the available literature [1] and the information available on the net.

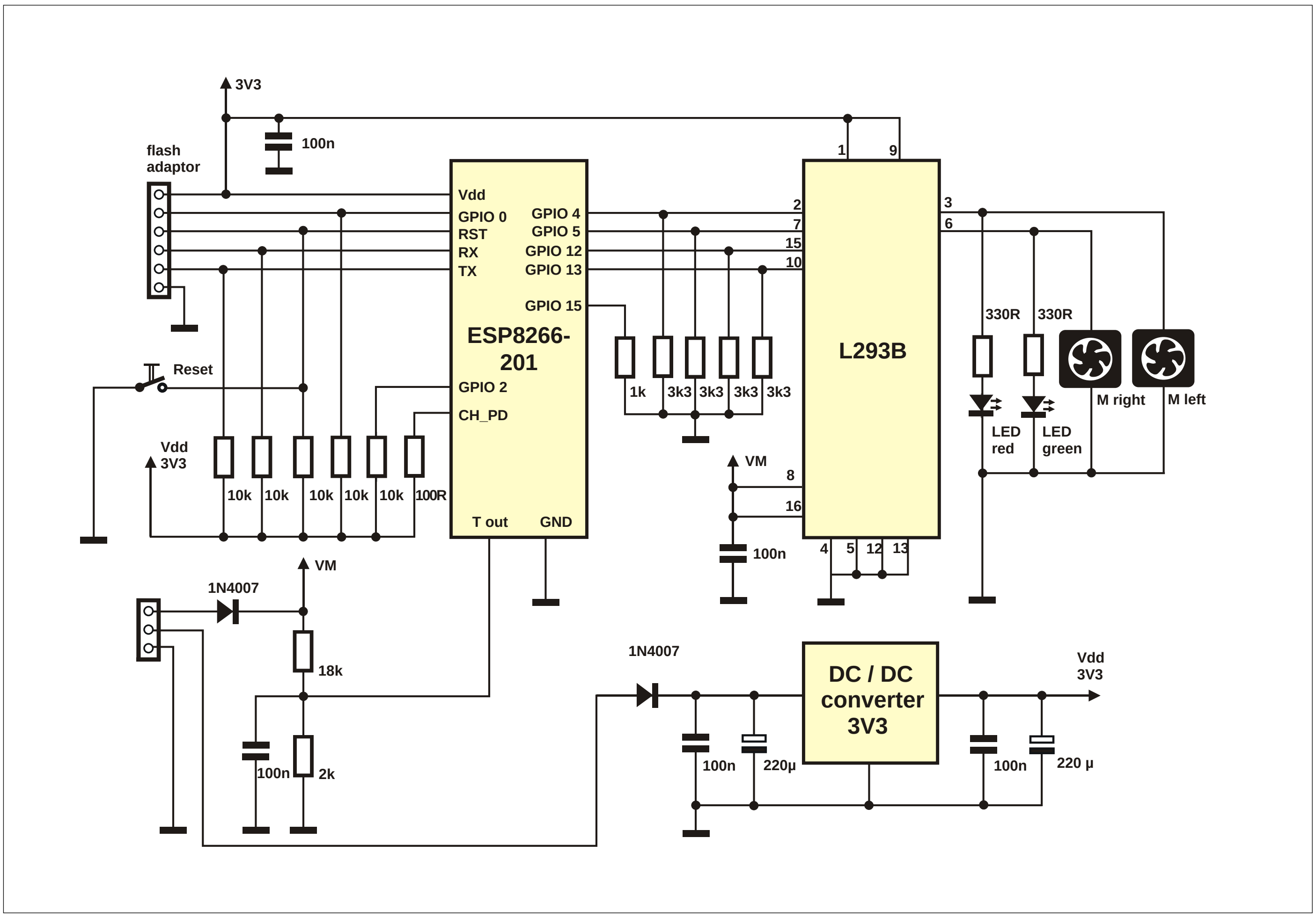


Figure 1. The fan catamaran electronics.

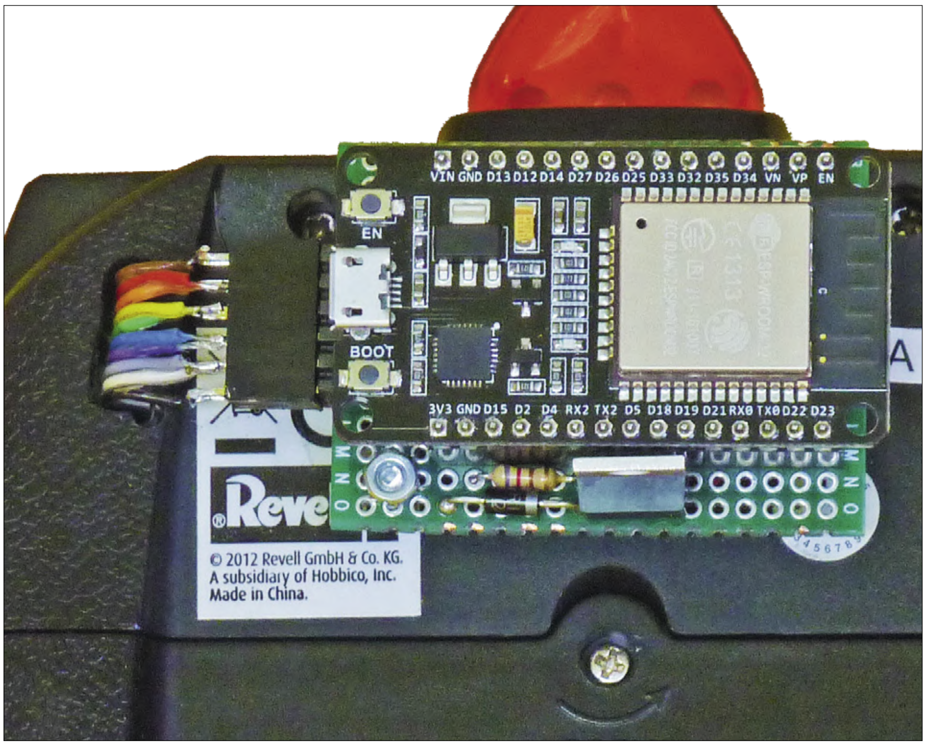


Figure 2. The ESP32 module is mounted externally on the back of the RC remote controller.

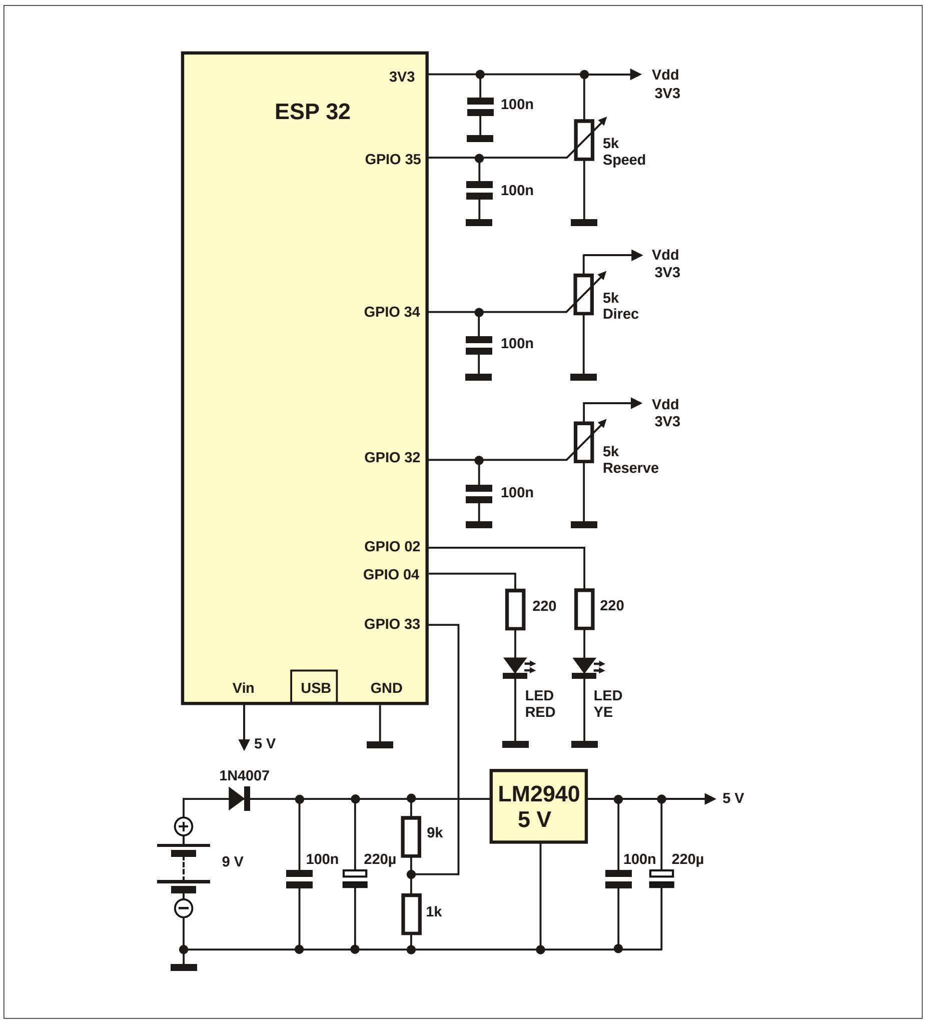


Figure 3. Two pots control the catamaran. The third one is reserved for use later.

The ESP8266, type ESP-201 consists of a small breakout board with many pin outs for connection to the processor I/Os. This board is mounted on a small carrier board with an external voltage regulator and driver chip.

Figure 1 shows the electronics mounted in the catamaran. In addition to controlling the catamaran fans, the ESP8266 creates a dedicated Wi-Fi radio network and also provides a cut-down MQTT broker service.

The fans draw up to 0.5 A at full load so a powerful L293B driver chip handles the power to the fans. Just two of the four driver stages are used in this application. This leaves two spare which could later be used to build a bridge circuit if brushed motors (reversible) were used instead of the PC fans in the future. LEDs are connected in parallel to the fans, so you can see when power is applied to them. The fan motors are supplied directly by the battery voltage VM which works out at 7 to 8.4 V, minus the voltage drops produced by the series protection diode and losses in the driver stage. GPIO control lines 4, 5, 12 and 13 are pulled down to ground with 3.3 kΩ resistors so that they are at a defined voltage level during the controller initialisation period.

The battery voltage level is continually monitored; it is connected via a 10:1 voltage divider network to the single analog input of the ESP8266. The complete voltage input range of the controller analogue input extends from 0 to 1 V which allows voltage levels up to 10 V to be connected at the divider network input. Pins CH_PD and GPIO 2 are connected to Vdd and GPIO 15 to GND. For programming, the RST, GPIO 0, TX and RX pins are led out to a six-pin flash connector and also pulled up to Vdd via resistors.

A linear regulator type LM2940 is used to supply the 3.3 V necessary for the controller board. Decoupling capacitors are connected to the supplies before and after the regulator to iron out any voltage spikes produced by motor switching. A diode is connected in series with the battery supply to prevent any damage caused by inadvertent battery reversal.

A repurposed RC hand controller

An old model helicopter hand-held controller is given a new lease of life here. The original circuit used an IR link to control the helicopter but we won't be using that method here. Much of the control-

ler hardware such as the battery compartment, three control potentiometers, the red and yellow LED and the on/off switch will make a handy and intuitive 'user interface' for catamaran control.
To communicate with the catamaran requires a Wi-Fi connection and an MQTT client. These functions can be perfectly well handled by an ESP32. And why is its smaller brother, the ESP8266 (used in the catamaran, where it has more work to do) not used in the controller? The answer lies in the number of analogue input channels available (needed to read the pot positions); the ESP8286 has only one, while the ESP32 provides several such inputs
Since most of the space in the controller is taken up by the battery compartment (6xAA cells), the ESP32 needs to be mounted externally. The small board made up of the ESP32 development module with USB programming port is wired to the components in the console housing using flat cable, as shown in **Figure 2**.
The circuit (**Figure 3**) is quite straightforward. The controller has two potentiometers; a forward speed controller and a left/right joystick. The wiper connections of the pots must be connected to the analogue inputs of the controller. To control the catamaran, only the speed and left/right direction controls are required, the unused forwards/backwards joystick control is wired to the controller but its use is reserved and may be used in future projects. The analogue inputs have a measuring range of 0 to 3.3 V, which corresponds with the voltage range provided by the potentiometer wipers. The battery voltage is supplied via a 10:1 divider to the analogue input GPIO 33 (ADC1_CH5).
The two digital outputs GPIO 2 and GPIO 4 control the LEDs; the yellow LED lights up as a power indicator and the red one indicates low battery alert in the catamaran or console.
An LM2950 linear regulator converts the 9 V battery supply to 5 V to power the ESP32 board which also has its own on-board regulator to convert this 5 V input down to 3.3 V for the ESP32 controller.

Preparation is the key

The Arduino IDE was used to develop the project software. This development environment is not only easy to use, but gives access to many libraries and application examples. Fortunately, Espressif's ESP8266 and ESP32 MCUs can be integrated into this IDE by adding the core modules. To develop the software for this project it was first necessary to install the following components on a PC:
• the Arduino IDE: download the latest version of [2] and install it.
• the ESP8266-Core: Under [3] it is documented how the add-on module can be inserted into the IDE.
• the ESP32-Core: Here too there is a very good installation manual at [4].
Once the IDE with the two cores are installed, additional libraries must be added. This (and how certain parameters are set) can be found in the following sections about the control programs.

In short: MQTT

Of course, communication between catamaran and console could have been done through simple TCP packets, but I was keen to find out if an MQTT broker service could indeed be squeezed into the small ESP8266. An advantage of this communication method is that you can fire up an MQTT client on a PC and watch the complete data exchange. If you are new to MQTT it might be worthwhile reading the information provided in the box below.
The broker forms the central hub of the MQTT message log. There are quite a few MQTT brokers, proprietary and open source solutions like Mosquitto. However, for MQTT to run on such a low-power MCU as the ESP8266, a special broker is required, which will have to forego some of the capabilities of MQTT, mainly because of the limited main memory.
The ESP8266 broker used here supports
- protocol versions MQTT v3.1 and v3.1.1 at the same time;
- up to eight connected clients;
- quality level 0;
- testament and last will;
- provision of retained messages also for newly joined clients (retained);
- authentication by use of name and password.

Altogether that's pretty neat, and for this application we can happily sacrifice:
- the transmission qualities 1 and 2 for guaranteed delivery;

MQTT at a glance

The MQTT (Message Queuing Telematic Transport) protocol was developed by IBM and is now freely available, in contrast to the more familiar HTTP protocol which uses request/response MQTT uses a publish/subscribe architecture in which a central broker (server) is used. Devices that have something to report send (publish) their messages to the broker, who sends these messages to other devices or clients who have subscribed to these messages or topics.
The communication can take place in both directions, so a client can publish and subscribe. A node can be a simple microcontroller, a PC or a Linux server. A prerequisite is that a TCP stack and the MQTT protocol are implemented on it.
Addressing when sending and receiving messages works via so-called topics. These are strings that represent a kind of subject of news, but are similar in structure to a URL. For example, a temperature sensor in workroom might publish its current temperature on a topic such as 'house/office/temperature'. In addition to the topic, the payload and other parameters are transmitted. The data can be transferred as binary values, texts and even XML or JSON structures.

To make the system flexible three levels of message security can applied:

Level 0: No special security according to the motto: fire and forget.
Level 1: Guaranteed transmission of at least one message, more copies can arrive.
Level 2: Guaranteed delivery of exactly one message without copies.

Ever heard of a testament protocol? That is also possible with MQTT. In the MQTT world a client can formulate a 'last will' or testament, in which it declares what message should be sent on it's behalf by the broker, after it has gone offline This could be, for example, a notification to a responsible administrator.

There are many more features of MQTT such as filtering, multi-level security concept, failure protection and and and...

Listing 1. Initialisation

```
#include
#include
#include "Ticker.h"
#include

#define speed_TOPIC    "speed"
#define direc_TOPIC    "direc"
#define katama_TOPIC   "catama"

const char* ssid = "ESP_HOST";
const char* password = "";
const char* mqtt_server = "192.168.4.1";
```

Listing 2. The MQTT-relevant instructions.

```
WiFiClient espClient; // Definitions: WiFi- and MQTT-Client
PubSubClient client(espClient);


client.setServer(mqtt_server, 1883);  // Setup: Broker Address and
Port
client.setCallback(receivedCallback); // Setup: Start Callback for
Subscription

if (!client.connected()) {            // Setup: Connection to Broker
    mqttconnect();  }


client.loop(); // Loop: Calling the MQTT Cient and
send_MQTT();   // execution of required activities

client.publish(speed_TOPIC, spdmqt); // Loop: Publishing of actuator
data
delay(100);
client.publish(direc_TOPIC, dirmqt);
delay(100);


// Callback Function
void receivedCallback(char* topic, byte* payload, unsigned int
length)
```

These are the MQTT commands used:

Topic	Payload	Values
speed	speed_xxx	speed_000 = min, speed_100 = max
direc	direc_xxx	direc_000 = links, direc_100 = geradeaus, direc_200 = rechts
vbatt	vbatt_xxx	vbatt_999 = 9,99 V

- secure encryption TLS (Transport Layer Security);
- permanent storage of publications (non-clear sessions).

The developer of this free software [5], Martin Ger, has chosen the ESP8266 as a platform, because this MCU is used in the low-priced Sonoff switches and thus he could save himself the requirement to use an additional PC or Raspberry Pi to supply the broker function. Its development is very efficient, according to my measurements; the broker creates about 100 publications/subscriptions per second. Hats off to you Martin!

The console publishes the data of the joysticks under the topics ‘speed’ and ‘direc’ whenever a control lever position is moved, or at least every ten seconds. Although the topic name is also included in the payload, it is only used for better identification of the communication. The catamaran also cyclically sends its battery voltage measurement, which is evaluated in the console and displayed when the voltage drops too low.

The controller unit software

The software for the control console is also implemented as an Arduino sketch. It transmits via MQTT the current joystick positions to control speed and direction of the catamaran. It continually checks the battery voltage levels in the control console and the catamaran and reports an under-voltage by flashing the red LED. The yellow LED flashes every second as a status indicator.
The software called **Konsole.ino** and the necessary libraries are available in the Elektor project folder [6]. In the Arduino IDE under ’Tools/Board’, select your ESP32 module (for example, ’ESP32 Dev Module’) and set the port number and transmission speed to 115200 Bit/s. The MQTT client software is located in the file ‘pubsubclient-master.zip’ and should be installed using ’Add Sketch/Library/ Insert.ZIP Library’. In addition, the two files ’Ticker.h’ and ’Ticker.cpp’ should be in your project folder.
The program structure is shown in **Figure 4** and corresponds to the typical Arduino pattern with the setup part and then the loop routine. In order not to block the loop routine waiting for an incoming MQTT message, the subscription of the catamaran battery voltage is outsourced to the callback func-

tion `receivedCallback()`. If a message arrives, this function becomes active and evaluates the message. Two additional routines are started at ten or one second intervals by using timers in the ticker library. `firemqtt()` triggers a publication of the joystick data at least every ten seconds, while `blinker()` switches the LEDs every second.
As usual, the initialisation takes place in the setup function, which includes, among other things, the connection to the Wi-Fi, the start of the MQTT client with connection to the broker and the launch of the callback functions.
The analogue values of the controller positions are first determined in the endless loop and then immediately published to the catamaran to ensure minimal control delay. If the joysticks have not been moved, their values will be transmitted at the ticker interval. In this cycle, measurement and evaluation of the battery voltages takes place. When necessary,the LED battery-low warning indicator flashes alternately with the yellow status indicator).
At program initialisation (**Listing 1**) the libraries required for the Wi-Fi, MQTT and ticker for the timer interrupt are included. After that the MQTT-Topics and the credentials for the local catamaran Wi-Fi and the broker are listed. Wi-Fi access protection is not required here, there will be no link to the internet, here *ESP_HOST* is the network SSID but no password is used. For more critical applications it would be more important to assign a password, ensure good encryption and secure access to the broker with name/password. The MQTT server IP address is predefined as *192.168.4.1*.
Following are the MQTT-relevant instructions (**Listing 2**, here the position in the

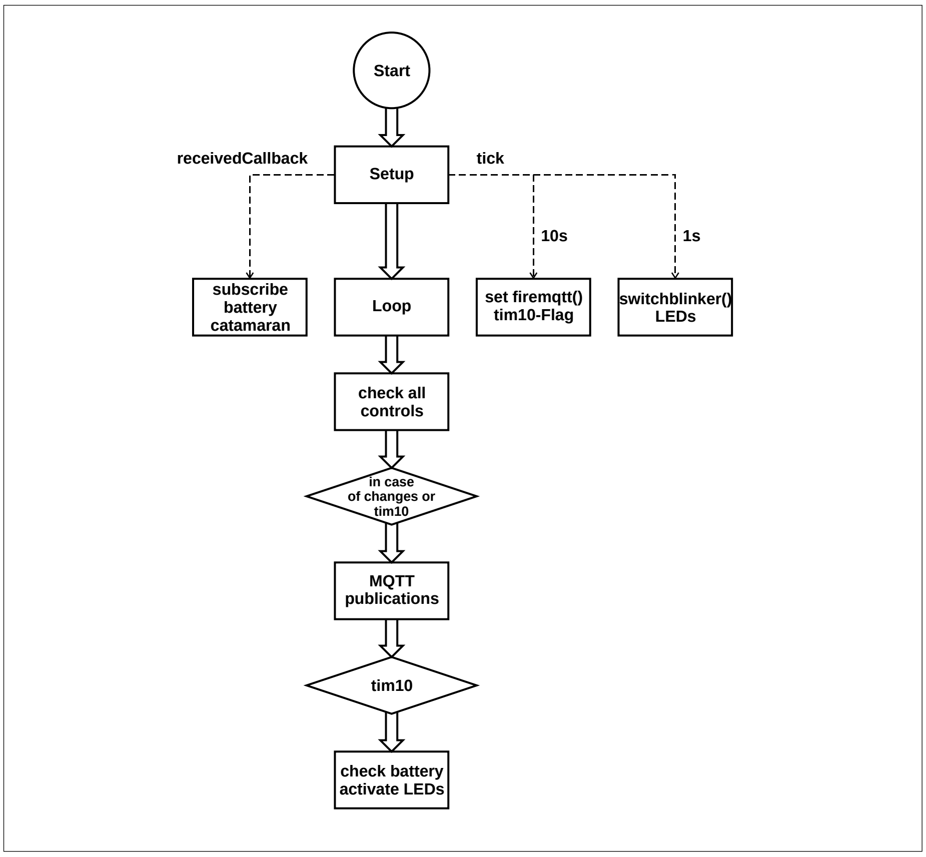

Start
Setup
receivedCallback
tick
10s
1s
subscribe battery catamaran
Loop
set firemqtt() tim10-Flag
switchblinker() LEDs
check all controls
in case of changes or tim10
MQTT publications
tim10
check battery activate LEDs

Figure 4. The console program in the typical Arduino environment.

Listing 3. The two ticker timers.

```
Ticker minmqt; // Definition: Ticker allocation
Ticker ledblink;

ledblink.attach_ms(1000, blinker); // Setup: Time interval and
service function
minmqt.attach(10, firemqtt);

void firemqtt(){      // Function: firemqtt()
  tim10 = true;
}
```

Web Links

[1] ESP32/ESP8266 compilation: www.elektor.com/esp32-esp8266-compilation-en
[2] Arduino IDE: www.arduino.cc/en/Main/Software
[3] Start ESP8266 (in German): www.heise.de/ct/artikel/Arduino-IDE-installieren-und-fit-machen-fuer-ESP8266-und-ESP32-4130814.html
[4] Start ESP32: www.elektormagazine.com/labs/esp32-getting-started
[5] Martin Ger: https://github.com/martin-ger
[6] The project at Elektor Labs: www.elektormagazine.com/170198-01
[7] Broker Arduino: https://github.com/martin-ger/uMQTTBroker
[8] Broker C: https://github.com/martin-ger/esp_mqtt/
[9] Broker explained: www.youtube.com/watch?v=0K9q4IuB_oA
[10] MQTT-Praxisbuch (in German): https://www.elektor.de/das-mqtt-praxisbuch

Listing 4. Details of the Catamaran software.

```
#include
#include "uMQTTBroker.h"           // MQTT-Broker with client
#include "Ticker.h"
#include

char ssid[] = "ESP_HOST";          // SSID (Name) of catamaran-WLAN
char pass[] = "";                  // WLAN Password, here: open
bool WiFiAP = true;                // should a dedicated WLAN be set
up?

#define speed_TOPIC    "speed"     // Definition of Topic
#define direc_TOPIC    "direc"
#define katama_TOPIC   "katama"

myMQTTBroker myBroker;            // Definition of Broker instance
```

Listing 5. MQTT broker callback functions.

```
if (WiFiAP)                  // Own WLAN …
  startWiFiAP();
else
  startWiFiClient();         // without participation in home net

myBroker.init();             // Setup: Broker inititialisation

myBroker.subscribe(speed_Topic);
                             // Subscriptions of Speed and Direction
myBroker.subscribe(direc_Topic);

myBroker.publish(katama_TOPIC, batmqt);
                             // Loop: battery voltage publication
```

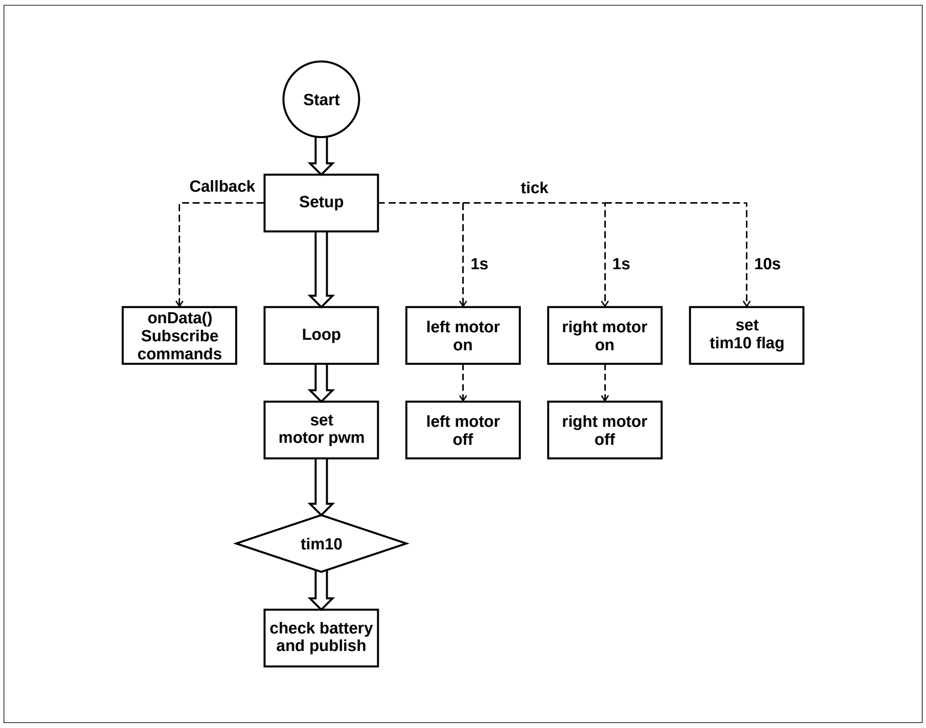


Figure 5.The catamaran control flow chart.

source code is also indicated). The publications take place in the loop at intervals of 100 ms, which gives the controller in the catamaran enough time to evaluate the message without being overloaded. It receives a maximum of ten messages per second, which the broker can easily handle. The callback function returns all necessary data of the received subscription such as topic, payload and payload length.

The two tickers are defined at the beginning of the program (**Listing 3**). In the setup they are assigned a time interval in milliseconds and seconds and a Service-Function. These interrupt functions should be kept as short as possible so that they do not introduce unnecessary delays. In the `firemqtt()` function, only a flag is set and taken into account in the loop.

More detailed information about the program is provided by comments in the source code. Note the inclusion of some `Serial.print` output commands that allow you to follow the program flow via the Arduino monitor.

The catamaran software

It will also be necessary to make some setup in the Arduino IDE used to generate the catamaran software (**Figure 5**). Select the appropriate ESP8286 module under *Tools/Board* (here a *Generic ESP8266* modules is chosen) and then the port number and transmission speed (115200 Baud) are set. It is best to set the Reset-Method as *nodemcu*, so that you will not need to worry about pressing pushbuttons during the program flash procedure. It is also important to set the *LwIP Variant* parameter to *v1.4 Higher Bandwidth* in order to allocate maximum resources to the TCP stack.

The Broker-Software for the Arduino IDE is available at [7], alternatively it is also a version in C available at [8]. It is recommended to first watch the developer's Youtube video [9] which gives a very good functional description. The software is included in the Elektor folder [6] for this project, it just needs to be stored in your project folder, no installation in the Arduino IDE is required. The MQTT server IP address is set to *192.168.4.1* by default.

The program which controls the catamaran has the file name **Katamaran.ino** and can be found in the project folder. Its structure is similar to the console software. Here, too, callback and ticker

timer routines are started in the setup, the `onData()` function is always activated when a console message arrives and the ticker timer functions start according to the associated time interval parameter.
The fans used only react very slowly at the usual PWM frequencies, so in this project the PWM frequency is very low frequency at 1 Hz. The fans are switched on every second and then off after a time dependent on the set speed, the flywheel effect of the rotating fan blades smooth out the fan speed changes.
Another ticker timer triggers the measurement and publication of the battery voltage every ten seconds. The switch-on intervals are calculated in the program loop and passed to the ticker timer functions.
There are some interesting passages of the software In **Listing 4**. In the Definitions, the *WiFiClient* the MQTT broker and the ticker timer library functions are called. The catamaran's local Wi-Fi network with the SSID '*ESP_HOST*' is specified without a password, which means access is not restricted by any security measures. Setting WiFiAP to *true* sets up a dedicated Wi-Fi Access Point.
The MQTT broker provides three important callback functions (**Listing 5**):

- `onConnect`: Activates when connecting a new client and transmits its IP address and the number of connected clients. This can be used to decide if this client is welcome and operational.
- `onAuth`: When connecting to a new client, will provide name and password, if installed. A return of `true` will result in an approval. In this project, all clients, with or without user data, are admitted.
- `onData`: Receives all subscribed top-

Listing 6. The WiFi and MQTT instructions

```
class myMQTTBroker: public uMQTTBroker
{
public:                  // A client logs on
  virtual bool onConnect(IPAddress addr, uint16_t client_count) {
    Serial.println(addr.toString()+" connected");
    return true;         // true = allowed, false = rejected
  }
                         // Authentication of Client
  virtual bool onAuth(String username, String password) {
    Serial.println("Username/Password: "+username+"/"+password);
    return true;         // true = allowed, false = rejected
  }
                         // Reception of subscription
  virtual void onData(String topic, const char *data, uint32_t length) {
    char payload[length+1];
    char payval[4] = "000";

    os_memcpy(payload, data, length);
    payload[length] = '\0';
    if(topic == speed_TOPIC){         // Speed command arrived
      payval[0] = payload[6];
      payval[1] = payload[7];
      payval[2] = payload[8];
      spdval    = atoi(payval);
      if((spdval >= 0) && (spdval <= 100))   // Allowed range: 0..100
        spdok = true;
    }
    if(topic == direc_TOPIC){         // Direction comamnd arrived
      payval[0] = payload[6];
      payval[1] = payload[7];
      payval[2] = payload[8];
      dirval    = atoi(payval);
      if((dirval >= 0) && (dirval <= 200))   // Allowed range: 0..200
        dirok = true;
    }
    Serial.println("received topic '"+topic+"' with data '"+(String)payload+"'");
  }
};
```

ics and transmits the payload with its data length in addition to the topic name. Then, the payload's numerical portion is transferred and its conversion into the integer variables `spdval` and `dirval`, respectively. Then it is checked whether this value is within the valid range and transferred to the motor control by means of a flag.

Listing6 lists the relevant WiFi and MQTT instructions.

The actual engine control in **Listing 7** is based on four ticker timers. The start functions `Tlion` and `Treon` are assigned fixed start intervals of 1000 ms. The service functions `Molion` and `MoReon` turn on the fans and start the `Molioff` and `Moreoff` routines with the calculated pulse lengths, which eventually shut down the motors. A fifth ticker timer switches the `tim10` flag every ten seconds to trigger a battery measurement. The Arduino IDE carried out all the necessary tasks with flying colours and during development I didn't encounter any problems or restrictions. It shows you don't always need to resort to more sophisticated development environments to build small or medium-sized projects.

Were the kids impressed?

With some glee the grandchildren took their creation down to the pond. It performed really well and has a range of about 50 m although it sits a bit high in the water and is quite light so tends to be at the mercy of any passing gust of wind. Engineers are always looking for improvements; it may be worth converting to the more conventional submerged propeller propulsion to help reduce superstructure height and influences of the wind. The project has demonstrated to me that an MQTT broker running on an ESP8266 is a practical and useful control solution for smaller projects and given me some ideas for future projects. More importantly, working with the grandchildren is a great learning opportunity; I hope they will get as much fun playing with it as I did by helping build it. ◀

170198-02

Listing 7. The motor control.

```
             // Setup: Definition of Tickers
Ticker Tlion;                // Left Motor on
Ticker Tlioff;               // Left Motor off
Ticker Treon;                // Right Motor on
Ticker Treoff;               // Right Motor off
Ticker Tick10;               // Ticker 10 sec.

Tlion.attach_ms(1000, Molion);       // Start left Motor each second
Treon.attach_ms(1000, Moreon);       // Start right Motor Motor each
second
Tick10.attach(10, SendBatt);         // Ticker 10 sec.

void Molioff(){              // Switch off Motor
  digitalWrite(MOLI,LOW);
}
void Molion(){               // Switch on Motor ein
  digitalWrite(MOLI,HIGH);
  Tlioff.attach_ms(molipow, Molioff);
                 // Switch off Motor with variable time 0..1000
}

void SendBatt(){             // Set Ticker Flag
  tim10 = true;
}
```

MQTT spans the seven seas

Installation information

Software folder contents

`Konsole.ino`: ESP32 console control program
`Katamaran.ino`: ESP8266 catamaran control program including MQTT broker.
`uMQTTBroker-master.zip`: MQTT-Broker library functions for Katamaran.ino
`pubsubclient-master.zip`: MQTT-Client library functions for Konsole.ino
`Ticker.h / Ticker.cpp`: Library for the Ticker timer functions
`mqtt-brok-E.zip`: C-Version of the MQTT Broker for use with the Espressif IDE (not required for the Arduino IDE)

Software installation

Arduino-IDE: Install ESP8266 and ESP32 Core.
Catamaran: Install uMQTTBroker-master.zip in the Arduino-IDE (as a zip file).
Console: Install pubsubclient-master.zip in the Arduino-IDE (as a zip file).

A Simple A/D Converter Using a PLD

Implementing a sigma-delta ADC with minimum complexity

By **Guido Nopper** (Germany)

You can really do a lot of things with PLDs. For example, with just 26 macrocells you can implement an 8-bit A/D converter based on a sigma-delta modulator (ΣΔM) followed by a digital low-pass/conversion filter. The accuracy, resolution and complexity of this project are described in this article. An interesting aspect is that the digital filter accounts for most of the complexity.

There are various ways to convert analogue signals into digital data. The formal name of this sort of circuit is ‘analogue to digital converter’, but informally it is also known by the name ‘A/D converter’ or the abbreviation ‘ADC’. The ADC described in this article uses the oversampling method.

Oversampling

With oversampling, the analogue input signal is sampled at a significantly higher frequency than what is required by Nyquist–Shannon sampling theory, which says that the sampling frequency must be at least twice the bandwidth of the desired signal. **Figure 1** shows the operating principle of this type of ADC. Here the modulator generates a 1-bit data stream with a bit rate f_S. With just 1 bit, the amplitude resolution is extremely low, but the resolution in the frequency domain is significantly enhanced. The oversampling ratio, designated as R or OSR (for ‘OverSampling Ratio’), often exceeds the bandwidth of the signal to be converted by a factor that is a power of 2, such as 32, 64, 128, etc.

The task of the downstream conversion filter is to increase the amplitude resolution to a specific number of bits by averaging the data stream. This corresponds to a low-pass filter. The conversion filter allows the sampling frequency to be reduced to a lower output sampling frequency f_{SO} without folding too many undesirable spectral components into the frequency range of the desired signal. **Figure 2** shows the spectra of the signals in Figure 1. A welcome side effect of oversampling at a high frequency f_S is that the necessary anti-aliasing filter can be implemented very easily, for example as a low-order filter, or even eliminated. However, in that case reducing the sampling rate at the output to f_{SO} could allow

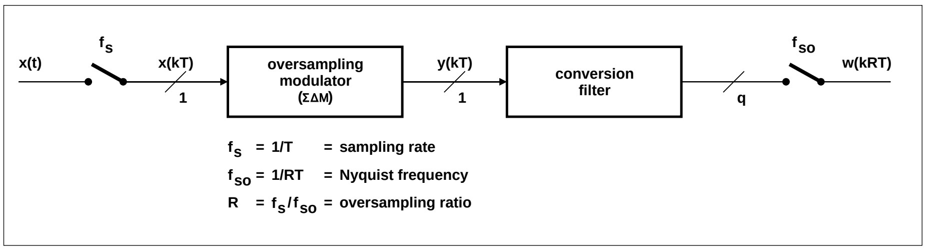

x(t)
f_S
x(kT)
1
oversampling modulator (ΣΔM)
y(kT)
1
conversion filter
q
f_{SO}
w(kRT)
f_S = 1/T = sampling rate
f_{SO} = 1/RT = Nyquist frequency
R = f_S/f_{SO} = oversampling ratio

Figure 1. Operating principle of an ADC with oversampling.

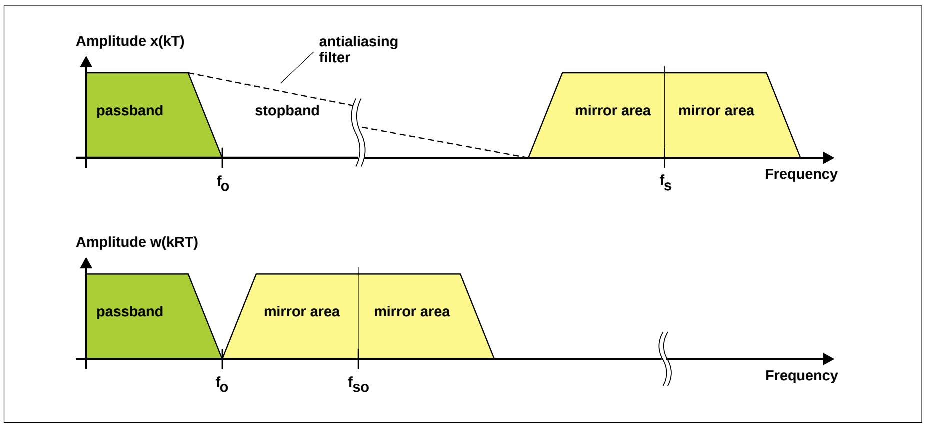


Figure 2. Spectra of the signals in Figure 1.

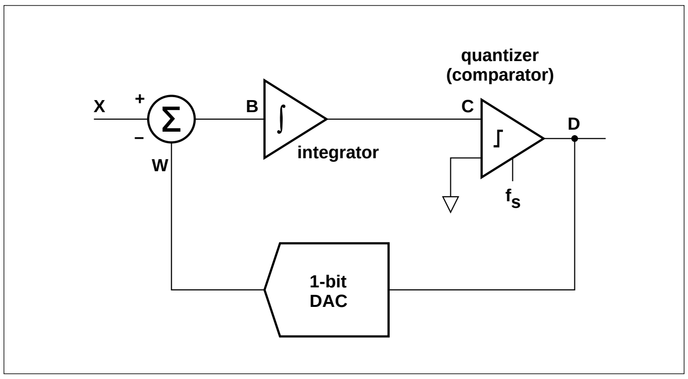


Figure 3. Block diagram of a simple sigma-delta modulator.

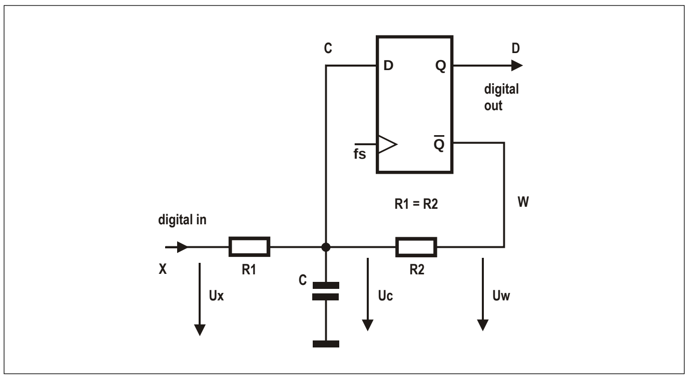


Figure 4. The simplest sigma-delta modulator.

undesirable spectral components in the filter stopband to be folded into the frequency range of the desired signal.

The sigma-delta modulator

A first-order sigma-delta modulator (ΣΔM) consists of a subtractor, an integrator, a quantizer (comparator) and a 1-bit DAC. **Figure 3** shows the corresponding block diagram. This method is sometimes also called 'delta-sigma modulation' because the difference between the input signal (point X) and the output signal from the 1-bit DAC (point W) is first obtained by the subtractor (Δ stage, point B) and then the difference is summed (Σ) by the integrator. The inputs to the quantizer (comparator) are the integrated difference signal (point C) and the clock signal at the sampling frequency f_S. This means that the output signal (point C) can only change at the sampling time points and is therefore quantised at the frequency f_S. Modulators of this sort have been known for many years [1].
A simple sigma-delta modulator can be built very easily from a D-type flip-flop, two resistors and a capacitor (see **Figure 4**). If the sampling frequency is significantly higher than the bandwidth of the signal to be converted, the A/D converter is called an oversampling converter. The signal designations in italics here are intended to show which points in Figure 4 roughly correspond to those in Figure 3. As you can easily see, the simplicity of the circuit essentially comes at the expense of lower accuracy.
In the sigma-delta modulator shown in Figure 4, the inverted digital output signal (point *W*) is converted from digital to analog by passing it through a low-pass RC filter and then added to the input signal. Due to the inversion, this corresponds to subtraction or comparison. The signal level at point *W* forms the reference voltage for the A/D conversion, which means it must fulfil analog requirements. The high and low output levels of a CMOS flip-flop are close to V_{CC} and GND.
The D input of the flip-flop acts as an analog comparator, so the accuracy of the system transfer function depends on its trigger level. The capacitor voltage U_C (point *C*) always remains close to the trigger level. When the input voltage at point *X* rises and the voltage U_C rises accordingly, the flip-flop outputs a higher proportion of low bits at point *W*, causing U_C to drop back down. In effect, the average value of the sigma-delta modulator output signal follows the signal level at the analog input.

Input voltage range and transfer characteristics

To prevent limiting in the sigma-delta modulator shown in Figure 4, the input voltage must remain within a suitable range. After all, the flip-flop can only persistently output a high level or a low level. At the maximum input voltage U_X, the Q output of the flip-flop is constantly high and the inverted output is constantly low.
If we ignore the input current of the D input and the leakage current of the ceramic capacitor C, the average currents through the two resistors R1 and R2 must be the same. This means:

$$I_{R1} = I_{R2}$$

$$\frac{U_X - U_C}{R1} = \frac{U_C - U_W}{R2} \quad \text{with R1 = R2}$$

$$U_X - U_C = U_C - U_W$$

$$U_X = 2U_C - U_W$$

This means that at the maximum input voltage, point *W* must be constantly low:

$$U_{Xmax} = 2U_C - U_{Wlow}$$

At the minimum input voltage, point *W* must likewise be constantly high:

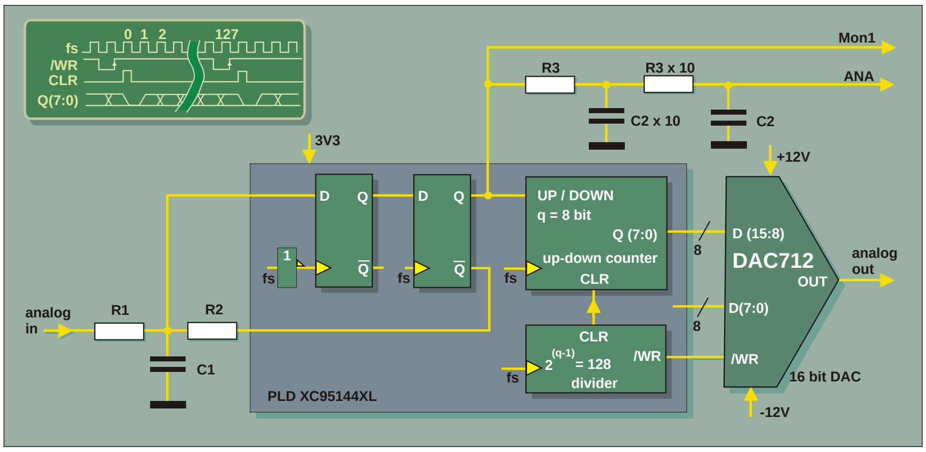


Figure 5. Block diagram of the sigma-delta ADC test arrangement.

$U_{Xmin} = 2U_C - U_{Whigh}$

If we assume that
U_{Wlow} = 0-5% V_{CC},
U_{Whigh} = 95-100% V_{CC} and the trigger level U_C = 45-55% V_{CC}, we have:

U_{Xmax} = 85-110% V_{CC}

and

U_{Xmin} = -10-14% V_{CC}

The input voltage range ΔU_X is equal to the difference between U_{Xmax} and U_{Xmin}:

ΔU_X = 90-100% V_{CC}

This means it is certainly possible for the input voltage to reach the limits and the range of V_{CC}.

For determination of the dynamic input impedance, we can assume that U_C is held nearly constant, so this point will have a very low impedance. The input impedance for AC signals R_{INac} is therefore given by:

R_{INac} = R1

For the DC analysis, it is necessary to consider that the far end of R1 is tied to U_C. This means that current will flow into the input *X* when U_{IN} is higher than U_C and out of the input *X* when U_{IN} is lower than U_C.
A difficulty is that the D input of the flip-flop is always held at the trigger level, which means it is outside the specified operating range. There is no compliance with nominal high and low levels at the input, nor with nominal setup and hold times. As a result, the flip-flop may exhibit metastable behaviour and its outputs may not always be precisely opposite to each other. To compensate for this, the output signal can be resampled with a second D-type flip-flop as shown in **Figure 5**. To avoid having too much signal delay in the control loop, the opposite edge of the clock signal is used here for resampling.
It may also be necessary to deactivate any PLD pull-up or pull-down resistors on the D-type flip-flop inputs to avoid additional inaccuracies. An existing 'keeper' function in the form of a positive feedback resistor in the kilo-ohm range over the internal non-inverting input amplifier is acceptable because the relatively large input capacitance C prevents any significant change in the signal voltage during a sampling interval.
With a full-scale signal, the sigma-delta modulator shown in Figure 4 achieves a signal to noise ratio (SNR) of:

$$S/N = \frac{3}{4\pi}\left(\frac{f_s}{f_o}\right)^{\frac{3}{2}}$$

Here f_S is the sampling frequency and f_O is the upper limit frequency of the desired signal bandwidth (0 to f_O). As you can see, doubling the sampling frequency f_S increases the SNR by a factor of $2^{3/2}$, which is 9 dB. How is this possible?
With a conventional Nyquist ADC the SNR rises by a factor of 2 as a result of oversampling, but here it rises by an additional factor of $2^{1/2}$ (3 dB).

The reason is that with a conventional ADC the noise is evenly distributed over the frequency range. With a sigma-delta modulator, by contrast, the noise is shifted to the higher frequencies because the resistors and capacitors of the modulator act as a low-pass filter for the input signal and as a high-pass filter for the noise.

In **Figure 6** you can see that the noise starts rising at f_O and only levels off at $f_S/2$. The value of f_O is determined by the combination of R1, R2 and C (Figure 4), with R1 and R2 more or less in parallel:

$$f_o = \frac{1}{2\pi\left(\frac{R1 \bullet R2}{R1 + R2}\right)C}$$

With R1 = R2 = *R*, this gives:

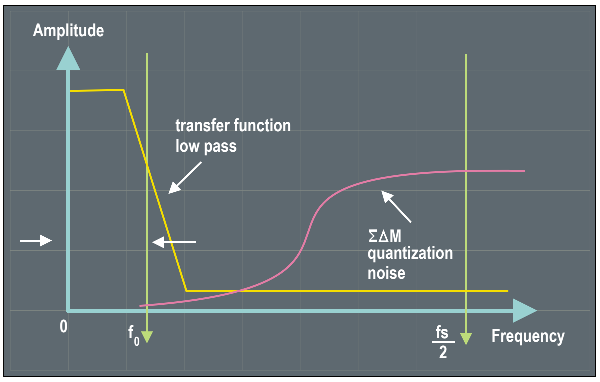


Figure 6. Sigma-delta modulator quantisation noise and frequency response of the digital filter.

$$f_o = \frac{1}{\pi RC}$$

Using the SNR formula with f_S = 12 MHz and a signal bandwidth f_O of 14.4 kHz gives a calculated SNR of 5743, equivalent to 75.2 dB. It is therefore reasonable to use a conversion filter with a resolution of 8 bits (or more) to generate the output signal from the sigma-delta modulator.

The digital low-pass/conversion filter

The main task of this filter is to attenuate the quantisation noise before the sampling frequency is reduced and this noise is folded into the frequency range of the desired signal. As the quantisation noise curve of a first-order sigma-delta modulator rises at approximately 20 dB/decade above f_O, a first-order digital low-pass filter is adequate for simple applications. This filter increases the word width from 1 bit to a larger number of bits *q*, which is why it is also called a conversion filter. Various types of filter are suitable for this purpose. A resettable 8-bit up/down counter, such as in Figure 5, can be used to implement this filter with minimum complexity. Its transfer function corresponds to a first-order sinc filter with M = $2^{(q-1)}$ stages. However, it uses $2^{(q-1)} - 1$ output bits from the sigma-delta modulator instead of $2^{(q-1)}$ as with an equivalent sinc filter. Resetting the counter after $2^{(q-1)}$ clock pulses yields a second timebase (f_{SO}), so the *z*-transform is no longer directly applicable [2]. Resetting the counter for one clock interval has the advantage that the counter output can assume maximum and minimum values of $\pm(2^{(q-1)} - 1)$. This means that the word width of the counter is fully utilised and twos-complement overflow is reliably prevented.
A complete ADC with this sort of filter and up/down counter for 8-bit output resolution (*q* = 8) can be implemented using just 26 macrocells of the selected PLD. Each additional channel requires only eleven additional macrocells if they have a shared clock divider. If a first-order sinc filter with a differentiator and integrator were used, all 144 available macrocells [3] would normally be necessary to build the ADC and filter with an M value of just 112, where M corresponds to the number of output bits of the sigma-delta modulator that are continuously averaged.

However, this arrangement can be built with just 50 macrocells by swapping the order of the integrator and differentiator. If the digital filter is implemented as a lossy accumulator with k = 0.99609375 (m = 8) and an output word width of 8 bits, a total of 61 macrocells will be needed for this ADC. Here the factor k is the multiplication coefficient used to multiply the previous result of an accumulation before it is added to the new input value. If the factor k is implemented by shifting and adding instead of using a full multiplier, m corresponds to the bit shift. Accordingly, a division by 2m can

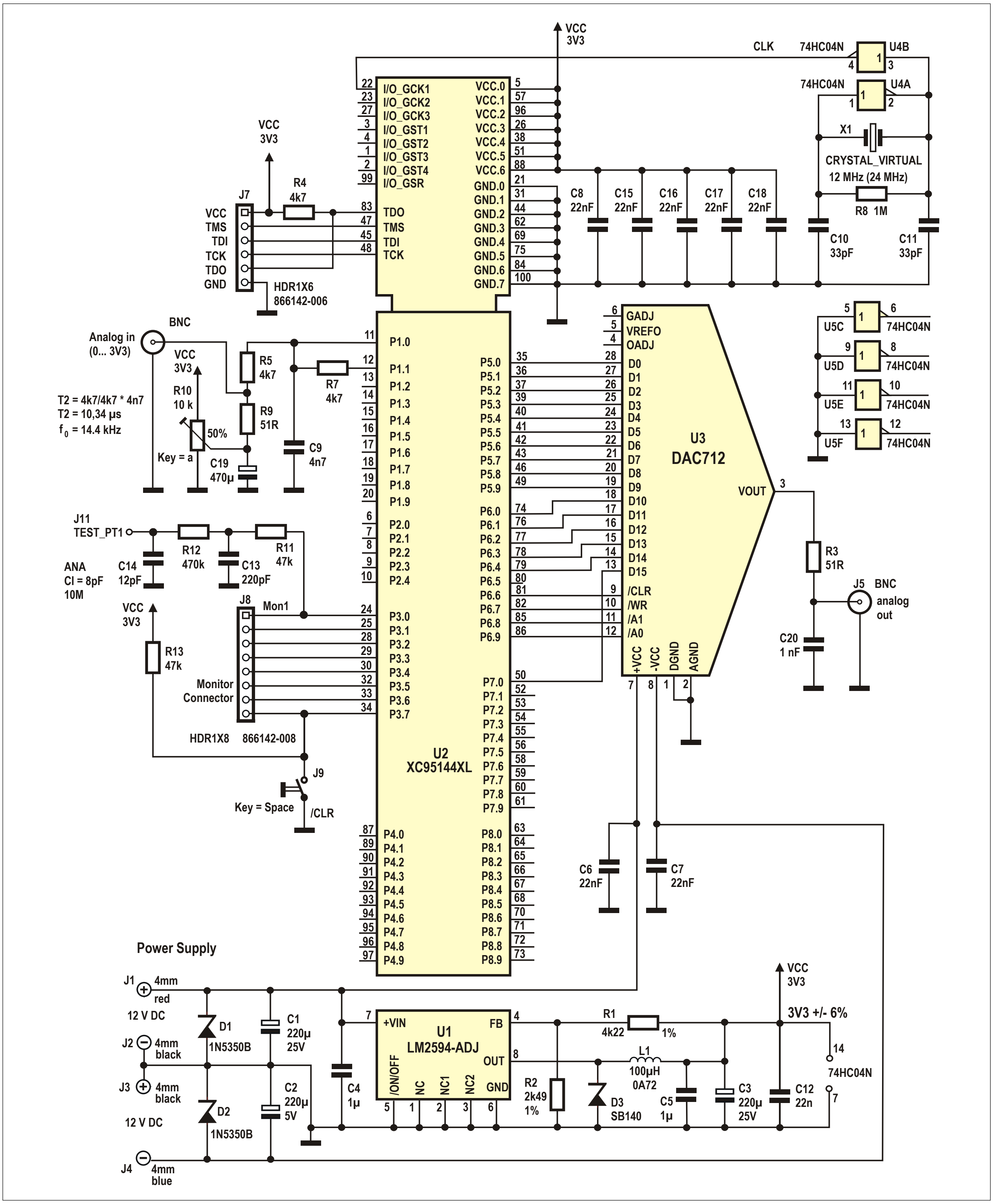


Figure 7. Circuit of the sigma-delta ADC test arrangement.

be achieved by connecting the binary data word shifted to the right by *m* bits.

Practical implementation

Figure 5 shows the block diagram of the sigma-delta ADC test arrangement. Here the sigma-delta modulator consists of two D-type flip-flops, which are triggered by opposite edges of the sampling clock (at a frequency f_S of 12 MHz) to avoid metastable states. The combination of R1, R2 and C1 for this sigma-delta modulator is dimensioned for f_O = 14.4 kHz. The output of the sigma-delta modulator is connected to Mon1 for checking. The analogue input signal contained in the output pulse sequence of the sigma-delta modulator appears at the ANA output after low-pass filtering (third order with f_{lim} = 15.4 kHz). The supply voltage of the PLD is 3.3 V, so the maximum allowable level of the analog input signal is also 3.3 V.

The output of the sigma-delta modulator is fed to the up/down input of the corresponding 8-bit counter. There a high level is interpreted as +1 and a low level as −1. The sigma-delta ADC output Q(7:0) supplies data in twos-complement format.

Every 128 clock cycles, the counter acting as a conversion filter is first read out and then reset for one clock interval. As a result, the maximum and minimum values possible at the counter output correspond to ±127 decimal, which nicely utilises the 8-bit value range and prevents twos-complement overflow. The maximum or minimum output value occurs when the analog input signal is at U_{Xmax} or U_{Xmin}, respectively. if the divider is always chosen to be one bit smaller than the up/down counter, the result is a symmetrical output signal that can never overflow. In the event of a twos-complement overflow, the signal would suddenly jump from the maximum value to the minimum value or the other way round, which would create very unpleasant harmonics. This is illustrated further on with a screenshot.

The sigma-delta ADC output Q(7:0) is connected to the high-byte input of a 16-bit twos-complement DAC. The low-byte input is not used here. However, it is wired in preparation for larger sigma-delta ADC word widths in the PLD. The balanced ±12 V supply voltages of the DAC allow an analogue output range of nearly ±10 V. The maximum DAC conversion rate of 100 kHz is slightly higher than the required rate of 93.75 kHz with f_{SO} = 12 MHz (12 MHz ÷ 128).

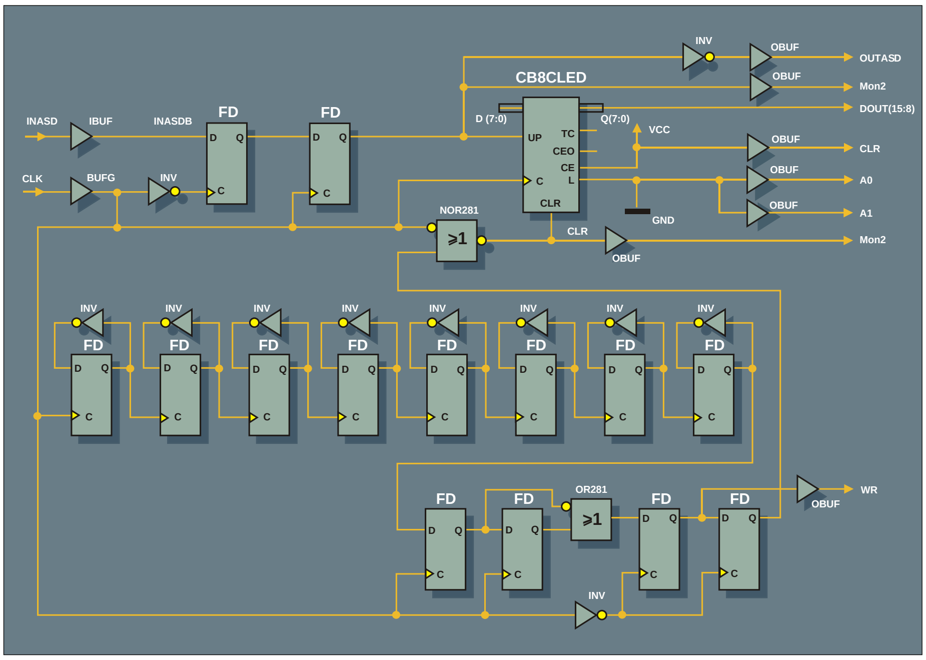


Figure 8. Internal PLD circuit of the sigma-delta ADC with 26 macrocells.

If this counter were laid out as a pure 8-bit up counter with the sigma-delta modulator output used as its Enable input, the counter output would be in offset binary format instead of twos-complement format. However, in that case the associated divider would have to be set to 2^q.

Figure 7 shows the circuit of the test arrangement. It also shows functions that are not necessarily part of the sigma-delta ADC, such as the voltage supply, clock generation, and 16-bit D/A conversion. **Figure 8** shows the internal circuit of the XC95144 PLD for the sigma-delta ADC. Only 26 of the total of 144 macrocells are needed. Together with the three external components R1, R2 and C1, this results in a complete sigma-delta ADC with 8-bit resolution. The output data is only valid on the rising edge of the /WR signal. If this is not suitable for your purposes, you must add an 8-bit D-type register triggered on this signal edge.

Time domain measurements

Several measurements were made on the test arrangement to show how this PLD ADC performs and determine its characteristics. **Figure 9** is a photo of the test setup for the time domain

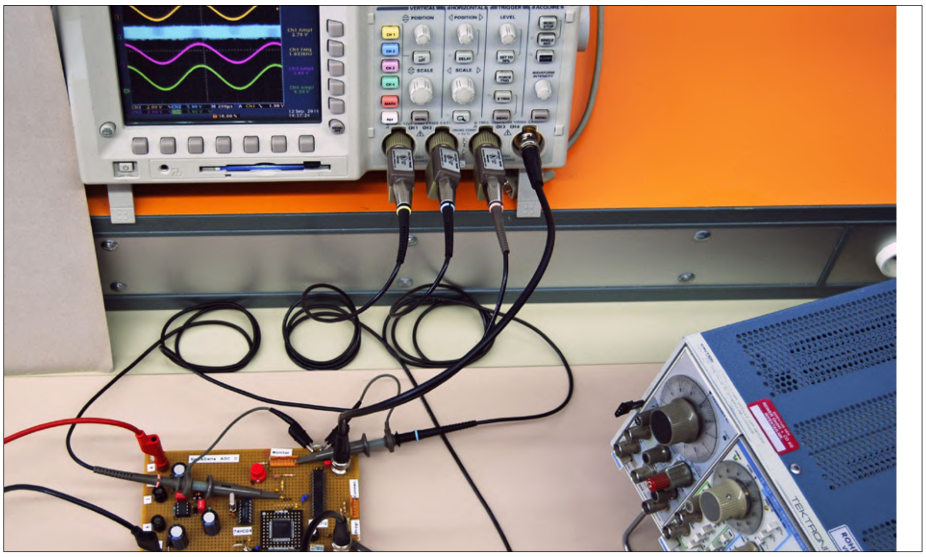


Figure 9. Test setup for time domain measurements.

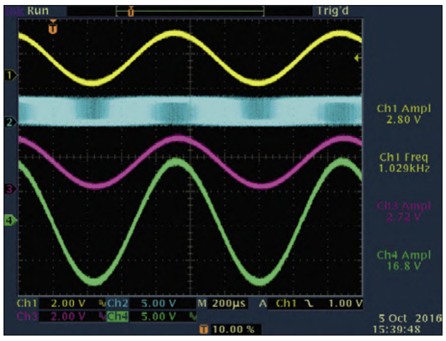


Figure 10. Time domain signals on the test arrangement with a nearly full-scale signal. Ch1: analogue in; Ch2: Mon1; Ch3: ANA; Ch4: analogue out.

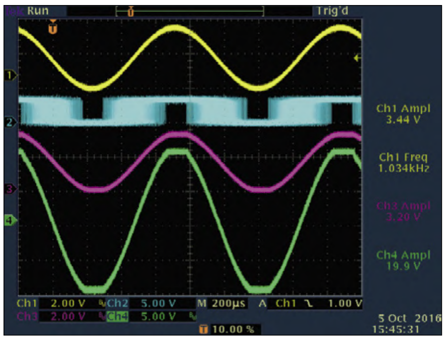


Figure 11. Time domain signals with overdriving. Ch1: analog in; Ch2: Mon1; Ch3: ANA; Ch4: analogue out.

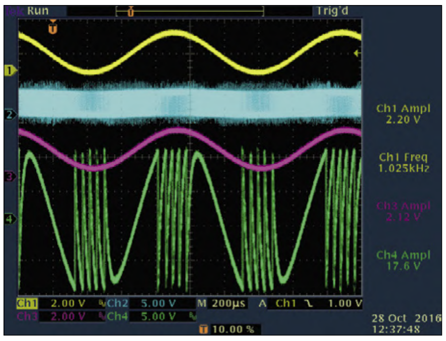


Figure 12. Time domain signals on the test arrangement with insufficient word length in the digital filter. Ch1: analog in; Ch2: Mon1; Ch3: ANA; Ch4: analogue out.

measurements.

Figure 10 shows the signals at the sigma-delta ADC input and outputs with a signal frequency of 1 kHz. The sinusoidal analog input signal (Ch1, yellow) has an amplitude of 2.8 V_{PP}, whuich is close to the maximum level. At the digital output (CH2, blue) you can see that the pulse density varies in proportion to the analog voltage. With a low input voltage the output is predominantly low, while with a high input voltage it is predominantly high.

In the ANA output signal (CV3, pink) you can see that the input signal is represented quite accurately in the digital output of the sigma-delta modulator. The slight attenuation of the amplitude is probably attributable to the passive RC low-pass filter. The analog DAC output (Ch4, yellow) with an amplitude of approximately ±8.4 V shows that the signal is also nearly full scale at this point. Its symmetry shows that the digital DAC input signal Q(7:0) is in twos-complement format.

The behaviour of the sigma-delta ADC when it is overdriven can be seen in **Figure 11**. Overdriving can be recognised by the fact that the sigma-delta modulator output Mon1 sometimes appears stuck at high or low. As a consequence, the analog DAC output remains at its maximum or minimum level during these peaks, but there is no twos-complement overflow. The precisely limited ADC output signal proves that the word widths in the digital filter are correct.

Figure 12 shows how the sigma-delta ADC behaves when the word width of the digital filter is too small. Here there is twos-complement overflow and the DAC output signal jumps between the maximum and minimum values, resulting in many strong harmonics.

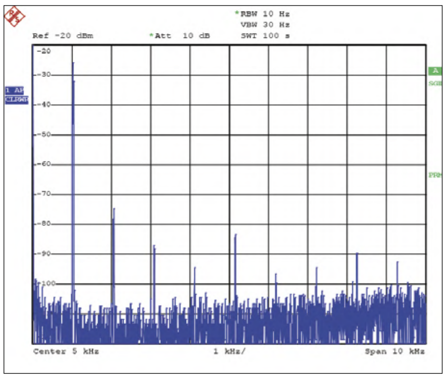


Figure 13. Spectrum for Figure 10 with 10 kHz bandwidth.

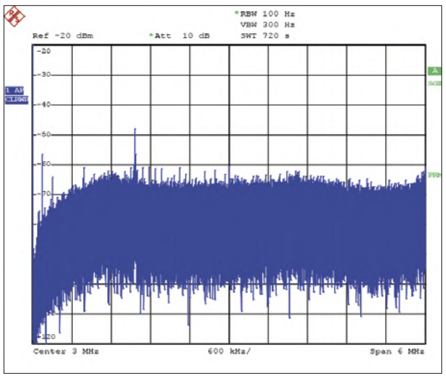


Figure 14. Spectrum for Figure 10 with 6 MHz bandwidth.

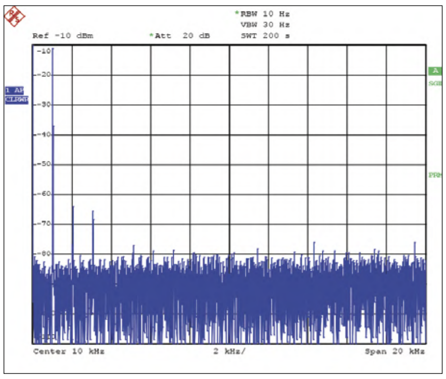


Figure 15. Spectrum for Figure 10 with 20 kHz bandwidth, but at the analog out point.

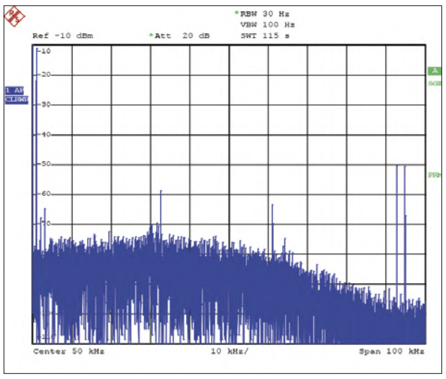


Figure 16. Spectrum under the same conditions as Figure 15 but with 100 kHz bandwidth.

Frequency domain measurements

For the frequency domain measurements, the test setup shown in Figure 9 was modified slightly. An LC low-pass filter was connected to the generator output to attenuate the harmonics of the test signal. The probe connected to the analyser input has an attenuation of 40 dB and provides impedance conversion for the 50 Ω analyser input.

Figure 13 shows the spectrum at the digital output Mon1 in the range of 0 to 10 kHz with a sinusoidal signal at a frequency of 1 kHz with an amplitude of 2.8 V_{pp} applied to the analog input. The noise level, at −100 dBm, is approximately 75 dB below the signal level at −25 dBm, which fits well with the theoretical considerations regarding the SNR. The harmonics are ≥50 dB below the signal level, indicating good linearity in the sigma-delta modulator.

Note that making measurements of this sort with an FFT analyser can lead to measurement errors because the sampling rates of these analysers are often too low relative to the measurement frequency range, causing the noise at higher frequencies to be folded into the measurement range.

Figure 14 likewise shows the spectrum at the digital output Mon1, in this case in the range of 0 to 6 MHz. Here you can clearly see that the noise level rises at higher frequencies. From around 1.2 MHz the noise level remains at approximately –60 dBm. The SNR at this point is only 35 dB.

Figure 15 shows the spectrum at the DAC analog output in the range of 0 to 20 kHz, again with a sinusoidal signal at 1 kHz with an amplitude of 2.8 V_{PP} applied to the analog input. The noise level, at –75 dBm, is approximately 65 dB below the signal level at –10 dBm.

As the sigma-delta ADC has a resolution of 8 bits, you would expect an SNR of 49.92 dB. However, as a result of oversampling there is a calculated additional gain of approximately 20 dB for a total of 70 dB, which fits well with the measured result. Here again, good linearity is indicated by the fact that the harmonics are ≥50 dB below the signal level.

Figure 16 likewise shows the spectrum at the analog output, in this case in the range of 0 to 100 kHz. Here you can clearly see the effect of the digital lowpass filter from the fact that the noise decreases with increasing frequency, unlike Figure 14. The symmetrical folding products of the input frequency about the 93.75 kHz DAC sampling frequency (f_{SO}) can be seen on the right in the figure.

Conclusion

If you are interested in PLDs and you enjoy experimenting, this article shows you how you can build relatively complex circuits with good technical characteristics using only a few external components. Of course, you do not have to limit yourself to copying the project described here. You can also draw inspiration from the concepts described in this article for your own projects, such as building an ADC with higher resolution (and narrower bandwidth). The formulas in this article have been kept to the minimum necessary for proper understanding. Nevertheless, they should be sufficient to allow you to modify and extend the basic circuit according to your own ideas. ◄

170566-02

About the Autor

Guido Nopper has a degree in engineering from the Furtwangen University of Applied Sciences and worked from 1981 to 1984 as an LSI MOS development engineer focusing on A/D converters and digital filters. Since 1984 he has been responsible for the entire hardware design of paper-processing office machines, in particular cheque printers and scanners.

Web Links

[1] The Evolution of Oversampling Analog-to-Digital Converters (PDF): https://bit.ly/2Z1q68G

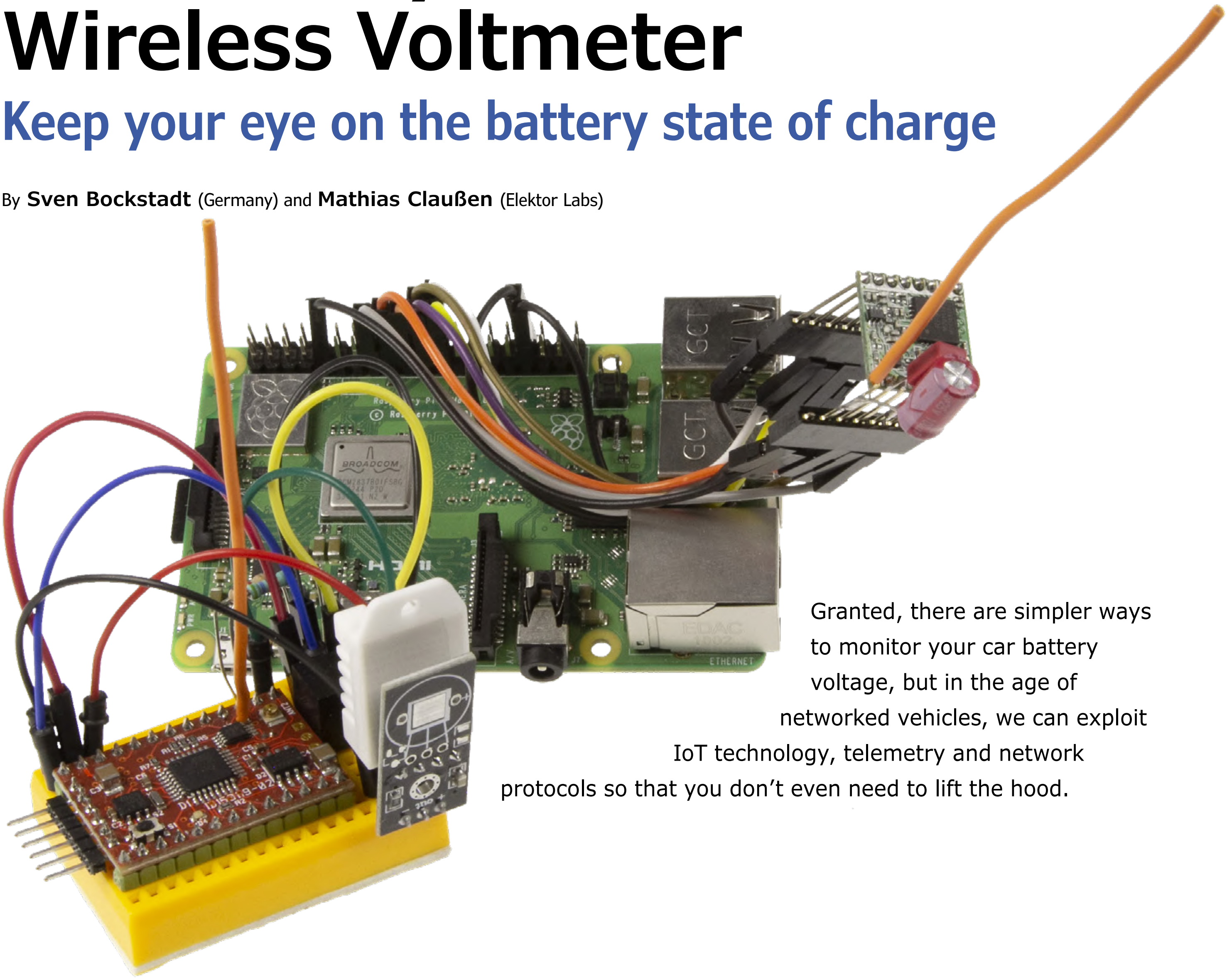
[2] Do Multirate Systems Have Transfer Functions?: www.dsprelated.com/showarticle/143.php

[3] CPLD XC95144XL data sheet: https://bit.ly/2XvhtTy

Car Battery Wireless Voltmeter

Keep your eye on the battery state of charge

By **Sven Bockstadt** (Germany) and **Mathias Claußen** (Elektor Labs)



Granted, there are simpler ways to monitor your car battery voltage, but in the age of networked vehicles, we can exploit IoT technology, telemetry and network protocols so that you don't even need to lift the hood.

It's true that modern vehicles are fitted with ever more sophisticated battery management systems but that does not mean that the car battery will always have a 100% charge. Modern start-stop engine management and emission regulations ensure that the battery is usually charged only in drive and acceleration mode. In addition, when the weather turns a bit colder the battery is subjected to temperatures that can reduce its charge capacity and lead to a spontaneous early-morning battery failure or a damaging deep-discharge condition.

In order to help the battery maintain a healthy state of charge the author Sven Bockstadt has developed an energy-saving solution that takes a quick battery voltage measurement at regular intervals. The voltage level is transmitted from the point of measurement to the evaluation unit via a wide-area LoRa radio link. Here the measured values are displayed and also stored for later graphical evaluation.

Development platforms such as the Arduino or Raspberry Pi offer a variety of compatible add-on boards with wireless radio capability; these provide a simple way to transfer the data without too much effort or hardware building. To keep things simple the author chose finished boards from the Arduino system as much as possible for his original project. Hence no board layout was required and external wiring was kept to a minimum. Monitoring software was also developed for the transmitter and receiver hardware, in order to be able to control and debug both devices via the serial interface. The data-logging function produces a continuously expanding CSV file that can be saved onto a microSD card and opened on a PC using Excel or some other spreadsheet software where it can be graphically evaluated with just a few clicks.

This wireless car voltmeter was first

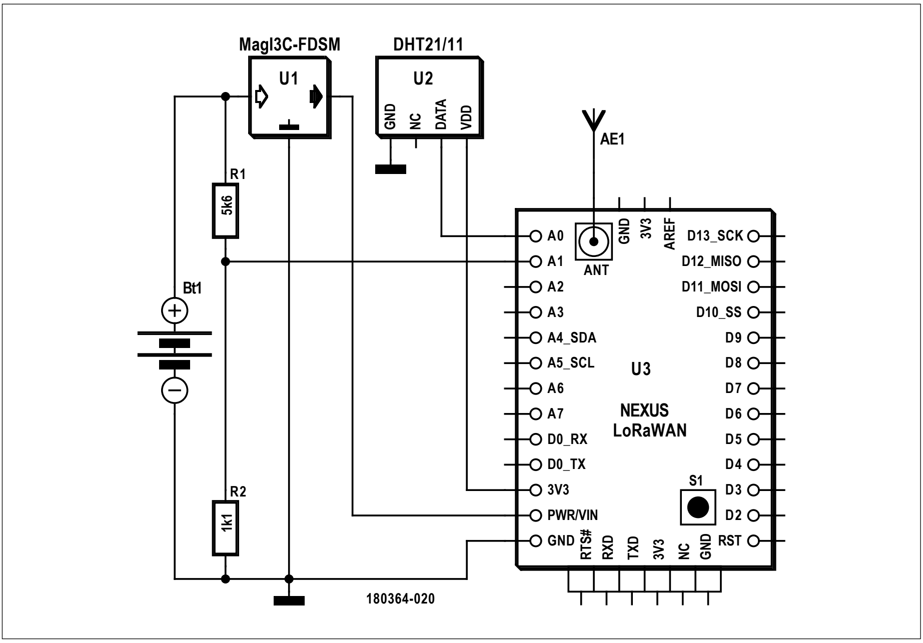


Figure 1. The transmitter consists of a voltage divider and the LoRa module. The optional weather sensor DHT21/11 uses just one of the many free analogue inputs.

PROJECT DECODER

ESP32 LoRa

Raspberry Pi

MQTT

entry level

➡ intermediate level

expert level

6 hours approx.

Soldering iron

€70 / £60 / $85 approx. (RPi not included).

introduced by the author some time ago on the Elektor Labs project platform [1]. For this release however, the hardware and software went through the mills of the Elektor Labs where some changes, improvements, simplifications, and quality control were made. What follows is the result…

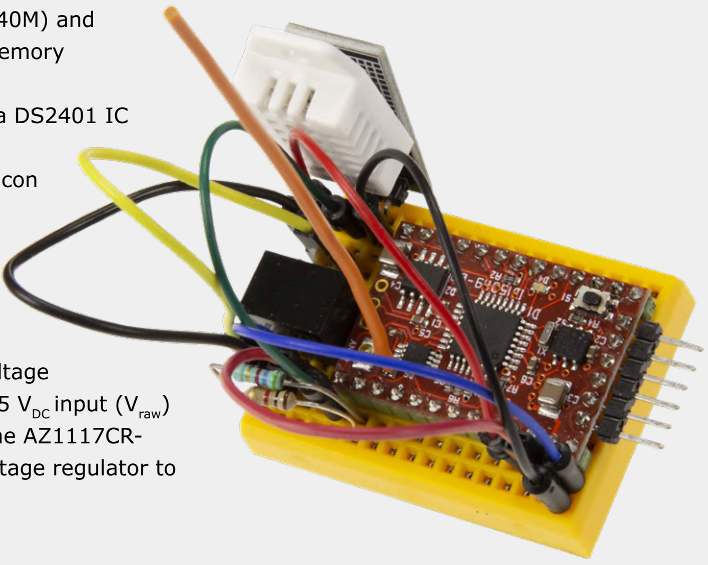
The transmitter

The transmitter used in the original design was an Adafruit Feather M0 RFM96 LoRa board containing a 433-MHz LoRa radio module type RFM96 from HopeRF [2]. This board however, requires an external analogue/digital converter with logging function and an external timer module to clock the data traffic (for energy-saving). This seemed like an unnecessary amount of work that would take up too much time so we replaced the Feather M0 with an Arduino-compatible LoRa Nexus board ([3], see **text frame**), this module not only includes an RFM95 LoRa module transmitting at 868 MHz [2] but also has an ATmega328P microcontroller at its heart, which conveniently has an internal 10-bit ADC. **Figure 1** shows that only a simple voltage divider network with R1 = 5.6kΩ and R2 = 1.1kΩ needs to be connected at the input to the converter (input 1 of the controller) in order to scale the battery voltage to the ADC range of the Atmega328P. The LoRa Nexus board has a flash memory type W25X40CLSNIG and a real-time clock, eliminating the need for external components. Timing of data transfer is taken over by the controller firmware.

To supply power to the module, a DC/DC converter with fixed voltage output from Würth Elektronik [4] is used. This generates a stable +5 V output voltage at a maximum current of 1 A from an input voltage in the range of 8 to 28 V. If you think these characteristics remind you of the good ole 7805 linear regulator that will be no surprise because this small switching regulator is one of a series advertised by Würth as a direct replacement for the L78xx family of linear regulators. You don't even need to

The LoRa Nexus Board

The 23 x 33 mm LoRa Nexus Board uses the 8-Bit ATmega328P-MPU von Atmel — the same controller used in the Arduino Nano. The board includes a RFM95W-LoRa radio module mounted on the PCB. Other peripherals on the board include a real-time clock with 64-bytes of SRAM (MCP7940M) and a 4-Mbit-Flash memory (W25X40CL). Also on board is a DS2401 IC which provides a unique 48-bit 'silicon serial number'. This is used to identify the board in a network. The operating voltage is 3.3 V_{DC}, a raw 5 V_{DC} input (V_{raw}) is regulated by the AZ1117CR-3.3TRG1 LDO voltage regulator to provide 3.3 V.



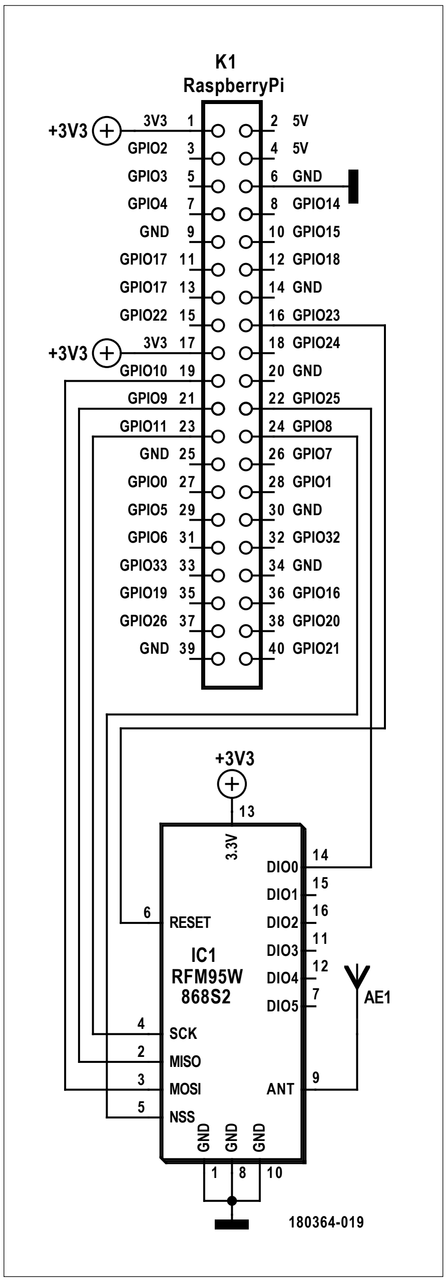


Figure 2. The receiver uses an RFM95 connected to a Raspberry Pi-GPIO connector.

add any capacitors at the input or output. The +5 V output voltage of the switching regulator is reduced on board by another (this time linear) voltage regulator to 3.3 V, which not only supplies all the ICs of the module, but could also be used as the I²C bus voltage (not used here) and serve as thr reference voltage for the ADC at the same time.
It also powers the DHT21 / 11 ‘weather sensor’, which measures humidity and temperature. The software evaluates the environmental conditions at the battery location in order to make a better estimate of what performance we can expect from the battery. The sensor connects to the ADC input 0.
These few parts have been built on a breadboard and are programmed using a suitable FTDI cable (V_{CC} = 3.3 V) which plugs into the FTDI connector on the Nexus board.

The receiver

Originally the receiver was also built with an Adafruit Feather M0 board, again associated with an external ADC/Logger, a timer module and voltage regulator. After tweaking the design in the Elektor lab, the only thing remaining of the original is the RFM95 LoRa modem module [5] which now connects to a few of the GPIO pins on a Raspberry Pi header (see **Figure 2** and **Table 1**). The controller board is now a Raspberry Pi model 3+ (instead of an arduino controller), which offers much more possibilities for data processing than was possible with the Arduino. Another reason for choosing

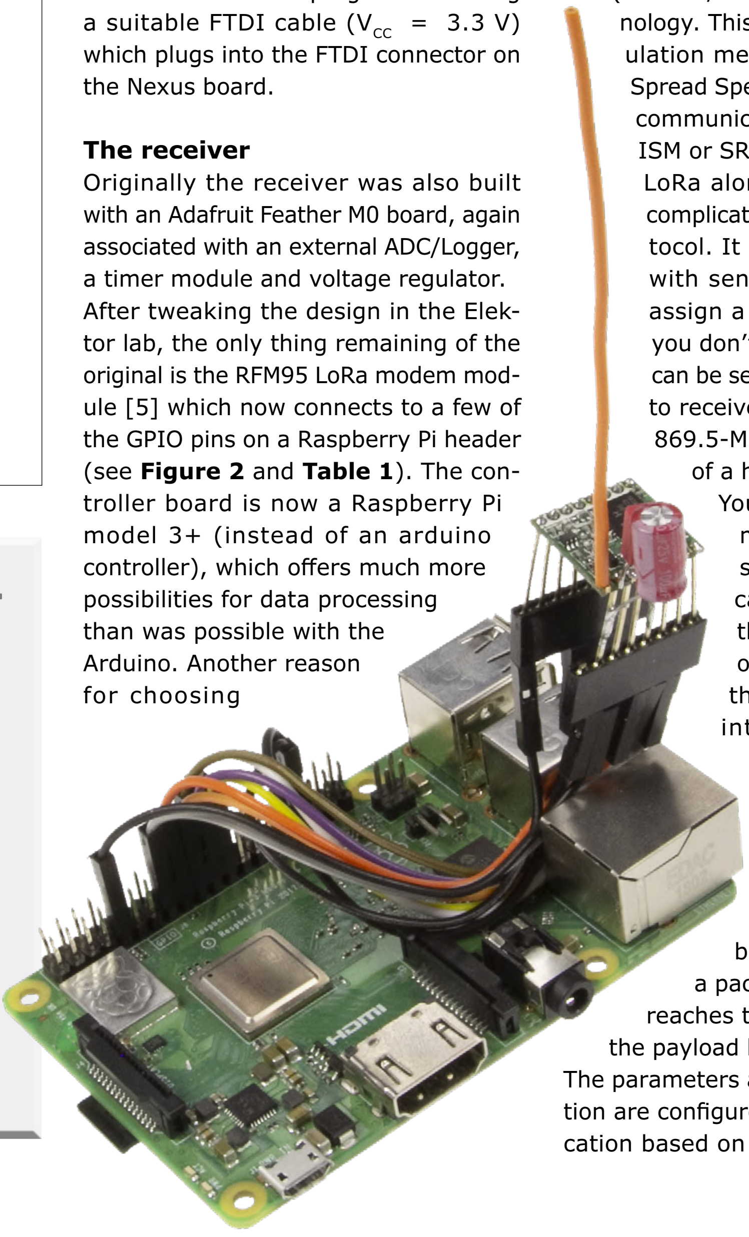
Table 1: RFM95 to RPi Connections.

RFM95	Raspberry Pi
SCK	GPIO11
MISO	GPIO09
MOSI	GPIO10
NSS	GPIO08
RESET	GOIO22
DIO0	GPIO25

the RPi was easier control of a display monitor. The RPi also outputs a clean 3.3-V supply voltage for the radio module so no additional regulators are required. Both modules need to be equipped with antennas. Two 86 mm long wire pieces are all that’s necessary. As far as the hardware goes, that’s it — the rest is up to the software.

Radio link using LoRa-PHY

The radio link used here uses the LoRa (LoRa RF/PHY) communication technology. This just defines the modulation method (similar to Chirp Spread Spectrum or CSS) and the communication frequency in the ISM or SRD band.
LoRa alone does not specify a complicated communication protocol. It can use just a header with sender address. You can assign a receiver address, but you don’t have to: The receiver can be set (as in our application) to receive all data packets on a 869.5-MHz carrier, irrespective of a header address.
You simply indicate how many bytes you want to send and the communication is carried out by the Semtech chip without the need for any further action. To improve interference immunity the LoRa module adds a preamble to the data which can be set by the user. The receiver is then set to the same preamble sequence and when a packet with this preamble reaches the receiver, it unloads the payload bytes.
The parameters at the RPi receiver station are configured using a small application based on the Radiohead library



COMPONENT LIST

R1 = 5.6kΩ 5%, 0.25W, 250V
R2 = 1kΩ 5%, 0.25W, 250V
MOD1 = MagI³C-FDSM Fixed Step Down Regulator
MOD2 = DHT11 from the Elektor-ESP32-Smart Kit
MOD3 = LoRa Nexus Board
MOD4 = RFM95 Ultra LoRa Transceiver Module (868 MHz)
Raspberry Pi 3 B+ Starter Kit + free RPi compilation
J1,J2 = jumper wires from the Elektor ESP32-Smart-Kit
(optional) Large plug board
(optional) FT232R-USB/Serial Bridge/ BOB (Elektor no. 110553-91)

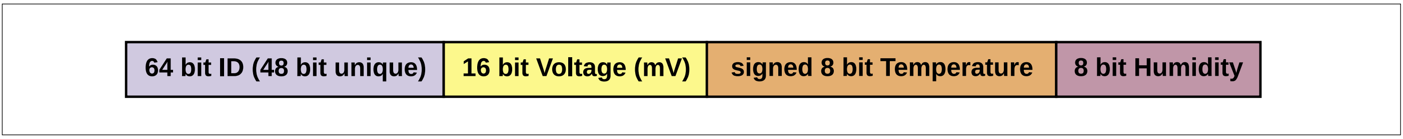


Figure 3. The simple user data protocol consists of a 64-bit payload ID with three measurement values.

[6]. The same default settings need to be used for this station.
Software running on the RPi reads the received LoRa data packets and looks for a message with 0x20 as sender, which is transmitted with a payload length of 12 bytes to 0x10 as the receiver. If such a message is found, a payload packet with a structure as shown in **Figure 3** should be received. The payload ID is followed in sequence by the voltage value (16-bit), the temperature value (8-bit) and the air humidity value (8-bit).
The 64-bit ID is made up of the unique 48-bit address provided by the DS2401P 'silicon serial number' IC on the LoRa Nexus board (usually found on a label on the board, or can also be read by the Arduino), and two bytes which are always the same for the wireless car voltmeter application.

MQTT and a broker called Mosquitto

The rest of the software in the RPi handles the incoming data and sends it to an MQTT broker with the help of *libmosquitto* library routines. The library uses the 64-bit ID to decide if the message belongs to its own station. MQTT (see **text frame**) decribes a lightweight, efficient protocol for exchanging data between different (IoT) devices. If you want to know more about MQTT, there is a link [8].
With MQTT you always need to access a kind of server called an MQTT Broker. A well-known MQTT open-source broker for various platforms is Mosquitto, which is now also included in Raspbian. The broker receives data source MQTT messages and sends them out to data consumers who have 'subscribed' to this message topic.

In our case, the Lora receiver publishes the received Lora packets under various MQTT topics '*car\battery\voltage*', '*car\interior\temperature*' and '*car\interior\humidity*'. So all values can be processed individually. The message for the MQTT broker are written in JSON, so the three elements can be easily and flexibly processed.
The graphical programming language Node-RED — which has been specifically developed for IoT applications — can be used to create software that subscribes to MQTT topics. This works with the received messages so that you can use it to do things like switch appliances in your home, send emails or just display the message information on a monitor with some neat graphics (**Figure 4**). Such an application is very easy to develop using Node-RED [8].

A practical installation

The installation of Node Red and MQTT on a RPi has already been described in [19]. When the installation of Node Red and mosquitto is completed, there is still some preparation needed to compile the code for the software to **run on the RPi**. Only a few commands need to be entered in the terminal to install some dependencies for the software.

```
apt-get install libmosquittopp-dev
apt-get install libmosquittopp1
apt-get install libmosquitto1
```

In addition, the *bcm2835* library [9] must be installed on the RPi. Extract the file and go to the directory. From there run `./configure` to prepare the code for compilation. After this is done and no errors are displayed, you can enter:

```
make
sudo make check
sudo make install
```

MQTT and Mosquitto

According to the MQTT protocol (Message Queuing Telemetry Transport), data and commands between (IoT) *devices* are not exchanged directly but via a central MQTT server (*broker*). MQTT is efficient, secure and low on CPU resources. The protocol is message-oriented: A data source (publisher) sends data only to the broker to *publish*. A client does not have to constantly ask the server whether there is new data from the sender, but *subscribes* to a message *topic* and any messages arriving at the broker will be sent to subscribers of the topic. Senders and receivers of messages are thereby completely decoupled by the broker - anyone who provides data does not have to worry about who receives that data.
A message contains, among other things, the elements
- Topic is the topic of the message. Topics are simple strings separated by slashes. A topic such as car/battery/voltage contains a hierarchy of objects — the transmitter in the car, the battery and its voltage measurement value.
- Payload is the content of the message, usually commands or data.

To use MQTT (versions 3.1, 3.1.1 and 5.0) you need a broker like the popular Eclipse Mosquitto [17]. This open source broker is lightweight can be installed easily on any low-power single-board computer, including a Raspberry Pi (absolutely problem-free via the main repository).

Source: FHEM [7].

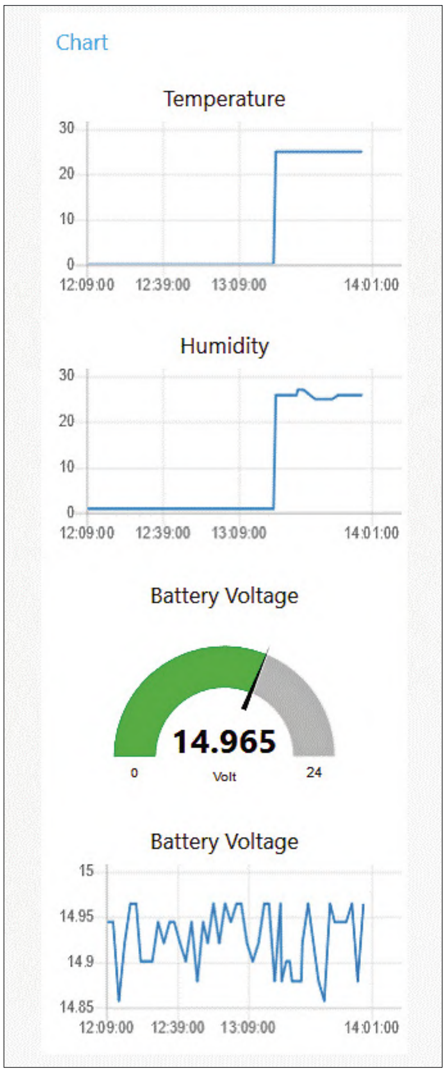


Figure 4. A neat looking GUI handles the data.

This should install the library in place. When all this is finished we are now ready to compile the code for the receiver. Get the code from the git [10] repository and copy it to the RPi. Go to the directory and run `make`. After that the receiver (hopefully!) Is ready for use. Unfortunately, you must run the program with root privileges to gain access to some of the Raspberry Pi subsystems. Using `./ sudo main` will start the software.

To save the data, you can use sqlite3 as a database engine. Install it with:

```
apt-get install sqlite3
```

The Sqlite database makes it possible to store all the information supplied by the Car-Voltmeter. You can then go to the home-folder of the RPi-users and create a database. Open the terminal and enter:

```
cd ~
sqlite3 carsensor.db
```

A new database is now created that still needs to be filled with three tables:

```
sqlite> CREATE TABLE battery_
   voltage ( id INTEGER PRIMARY
   KEY AUTOINCREMENT, uuid
   NUMERIC, voltage NUMERIC,
   timestamp DATETIME DEFAULT
   CURRENT_TIMESTAMP);
sqlite> CREATE TABLE humidity
   ( id INTEGER PRIMARY
   KEY AUTOINCREMENT, uuid
   NUMERIC, humidity NUMERIC,
   timestamp DATETIME DEFAULT
   CURRENT_TIMESTAMP);
sqlite> CREATE TABLE temperature
   ( id INTEGER PRIMARY KEY
   AUTOINCREMENT, uuid NUMERIC,
   temperature NUMERIC,
   timestamp DATETIME DEFAULT
   CURRENT_TIMESTAMP);
sqlite> COMMIT;
sqlite> .exit
```

This will create the required tables for the database. Now only Node-RED has to be installed:

```
cd ~/.node-red
npm i --unsafe-perm
   node-red-node-sqlite
```

After this is done, the RPi needs to be rebooted. Almost all the components are now running. It is now necessary to import the flowcode [10] into Node-RED, which is the same as in the Monster LED clock from Elektor May/June 2019 edition.
The last thing left to do on the receiver side is to start the compiled software. Go to *~/LoRA_MQTT_VoltMeter* and start the compiled program with `sudo ./main`. This is only necessary for the time being because later you can move the software to the RPi autostart area.
Now it's time to check out the **transmitter side** which uses the Arduino-compatible LoRa Nexus board [3]. The code has the task of initializing the hardware and outputting a new set of readings every 120 s. We need to set up a few libraries: the patched version of the *DS2401* library, which you will find as a download in GitHub [11], and a library based on Paul Stoffregen's *OneWire Library* [12]. To make use of energy saving features,

LoRa-PHY

The LoRa (Long Range) communication standard is specifically designed to send data from IoT sensors. With a data rate slower than 50 kbit/s, over long distances (10 km and more, depending on conditions) its philosophy is 'slow but sure' and low power requirements. LoRa is particularly suitable for non-urgent transmissions where a delay does not matter too much. LoRa devices transmit in Europe in the ISM band (433.05-434.79 MHz) or (as here) in the SRD band (863-870 MHz).
LoRa is the lowest physical layer in the OSI model that can be used by higher layers such as LoRaWAN (Long Range Wide Area Network), but also by other higher layers.

Unlike traditional wireless systems based on FSK modulation, LoRa uses a type of chirp spread spectrum modulation (CSS) invented by Semtech. By varying the spreading factor, LoRa can tune the data rate for the sensitivity at a fixed channel bandwidth and thereby achieves an enormous range with an extremely high transmission link budget of about 155 to 170 dB. In addition, its use of forward error correction coding makes the transmission less susceptible to interference.
Further basic information on LoRa can be found at the developer portal of the company Semtech [18].

the *LowPower* library [13] should also be installed. Since we also want to use the humidity and temperature sensor, we need the Adafruit *DHT* library [14]. And to get the LoRa module working, we also need the patched *RadioHead* version, which can also be found in the download package for the project [10]. The firmware of the receiver is loaded into the Arduino-IDE, all the named libraries are integrated and the program is then compiled. An FTDI cable (3.3 V) is used to upload the completed firmware to the LoRa Nexus Board.
The code for all the programs mentioned here can be retrieved from the Elektor GitHub repository [10].

@ **WWW.ELEKTOR.COM**

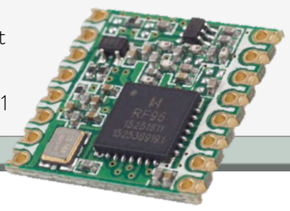
→ Elektor ESP32 Smart Kit
www.elektor.com/elektor-esp32-smart-kit

→ LoRa Nexus Board
www.elektor.com/lora-nexus-board-arduino-mini-shape

→ RFM95 Ultra LoRa Transceiver-Module
www.elektor.com/rfm95-ultra-lora-transceiver-module-868-915-mhz

→ Raspberry Pi 3 B+ Starter Kit
www.elektor.com/raspberry-pi-3-model-b-plus-starter-kit

→ FT232R-USB/Serial Bridge/BOB
www.elektor.com/ft232r-usb-serial-bridge-bob-110553-91



Ready to install?

One thing that Matthias Claußen discovered only on the Elektor lab bench is that the RFM95 should not be connected using wire jumper leads because this can lead to unstable operation and unpredictable resets. For this reason, the RadioHead library has been patched a bit so that it checks the modem configuration and, if necessary, reinitializes if something is not set correctly.
The prototype on a small breadboard was then successfully put through its paces in the lab. If you are now wondering whether you should go ahead and install the wireless car voltmeter in a vehicle, our advice would be **no**! Vehicle electrics are notoriously noisy environments. It's important to install some protection circuit and filters to attenuate interference signals and protect the electronics.
Some of the electrical problems you are likely to encounter are outlined in the article 'sources of interference in the automotive industry' [15]. However, the countermeasures you will need to take to protect the wireless car voltmeter module will depend (too) heavily on the type of vehicle and on the installation situation; there is no 'one-size fits-all' solution. ◀

180364-02

Web Links

[1] Project page at Elektor Labs: www.elektormagazine.com/labs/wireless-car-voltmeter

[2] LoRa module: www.hoperf.com/modules/lora/index.html

[3] LoRa Nexus Board: www.elektor.de/lora-nexus-board-arduino-mini-shape

[4] DC/DC Module Würth 173010578: https://katalog.we-online.de/en/pm/MAGIC_FDSM_FIXED_OUTPUT_VOLTAGE

[5] RMF95 as a LoRa-Modem: www.elektor.com/rfm95-ultra-lora-transceiver-module-868-915-mhz

[6] RadioHead library: https://github.com/hallard/RadioHead

[7] The MQTT protocol: https://en.wikipedia.org/wiki/MQTT

[9] Monster LED Clock with Wi-Fi and Temperature Display: www.elektormagazine.com/magazine/elektor-96/42659

[10] BCM2835: www.airspayce.com/mikem/bcm2835/

[11] GitHub repository for the project: https://github.com/ElektorLabs/180364-wireless-car-multimeter

[12] DS2401 library: https://github.com/sindrehal/Arduino_DS2401

[13] OneWire library: https://github.com/PaulStoffregen/OneWire

[14] Low-Power library: https://github.com/rocketscream/Low-Power

[15] DHT sensor library: https://github.com/adafruit/DHT-sensor-library

[16] Interference sources in Automotive applications: www.elektormagazine.com/magazine/elektor-88/42442

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

[18] LoRa developers portal: https://lora-developers.semtech.com/

[19] Bedroom Clock with Outside Temperature Based on ESP32:
www.elektormagazine.com/labs/bedroom-clock-with-out-side-temperature-based-on-esp32

HOMELAB PROJECT

A Time-Corrected Woofer Concept

for the lowest lows

By **Jan Breemer** (The Netherlands)

Although the DIY construction of audio equipment (amplifiers, speaker boxes) is almost never worthwhile these days (at least from a cost perspective), there is fortunately still a hard core of enthusiasts who continue to be active. This is also the case here. It was the author's intention to design a woofer that would be able to reproduce the very lowest frequencies as faithfully as is possible. In this article he describes the considerations that led to the desired outcome.

To prevent any disappointments: this article does not describe a ready-to-build design that you can make. Rather, the author would like to pass sufficient information to the interested reader, so that they can design and build their own woofer.

The problems

Most commercial audio systems (and also most DIY systems) are based on an ordinary stereo amplifier, that is, one that has an amplifier for a right channel and one for a left channel in the same enclosure, and that takes care of the entire audio spectrum. In this scenario, the loudspeakers are provided with passive cross-over filters that split the audio spectrum into two or three parts, which are then passed to loudspeakers that are optimised for each of these.

And with these cross-over filters the problems begin. In most cases they spoil the damping of the loudspeakers; they also make matching the sensitivity of the separate speakers difficult. And finally, the parts that are used, are often designed to 'a cost'.

And another problem is that most (low-frequency) loudspeaker boxes are built according to the bass-reflex principle, where the inside of the box is connected through a reflex tunnel to the outside world. These reflex systems are either unable, or barely capable, of reproducing very low frequencies (below about 30 Hz). Worse is that these systems,

below the resonant frequency, behave as a steep (fourth-order) high-pass filter. This steep filter causes delays and ringing that does the impulse response of the system no favours.
And finally: when we nevertheless drive such a box with extremely low frequencies, the cone of the speaker will indeed exhibit a very considerable excursion, but audible sound will not be produced as a consequence of the acoustic short circuit. Furthermore, the higher frequencies will be distorted as a result of these extreme excursions.

The approach

A completely closed box offers a number of advantages, and is the reason that the author used this as the starting point for his considerations. Below the resonant frequency, the sound pressure falls off less steeply at 12 dB/octave, so behaves as a second-order high-pass filter. To compensate, a special filter is introduced before the final power stage (as an aside: every speaker, and therefore also the woofer, has its own power amplifier); the remarkable feature of this filter is the correction/compensation of the self-resonance of the system.
And what is also not insignificant: the behaviour of a closed box can be relatively easily modelled with good accuracy. To make the life of the interested reader more convenient, the author has developed a computer program that takes care of all the difficult computing effort and displays the results graphically; this program can be downloaded free from the web page for this article (executable files for Linux and Windows, and the source code) [1].

Two gentlemen

These days, the design of a good loudspeaker box is no longer the extremely drawn-out process of trial, refinement, more attempts, further refinements and so on – and for that we can mainly thank the two gentlemen Thiele and Small. The Thiele/Small parameters named after them are perfectly suitable for predicting the behaviour of a woofer (we mean here the combination of loudspeaker and enclosure) with considerable accuracy.
We have summarised the most relevant TS (Thiele-Small) parameters in **table 1**. These, by the way, are not all of the TS parameters – there are more of them, but their effect on the final result is comparatively small, so we don't take them into consideration here.

Table 1.

Parameter	Description	Unit
M_{ms}	mass of the cone + dragged air	kg
C_{ms}	compliance (mobility) of the cone suspension	m
R_{ms}	mechanical resistance (damping) of the cone	kg/s
R_e	DC resistance of the voice coil	Ω
L_e	self-inductance of the voice coil	H
B_l	'power factor' of the voice coil	N/A or Tm
S_d	effective surface area of the cone	m²
X_{max}	maximum permitted cone displacement	$mm_{peak\text{-}peak}$
C_{bx}	compliance of the air in the box	m
R_{bx}	damping from the damping material (such as wool, Dacron) in the box	kg/s
Q_{tc}	total quality factor of the loudspeaker chassis	

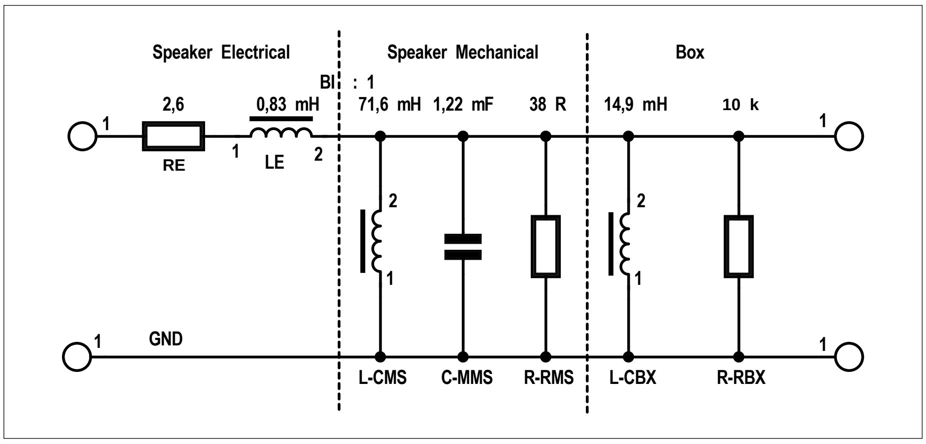


Figure 1: The electrical model for the loudspeaker/box combination. The values are for the ScanSpeak 30W4558T00 in a closed box with a volume of 41 l.

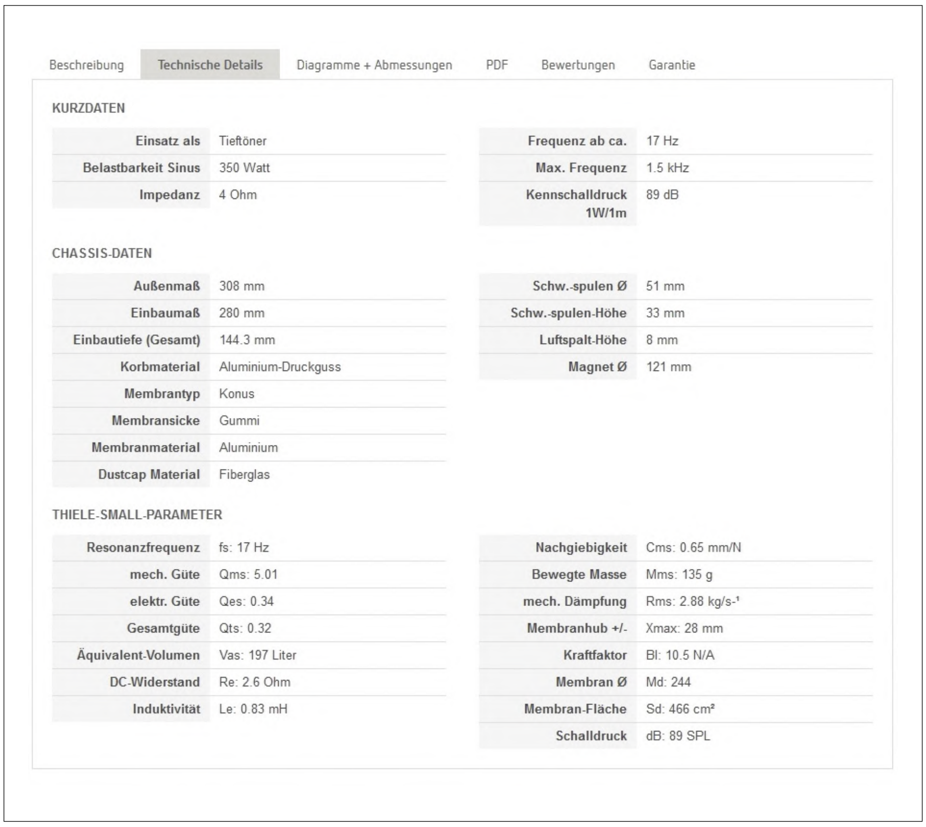
Electrical model

We now have a large number of mechanical and acoustical parameters and characteristics of the loudspeaker and the box that we have in mind; to compute with those conveniently and on paper (that was in the past, these days we use a simulator), we transform these into electrical equivalents. This enables the possibility of using any arbitrary simulator.
L_e and R_e are already electrical quantities; we can put these directly into the model. For the conversion of the mechanical quantities we use a logical approximation where the current through the voice coil is proportional to the force on the cone. It then turns out that the mass of the cone is equivalent to a capacitance, while the spring force of the suspension and of the air in the box end up in the model as self-inductances.

We can imagine the transformation factor as an 'ideal' transformer with a turns ratio of Bl:1. Via this 'transformer' the mechanical force is transformed into an electric current and the speed to an electric potential (and vice-versa, of course). And just as with an ordinary transformer, in the transformation of impedances, the parameter Bl appears squared.
When cast in the form of an equivalent electric circuit, the electrical model appears as sketched in **Figure 1**.

Summarizing, the transformations are as follows:

L-CMS = $C_{ms} \times B_{l2}$
C-MMS = M_{ms} / B_{l2}
R-RMS = B_{l2} / R_{ms}
L-CBX = $C_{bx} \times B_{l2}$
R-RBX = B_{l2} / R_{bx}


Beschreibung | Technische Details | Diagramme + Abmessungen | PDF | Bewertungen | Garantie

KURZDATEN

Einsatz als	Tieftöner	Frequenz ab ca.	17 Hz
Belastbarkeit Sinus	350 Watt	Max. Frequenz	1.5 kHz
Impedanz	4 Ohm	Kennschalldruck 1W/1m	89 dB

CHASSIS-DATEN

Außenmaß	308 mm	Schw.-spulen Ø	51 mm
Einbaumaß	280 mm	Schw.-spulen-Höhe	33 mm
Einbautiefe (Gesamt)	144.3 mm	Luftspalt-Höhe	8 mm
Korbmaterial	Aluminium-Druckguss	Magnet Ø	121 mm
Membrantyp	Konus		
Membransicke	Gummi		
Membranmaterial	Aluminium		
Dustcap Material	Fiberglas		

THIELE-SMALL-PARAMETER

Resonanzfrequenz	fs: 17 Hz	Nachgiebigkeit	Cms: 0.65 mm/N
mech. Güte	Qms: 5.01	Bewegte Masse	Mms: 135 g
elektr. Güte	Qes: 0.34	mech. Dämpfung	Rms: 2.88 kg/s-¹
Gesamtgüte	Qts: 0.32	Membranhub +/-	Xmax: 28 mm
Äquivalent-Volumen	Vas: 197 Liter	Kraftfaktor	Bl: 10.5 N/A
DC-Widerstand	Re: 2.6 Ohm	Membran Ø	Md: 244
Induktivität	Le: 0.83 mH	Membran-Fläche	Sd: 466 cm²
		Schalldruck	dB: 89 SPL

Figure 2: The TS parameters for loudspeakers from reputable manufacturers are not hard to find.

The compliance (mobility) of the air in the box depends — naturally — on the volume of the box, and on the (square of the) effective surface area of the loudspeaker cone (we won't bother you with the derivation of this relationship):

$$C_{bx} = 7.14 \times 10^{-6} \times V_{box} / S_{d2} \qquad [m]$$

Now in practice

So far now we have listed a number of theoretical dependencies and proposed an electrical model — that is all good and well, but not really satisfactory, because what can we actually do with all that? To state it more definitively: where do we obtain all the values that we have to enter into all those equations and in the electrical model? And what do we finally get out of it?

The easiest thing first: every self-respecting manufacturer of quality loudspeakers has measured the TS parameters of their loudspeaker chassis in good conscience, and makes these available free of charge. The author, for the loudspeaker system developed using the above considerations, used a ScanSpeak type 30W4558T00. A brief search on the internet effortlessly produces the necessary values (**Figure 2**).
The values from Figure 2 are also used to calculate the 'component' values that are shown in Figure 1.

And now it gets interesting: we are going to determine the transfer function of the loudspeaker system – because that is really what we are after: how the output voltage of the power amplifier is translated into sound pressure (SPL = Sound Pressure Level, usually specified at a distance of 1 m from the loudspeaker). For this we consider that the electrical potential at the output of the electrical model (Figure 1) corresponds to the speed at which the cone moves – and this will (instinctively) surprise no one that this proportional scaling is again the transformation factor Bl. It holds:

$$v = U / B_l \qquad [m/s]$$

where v = cone speed and U = output voltage of the electrical model.

When we combine that with the surface area of the cone, we find the volume V_{air} of the displaced air:

$$V_{air} = v \times S_d \qquad [m^3/s]$$

In 1954, Leo Leroy Beranek derived that the sound pressure SPL at a distance r from a spherical radiator (or from a point audio source) in free space is equal to:

$$P_{SPL} = V_{air} \times \rho \times f / (2 \times r) \qquad [N/m^2]$$

Here ρ is the density of the air (1.2 kg/m³). In the above equation it is assumed that the sound source is small, compared to the wavelength of the sound; if we assume a frequency of 30 Hz (this entire story is, after all, about low-frequency transducers), then the wavelength amounts to about 10 m and we can use the formula without any problem.
We can consider the distance r as the radius of a sphere with the sound source in the middle. Now, a sound source freely suspended in space doesn't really happen in practice; we have to deal with a realistic loudspeaker box that is standing on the floor. That means that the sound energy is not spread across an entire sphere but is spread only across a hemisphere and that results in twice the sound pressure.

Finally we convert the sound pressure into dB and for that we use the reference level:

$$0 \text{ dB} = 2 \times 10^{-5} \qquad [N/m^2].$$

Zobel

For audiophiles it is always interesting to know what impedance is 'seen' by the output of the amplifier. Most people, by far, don't worry too much about that, but: an amplifier is generally happier driving a load that behaves, as much as possible, as a pure resistance. The distortion will be a little smaller when the current and voltage are mostly in phase. And in this case a Zobel network can prove to be of good service.
This network is connected in parallel with the loudspeaker connections. It has barely any influence on the total transfer function, but it does affect the current that the amplifier must supply.
The network consists of two parts: one part compensates for the self-inductance Le of the voice coil, while the other part compensates for the resonator that is formed by the mass and the spring force of the loudspeaker in the box.
It is not all that difficult to calculate the component values. For the two resistors

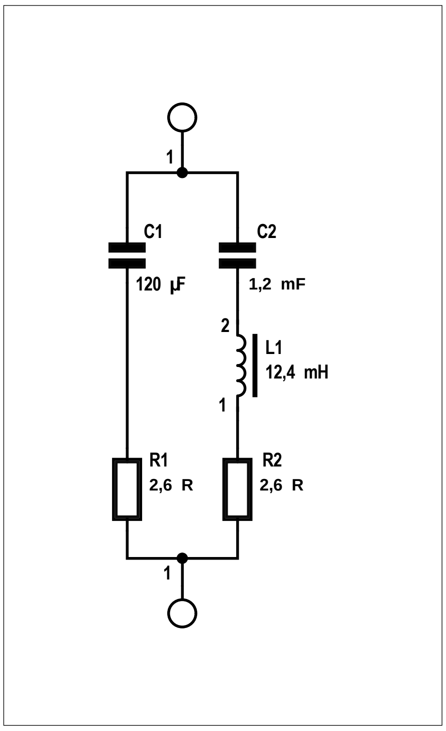
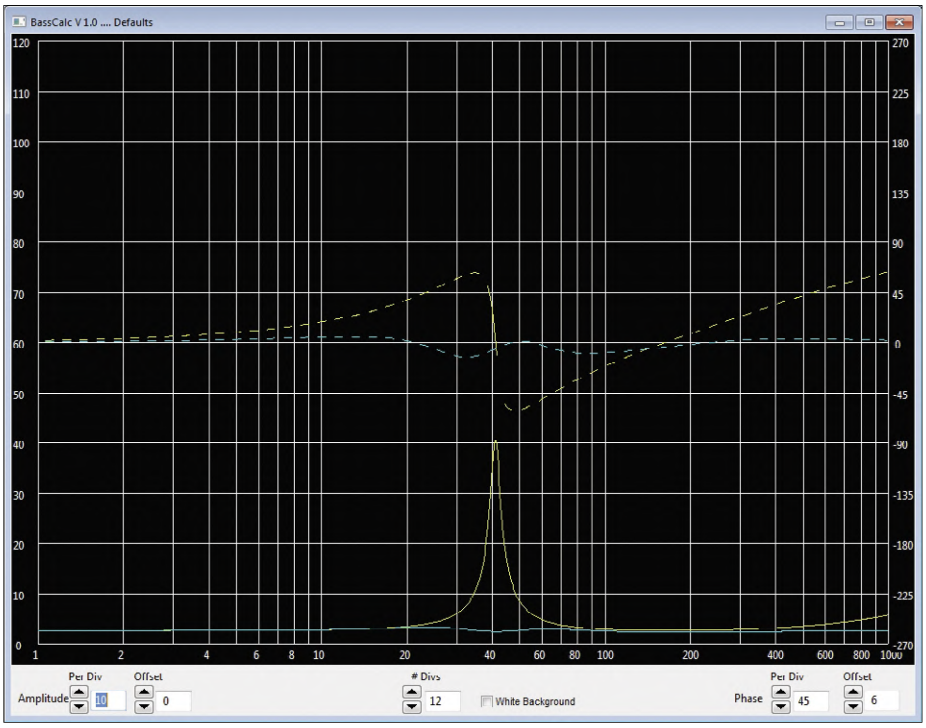


Figure 3: The Zobel network compensates for the resonant peak of a loudspeaker.



Amplitude Per Div 10 Offset 0 # Divs 12 White Background Phase Per Div 45 Offset 6

Figure 4: Here the effect of the Zobel network can be seen clearly: yellow is the transfer impedance without network, and blue with Zobel network.

we use the same value as the Re of the loudspeaker (in the ScanSpeak example that is therefore 2.6 Ω). C1, together with R1, must have the same time-constant as Re with Le. We then find:

$C1 = L_e / (R_e \times R1)$

For the series-resonant circuit we use C-MMS (see Figure 1) as the value for C2, and for L1 we take the parallel value of L-CMS and L-CBX (again see Figure 1). Granted: C2 with a value of 1.22 mF is a ‘fatty’ if we used a foil capacitor for that. However, it is also possible to use two electrolytic capacitors in anti-series (where the middle of the series connection is connected to the power supply voltage of the amplifier so that the electrolytic capacitors can never become reverse-biased).
Also inductor L1 at 12.4 mH is not particularly small. In theory a gyrator is also a possibility here, but that has the disadvantage that one of the loudspeaker connections has to be connected to ground. That is why the author used an inductor with a (closed) iron core; this is not a problem here because the Zobel network is not in the signal path. We just have to make sure that the core does not saturate.

In **Figure 4** the beneficial effects of the Zobel network can be clearly seen.

Compensation filter

Now we arrive at the final part of our considerations: the compensation filter that compensates for the attenuation of the low frequencies below the resonant frequency of the box (12 dB/octave, as mentioned earlier). Moreover, the filter also compensates for the self-resonance of the system. The result is an exception-

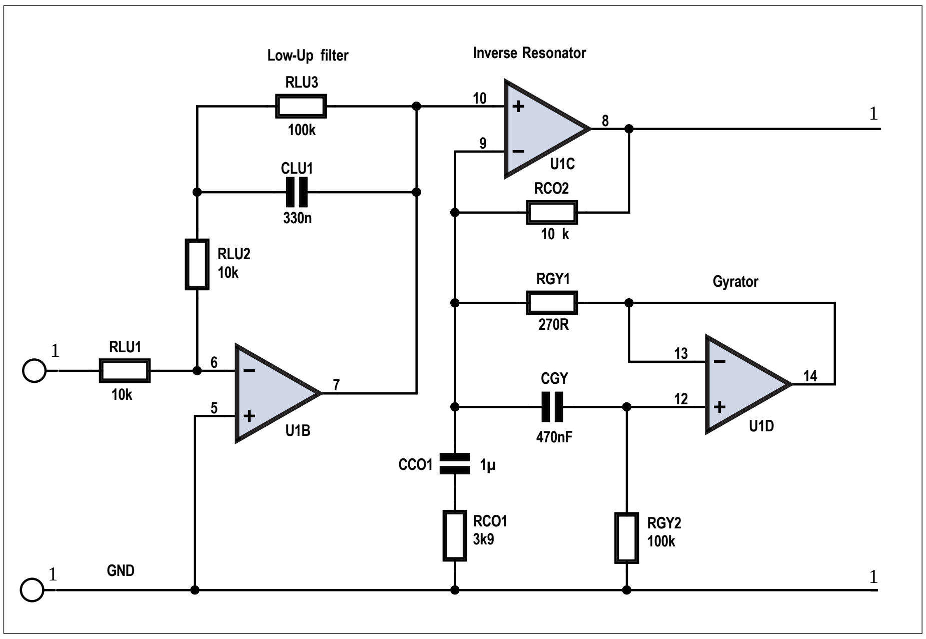


Figure 5: The schematic for the compensation filter. The component values are for the ScanSpeak loudspeaker that the author used; the reference designators of the components correspond with those in the BassCalc program.

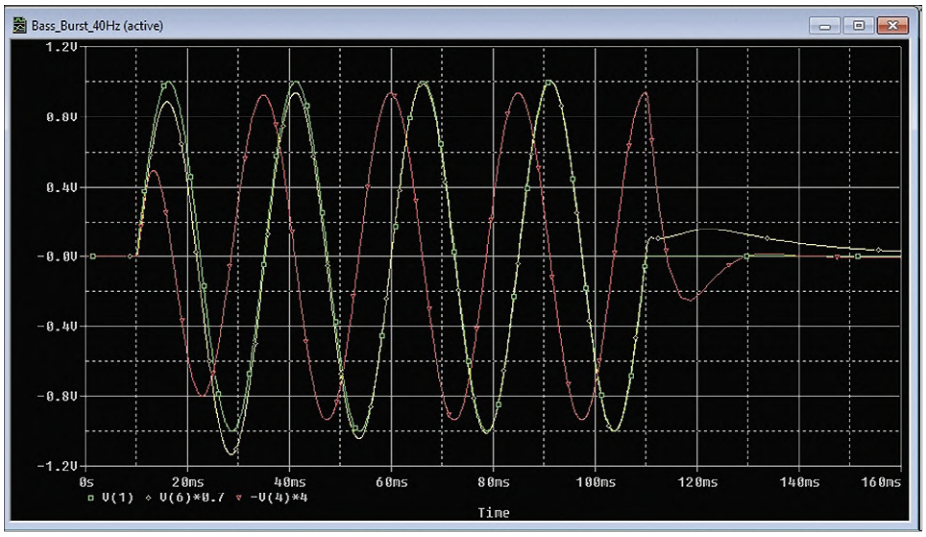


Figure 6: Spice simulation of the loudspeaker system with and without compensation filter. Green: input signal (a burst of 4 cycles of a 40-Hz sine); red: response without compensation; yellow: response with compensation. Further comment seems unnecessary...

ally 'tight' low-frequency reproduction. The schematic for the compensation filter is drawn in **Figure 5**. It again comprises two parts: a bass-boost amplifier for the low frequencies, and an 'anti-resonator' to compensate for the resonance of the loudspeaker in the box.

The bass-boost amplifier is built around U1B. RLU3 and CLU1 determine the frequency at which the boosting of the low frequencies begins; with the values shown here (for the ScanSpeak loudspeaker in a box with a volume of 41 litres) that is 4.8 Hz. RLU2 and CLU1 determine the frequency at which this additional amplification ends; here about 48.5 Hz.

This frequency-dependent amplifier is followed by a — well, lets call it an inverse-resonator, built around U1C. The gain of this opamp depends in the impedance of the inverting input to GND, comprising capacitor CCO1, resistor RCO1 and the gyrator around U1D. Such a gyrator is an extremely interesting circuit that could easily fill half the Elektor lab all by itself — but we will refrain from doing that here; instead we refer you to the extensive literature that can be found on the internet, among others [3].

A gyrator is, simply stated, a semiconductor circuit that behaves as an inductor, but then without the coil. For the equivalent self-inductance of the gyrator in the circuit of Figure 5 holds:

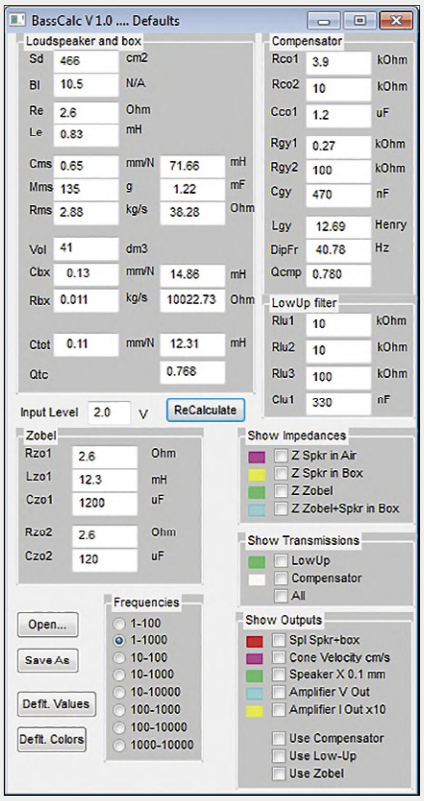
The BassCalc software

This program (**Figure 7** shows the user interface) was developed using Free Pascal and the Lazarus IDE. These are tools that can be freely downloaded for just about any platform (Windows, Linux, Mac, RPi, Android). The slogan is "Write Once, Compile Anywhere" and that is mostly true even in practice, although there are sometimes some platform dependent surprises. But whatever the case, the BassCalc program that was developed using this was tested with WinXP, Win7 and Linux Mint.

In the control window of **Figure 7**, the TS parameters are entered in the fields at top left (the default values in the program are the values that the author used for the ScanSpeak chassis that he used) and on the right the equivalent values of the electrical model, plus the resulting resonant frequency and the total Q value.

In the field on the right are the component values for the compensation filter and below that those for the bass-boost filter.

Then follow (below the button 'Recalculate') in the left field the values for the Zobel network, and below that


```
BassCalc V 1.0 .... Defaults

Loudspeaker and box                         Compensator
Sd    466     cm2                           Rco1   3.9     kOhm
Bl    10.5    N/A                           Rco2   10      kOhm
Re    2.6     Ohm                           Cco1   1.2     uF
Le    0.83    mH
                                            Rgy1   0.27    kOhm
Cms   0.65    mm/N   71.66     mH           Rgy2   100     kOhm
Mms   135     g      1.22      mF           Cgy    470     nF
Rms   2.88    kg/s   38.28     Ohm
                                            Lgy    12.69   Henry
Vol   41      dm3                           DipFr  40.78   Hz
Cbx   0.13    mm/N   14.86     mH           Qcmp   0.780
Rbx   0.011   kg/s   10022.73  Ohm
                                            LowUp filter
                                            Rlu1   10      kOhm
Ctot  0.11    mm/N   12.31     mH           Rlu2   10      kOhm
                                            Rlu3   100     kOhm
Qtc                  0.768                  Clu1   330     nF

Input Level  2.0  V     [ReCalculate]

Zobel                                Show Impedances
Rzo1   2.6    Ohm                    🟪 ☐ Z Spkr in Air
Lzo1   12.3   mH                     🟨 ☐ Z Spkr in Box
Czo1   1200   uF                     🟩 ☐ Z Zobel
                                     🟦 ☐ Z Zobel+Spkr in Box
Rzo2   2.6    Ohm
Czo2   120    uF                     Show Transmissions
                                     🟩 ☐ LowUp
                                     ⬜ ☐ Compensator
                                        ☐ All
                Frequencies
[Open...]       ○ 1-100              Show Outputs
                ● 1-1000             🟥 ☐ Spl Spkr+box
[Save As]       ○ 10-100             🟪 ☐ Cone Velocity cm/s
                ○ 10-1000            🟩 ☐ Speaker X 0.1 mm
[Deflt. Values] ○ 10-10000           🟦 ☐ Amplifier V Out
                ○ 100-1000           🟨 ☐ Amplifier I Out x10
[Deflt. Colors] ○ 100-10000
                ○ 1000-10000            ☐ Use Compensator
                                        ☐ Use Low-Up
                                        ☐ Use Zobel
```

Figure 7. The user interface for the BassCalc program. The default values are those for the ScanSpeak chassis used by the author.

we can indicate over which frequency range the calculations should be performed and displayed. In the fields on the right we can indicate which graphs we want to see (impedances, transfer functions, outputs), and finally we can indicate which corrections have to be 'brought along'.

With the coloured boxes other colours can be selected, if desired.

The buttons 'Open' and 'Save' offer the option of storing and retrieving a data set. The buttons 'Deflt. Values' and 'Deflt. Colors' finally restore the default values of the program. When a component value is changed, the button 'Recalculate' has to be clicked.

The input voltage that is applied to the loudspeaker is indicated at 'Input Level'. 2.38 V corresponds to 1 W into 8 Ω. For a 4-Ω version, such as the ScanSpeak of the author, requires 2 V for 1 W. Note: values are entered using a decimal point, not a decimal comma.

The program does not need to be installed: copy the executable file to a folder of your choice and run it.

Note: when 'silly' values are entered, the program will also return silly values – or crash.

L = CGY × RGY1 × RGY2 [values in farads, ohms and henries]

This equivalent self-inductance is in parallel of CCO1 and RCO1, so that effectively a parallel resonant circuit is formed, the resonant frequency of which is:

$f = 1 / (2 \times \pi \times \sqrt{(L \times C)})$

which with the component values from Figure 5 this comes to about 40 Hz.
If you would like to experiment with this circuit, then these are reasonable starting points: it is best if the resonant frequency and Q of the inverse resonator are equal to those of the loudspeaker model. For setting the *Q* you can change the value of RCO1., since it does not affect the resonance frequency, The corner frequency of the bass-boost filter is also preferably selected to be in the vicinity of the resonant frequency.
In the BassCalc program (see **sidebar**) you can play around with various values and parameters without the need to heat up your soldering iron; should you decide to build your own version of this circuit, it would be best if you used opamps with extremely small distortion, such as the OPA134.

The result

After all this theory, **Figure 6** shows the result of all our considerations: a Spice simulation of the woofer that the author has designed (see sidebar) and what formed the foundation for this article.
In the figure the input signal (a burst of four cycles of a 40-Hz sine) is shown in green. Red shows the response of the ScanSpeak loudspeaker in its 41-l box without compensation. It should be clear: the transducer clearly has trouble getting started, and afterwards it clearly lags the input signal.
Yellow shows the response of the system with compensation: the transducer follows the input signal nearly perfectly. And now it will also be clear what was meant with **time-corrected** woofer!

Of course, a simulation is nothing more than that: a simulation. Nevertheless the system built by the author sounds remarkably clear and detailed — the effort invested (theoretical and practical) appears to have been well worth it.

As already mentioned: this is not a ready-to-build construction project; the interested reader can use whatever they like from this and integrate it into their own system. It is also possible to contact the author through the Editor.

(180585-02)

An Unconventional Loudspeaker System

Figure 8 shows the loudspeaker system that is the result of the theoretical considerations in this article. It is a three-way system where each loudspeaker was given its own (class-D) amplifier. The lower five-sided enclosure houses the ScanSpeak woofer; in this configuration it reproduces frequencies to well below 20 Hz. The system as a whole was inspired by the 'Diamond' system from Hans van Maanen [7].
The auteur has made available an extensive (English-language) description of his loudspeaker system on the internet at [8]. If there is sufficient interest then we may return to this in a future issue of ElektorLabs magazine.

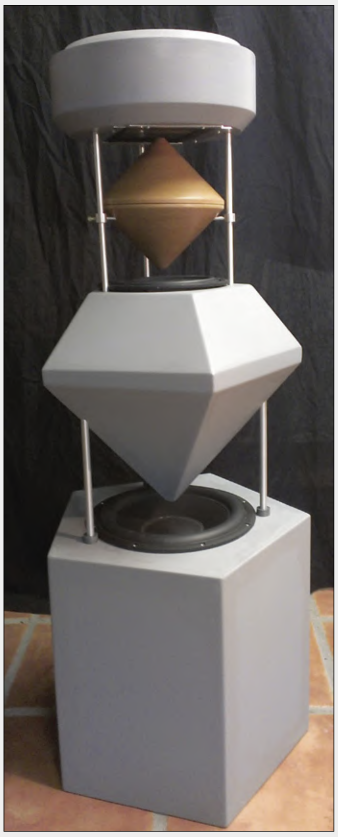


Figure 8: The three-way loudspeaker system as built by the author, based on the considerations in this article.

Links and Literature

[1] Web page for this article: www.elektormagazine.nl/180585-01
[2] Zobel network: https://en.wikipedia.org/wiki/Zobel_network
[3] Gyrator: https://en.wikipedia.org/wiki/Gyrator
[4] J. D'Appolito: Luidspreker-meettechniek (in Dutch), ISBN 90 5381 116 8)
[5] Vance Dickason: Luidsprekerkasten ontwerpen, (in Dutch), ISBN 90 5381 137 0)
[6] H.H. Klinger: Het luidspreker-bouwboek (in Dutch)
[7] Diamond loudspreaker system (in Dutch): www.temporalcoherence.nl
[8] An Unconventional Loudspeaker System (in Dutch): www.breem.nl/lsp/LoudspeakerSystem.htm

Qi Wireless Charging Pads

Adventures with smartphone inductive charging

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

Smartphones can do just about anything these days, except stray too far from a battery charging facility. For a journey of any distance, operation with a charging cable is more or less obligatory when the phone is used as a sat-nav in the car. The result is continual plugging and unplugging of the cable which the micro-USB socket on Android devices wasn't really designed for. Now, thanks to Qi-capability the newer smart devices can be charged inductively i.e. contactless. The author wanted this capability for his own iPhone X. Here is his report.

The continual plugging and unplugging of the charger cable on my smartphone was always an annoyance. 'Range anxiety' is not restricted to drivers of electrical vehicles; you just don't want to be caught out with a half-charged phone, topping up whenever you can is the order of the day. My PC has a 'Lightning' plug (for my iPhone) and a micro-USB cable (used by friends and for experimenting) permanently attached. Whenever I sit at the desk one of the first things I do is reach for the phone charger. In the Android world more and more of the top-end smart devices now come with Qi charging capability. I was a bit miffed that Apple who usually are right up there on the crest of the new technology wave, had somehow overlooked the benefits and convenience of wireless charging. That was all before September 2017 when Apple announced that the iPhone 8 series and the X would have Qi capability and that they were working on the AirPower charging mat capable of charging three devices simultaneously. At last; I couldn't wait to try out wireless charging.

Qi charging pads

Contactless battery charging using a resonant inductive loop has been around for a long time. Long before smartphones were equipped with the technology it was a common feature of rechargeable electric toothbrushes. Here it was introduced on the grounds of safety to reduce the risk of electric shock in a damp bathroom environment. The technology is not so complicated; the charger basically consists of a standard transformer with a large air gap in the iron core. In this case the insulated iron core couples with a receiver coil in the toothbrush body when it is fitted onto the charger (see **Figure 1**).

Many years later the Wireless Power Consortium developed the Qi (pronounced *chee*) open interface standard for inductive charging (see below: **The Qi standard**). In contrast to the basic toothbrush charger the Qi standard requires some electronic wizardry in order to ensure: a) the transmitter coil is not operating at full amplitude constantly, thereby reducing energy wastage. b) The charger can sense when a smart device has been placed over the charging coil and c) the charger can determine when the mobile device requires charging. To standardize the requirements but not make things overly complicated a basic communication link using a simple protocol is established between the charger and smart device via magnetic field modulation using the charge coil.

If you would like to delve a bit deeper into the subject check out the article published in the November & December 2016 edition of Elektor '**Wireless Power Converter**' [1], which describes how to build a contactless 50-watt energy transfer system. The article goes into details of the principles involved and the techniques used.

If you are just looking for an off-the-shelf Qi-compatible solution for contactless charging of your smart device there are so many charging pads available with many additional features and designs it's often difficult to see the wood for the trees. Prices range from around €10 up to ten times that amount.

My first Qi charging pad

Soon after my new iPhone X was delivered I got to thinking about which wireless charging pad I should buy. The surprising thing here is that Apple has chosen to adopt the industry-standard Qi wireless interface. This is a departure from Apple's usual practice of developing a proprietary interface and offering accessories at inflated prices that are only compatible with Apple products. This move is good news indeed; I am now free to choose a charging pad that best meets my needs and public Qi charging hotspots will work with Samsung as well as Apple products. As usual, whenever I need a new product I spend some time on the Internet comparing independent (hopefully) reviews of competing products, checking out customer feedback and user experience. I wanted, at first to go low cost and choose a practical design. I opted for the beautifully flat and really cheap C63 model from Pulesen priced at €12.85 which comes all the way from China (**Figure 2**). It's a 5-watt Qi-compliant charging pad that at least one user claims works well with an iPhone 8. This model and practically all other Qi pads are powered via a USB micro-B socket - so you still need an external power source. When the charging pad finally arrived (a micro USB cable was included), I connected it to a normal iPhone power adapter from Apple, which supplies 5.1 V at 1 A maximum. The charger flashed briefly and then went quiet. Nervously,

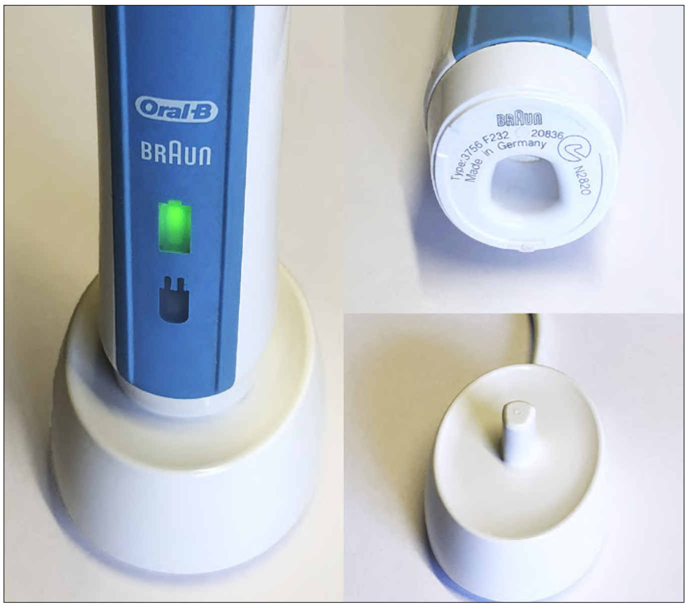


Figure 1. The forerunner to Qi: inductive toothbrush charging by Oral-B.

The Qi standard

The Wireless Power Consortium [2] published its 'Qi low-power specification' in 2009. Wireless charging pads compliant with this standard can contactlessly transfer up to 5 W of electrical energy for the purpose of battery charging. The medium-power variant of this standard released in 2011 allows a power rating up to 120 W which is beyond the requirements of smartphone charging. In 2015 the specification for the high-power variant of the standard was published, this takes care of energy transfer up to 1 kW for applications such as kitchen hobs.

It took a while before the standard was adopted by industry; in 2012 Nokia announced the Lumia 920 smartphone and Google chipped in with their Nexus 4 both featuring Qi wireless charging capability. Shortly after, car manufacturers such as Toyota and Ssangyong were offering Qi charging surfaces as options in their vehicles. Nowadays Qi-capable smartphones are nothing special, even Apple has eventually got on board with their iPhone 8 series and iPhone X in 2017.

Technically the Qi-system consists of a flat planar transmitter coil forming a tuned circuit in series with a capacitor fed by a power amplifier managed by an electronic controller which also handles the communication protocol between the charging pad and charged device. The magnetic field oscillates at a frequency in the range from 105 to 205 kHz (typically 140 kHz). Qi-compliant chargers designed for smartphones operate with a power rating between 5 and 10 watts.

Alternating magnetic fields easily penetrate non-metallic materials and is the reason Qi-capable devices use either plastic or glass material for the rear of the casing. Charging is possible even with a protective phone cover fitted (<1mm thick) although coupling losses increase as the distance between the transmitter and receiver coil gets larger.

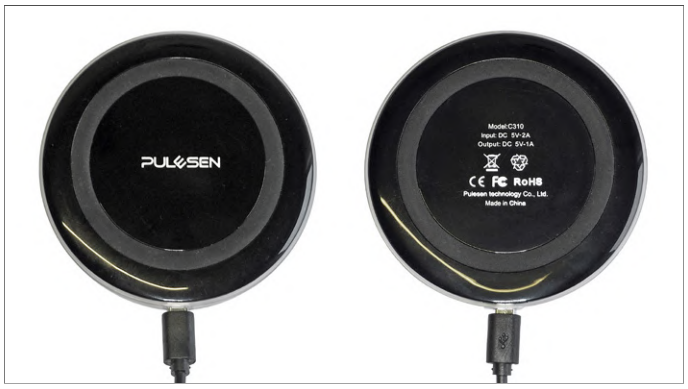


Figure 2. The C310 Qi charging pad from Pulesen can supply 5 W.

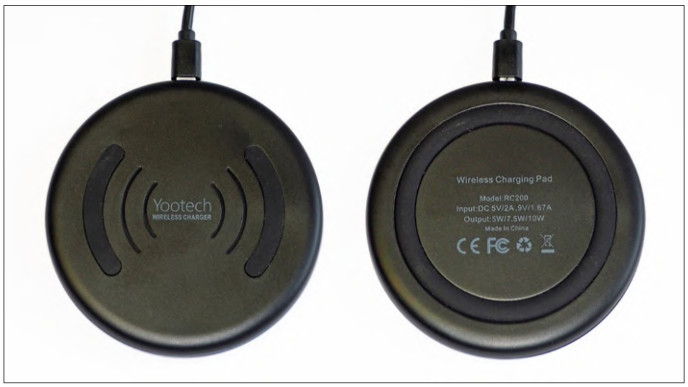


Figure 3. One smooth puck: The Yootech 10 W RC200 Qi charging pad.

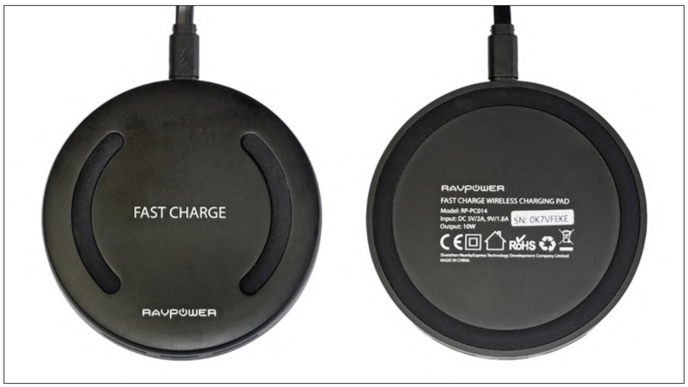


Figure 4. Distinctly chunkier: the 10-watt RP-PC014 from Ravpower.

I offered up my bright shiny iPhone and placed it gently onto the charging surface, I was a bit apprehensive; the cost of the phone is about one hundred times the price of the charger.
I need not have worried; the phone gave out a healthy ping indicating that charging had begun and the LED on the Qi pad flashed briefly. After about 30 minutes the partially discharged battery was full again. After the first test, I powered the pad from an external USB 3.0 hub with its own 5-V power supply, easily capable of delivering 5 A. That way if anything went wrong I would only need to replace a low-cost hub rather than something more expensive like a PC. In the following days no problems occurred, whenever I sat down at my desk I just placed the phone down on the charging surface.

More Power

As Voltaire reminded us 'The better is the enemy of the good'. I was quite happy with my charging solution before Apple released iOS 11.2. I was curious, the new OS supported faster charging rated at 7.5 W on both the iPhone 8 and X. I couldn't resist, after my normal research process I singled out the Yootech RC200 (**Figure 3**). According to the manufacturer this wireless charging pad is iPhone compatible and able to supply 5 W and 10 W as well as 7.5 W supported by Apple, priced at € 14.99 this wasn't going to break the bank.
The charging pad arrived and my USB hub had a spare port and power enough for both pads. This one worked fine as well but I couldn't help thinking it wasn't charging any faster than the 5-W model. A curious fault also developed with one of the chargers: after a few days use it just refused to charge. I assumed it had failed and switched to using the other pad that was also plugged into the hub, until it too stopped working after a couple of days. Unplugging and reconnecting one of the charging pads connected to the hub sometimes cured the problem… temporarily. Only when I disconnected both pads from the hub and reconnected them did the problem go away, for a few days…
It was time to find out what was going on here, after some internet research I had good reason to suspect that the controllers inside the Qi charging pads do not just control power to the coil but also connect to the USB data signals. With two chargers on the same hub it's possible there is a configuration conflict occurring. With just a single Qi charging pad on the hub or powered on its own from a USB supply the problem does not occur.
To be on the safe side I ordered another Qi pad. The RP-PC014 from Ravpower is another fast-charge 'Fast Qi wireless' charging pad priced at €10.92 (**Figure 4**). According to its documentation it can supply 10 watts for charging the Galaxy S8 and S8+ while Apple devices were guaranteed at least 5 W (it was a model from the previous year). This Qi charging pad also performed well — and better on its own.

Troubleshooting

Following on from my initial exploits with wireless charging I thought it was important to get a clearer picture of what was happening. First of all I needed to measure current consumption. To avoid breaking into USB cables I used a small diagnostic board which displays USB voltage level and current. It turned out that all three Qi pads were drawing roughly the same level of current in a range between 0.8 and 0.95 A. At the USB standard 5-V level this means that none of the chargers will ever be able to deliver 5 W. Taking into account an efficiency of around 80 to 90% for this charging method there will be no hope of charging a phone any faster wirelessly (including the 7.5-W charger) than directly with a USB cable.
I couldn't work out why all the Qi pads were drawing roughly the same current despite their differing technical specifications. The USB hub is able to supply up to 5 A and a separate power adapter is good for 2 A but despite this it seems that the charging pads just refuse to make use of the available power. It turned out there was a good reason for this behaviour; to implement faster charging the device must be charged from an

Quick Charge

Quick Charge is Qualcomm's proprietary 2013 standard for delivering higher performance mobile device charging via USB. The first version, QC 1.0, specified a charger output voltage of 5 V at 2 A maximum. Mains chargers or PC USB ports compliant with this spec will be able to charge smart devices at 5 W, a standard USB 2.0 port will only supply 2.5 watts.

The QC 2.0 standard introduced in 2015 increases the maximum power level up to 18 W. The higher level of current can produce significant voltage drops at the USB connector contacts and along relatively thin USB cables. In order to offset these effects the voltage at the interface has been increased. A QC 2.0 compliant USB charger can deliver 5, 9 or 12 V with a current rating of 3, 2 or 1.66 amps. The charge cycle will always begin at 5 V to ensure that a non-compatible device is not damaged. The voltage will only be increased following a successful communication exchange between the charger and the device under charge (smartphone, tablet or Qi charging pad etc.).
An even more subtle adaptation of the standard was introduced in 2016 QC 3.0 [3], in this version the voltage supplied can now range between 3.6 and 20 V in steps of 200 mV. 2017 saw the introduction of QC 4 und 4+ which specifies a maximum power of 27 W with even finer resolution of the supplied voltage.
Chargers conforming to the latest specification are backwards compatible with earlier versions. If you want to charge a smart device faster than the standard 5 W then choose at least a QC 2.0 compatible charger or AC adapter. USB mains power supplies conforming to the most recent spec however offer better value than previous models; the same could be said for smart devices generally. As usual Apple beats its own path; no Qualcomm chips are used to build iPhones. Some Qi-compliant chargers can work with Apple's 7.5-W charge mode.

'intelligent' **Quick Charge** (QC) compliant charger (see further on). There are already several revisions to this standard, a QC compliant charger is allowed to supply a higher output voltage than the 5 V you would expect from a standard USB charger. The iPhone is of course compatible with wireless Qi charging pads but unfortunately cannot be charged directly using a cable and Quick Charge-compliant charger. Some Android smartphones can accept the higher USB voltage but Apple does not use the technology. For wireless charging however if you want to charge at a level greater than 5 W it is just important to ensure that the Qi charging pad is compatible with a QC power supply (version 2.0 and above). My two Qi pads are both compliant with this standard. The printed input voltage rating can seen on the right view of Figures 3 and 4, they can accept up to 9 V (rather than the standard 5 V) to provide the extra charge capability. Running at 5 V the Qi pad cannot hope to get close to supplying the 5 W maximum.
All I needed now was a Quick-Charge 2.0-compliant AC powered charger. I went for the 18-W supply shown in **Figure 5**, which is actually compliant with the more recent Quick Charge 3.0 specification (as indicated in **Figure 6**). It has plenty of power in reserve and turned out to be a good choice.

Current

To test whether the Qi pads perform adequately with the new QC mains charger I could completely discharge the phone battery three times (maybe even more) and then measure how long it takes to reach full charge using the various Qi pads. Even though I know protection built into the battery management system will step in to ensure it is not allowed to go into a deep-discharge state, I am still reluctant to risk damaging the battery for the sake of the test. A simpler method would be to measure current taken by the different Qi pads to determine how much power they draw.
At first this seemed like a reasonable idea but then I realised my small test USB power meter would have been fried if I had inserted in between the QC 3.0 power charger and the Qi charging pad. It is only designed to operate at the USB standard

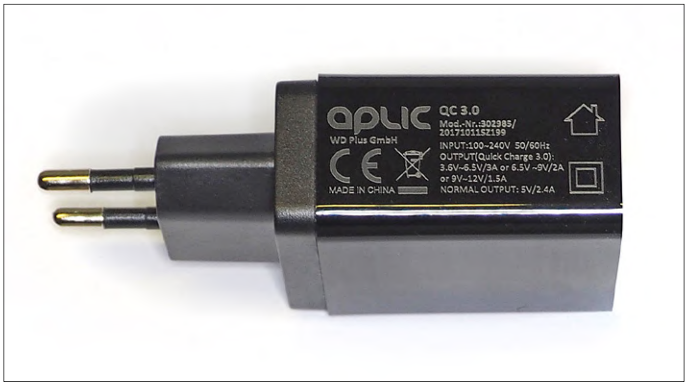


Figure 5. QC 3.0-compliant mains USB supply unit with printed technical spec.

Figure 6. It's chunky — you can't miss the 'Quick Charge 3.0' labelling.

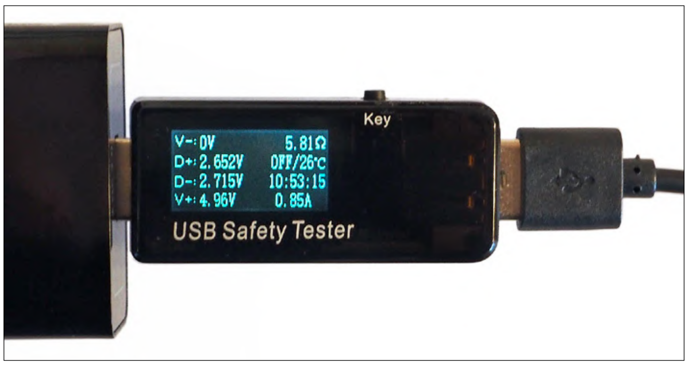


Figure 7. The 5-watt Qi charging pad from Pulesen draws 0.85 A at 5 V.

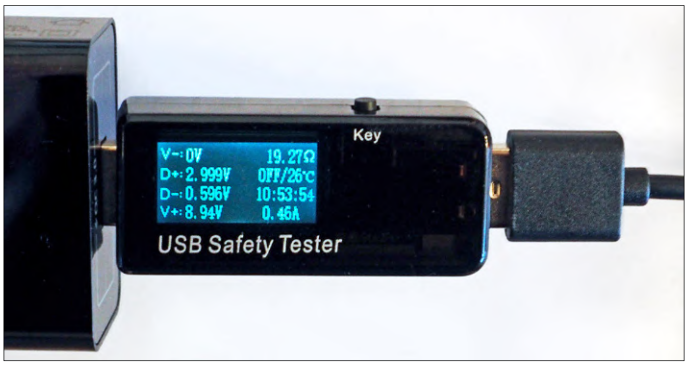


Figure 8. The Yootech Qi pad switches into 5-W mode taking 9 V.

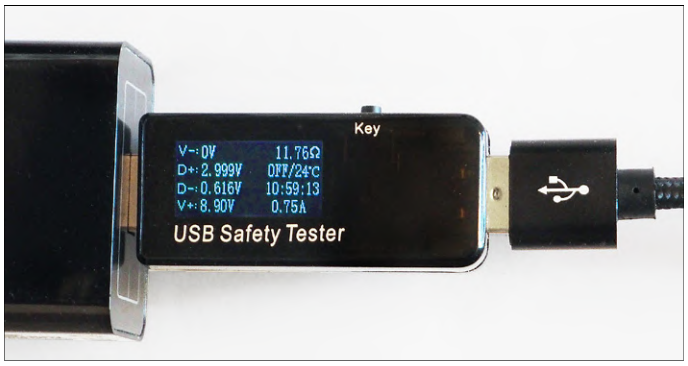


Figure 9. The Ravpower Qi pad draws around 7 W.

5 V and of course the QC power supply is capable of pushing out more than this. I needed a more versatile USB multimeter able to handle higher voltages. Some online research revealed a suitable candidate; the Muker J7 ‘USB MultiMeter’ priced at €9.99, according to the supplier it is compatible with both QC 2.0 and 3.0. In addition to voltage this little meter can measure the USB current, time in seconds, load resistance in ohms, power in watts, energy used in Wh and the capacity in mAh. **Figure 7** shows it in action fitted between the QC power charger and the 5-W pad from Pulesen. Here it shows the Qi pad powered at 4.96 V (measured at the output of the multimeter) and drawing 0.85 A. It looks as if an internal shunt is responsible for the small voltage drop; with no load the voltage rises to 5.03 V. It’s interesting that the 5-W pad only takes 4.22 W, if we assume the inductive charger has an efficiency of around 80% it works out that the iPhone is charged at 3.4 W. This figure corresponds to findings reported elsewhere on the net. The other interesting finding is that the Yootech pad often switches to 5 W charge mode even though it has 9 V input voltage. **Figure 8** shows that at 8.94 V only 0.46 A flows resulting in a charge of 4.11 W. There is no guarantee that all the devices in the charging loop understand each other properly. It seems as though you could well be charging at a rate less than 5 W even though the charger claims 7.5 or 10 W. The RAV-Power charging pad however proved more reliable **Figure 9** shows that at 8.9 V it takes at least 0.75 A making the input power draw 6.68 W. This charger gives you the chance to charge your phone faster than using a USB cable and a standard 5 W mains adapter.

Conclusion

The tests carried out here wirelessly charging my iPhone should also be relevant to Android smart devices. The top-end Samsung devices accept 10 W charging and the wireless charging pads tested here take between 8 and 9 W which provides faster charging than the standard USB mains charger supplied with the phone.
For me, slow charging speed is not a deal breaker; the big advantage of wireless charging is the convenience factor. Gone is the hassle of plugging and unplugging tangled cables and the nagging doubt about plug and socket wear and tear. Even the lowest power Qi charging pad is worth having and an improvement in terms of convenience and user friendliness. There are also some combi deals currently available offering both a QC mains power charger and wireless charging pad for just over €20, luxury items at a bargain price. ◀

(160662)

Web Links

[1] Wireless Power Converter, Elektor Magazine 6/2016: www.elektormagazine.com/160119

[2] Wireless Power Consortium: www.wirelesspowerconsortium.com

[3] Quick Charge 3.0 specification: www.qualcomm.com/products/quick-charge-3

Elektor Labs Pipeline

Here is yet another selection of interesting projects that can be found at the Elektor Labs website. Although some look bizarre at first blush, all have vast, sometimes unexpected application areas and/or user groups.

Get rich & famous, post your projects @ Elektor Labs

Add IoT to your mouse traps

New technologies often find applications in rather unexpected domains. Take for example LoRa, Sigfox and similar IoT technologies. While everybody is working on IoT temperature and humidity sensors for their homes, the real problem is to know if a rodent has been trapped somewhere deep down inside a large building or industrial site. The goal is, of course, to avoid sending out a maintenance engineer to check empty traps.

@ Elektor Labs: www.elektormagazine.com/labs/1761

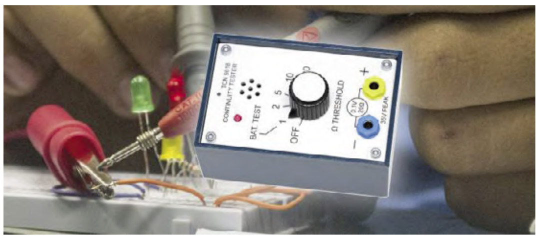
Build a wide input, wide output switched power supply preregulator

Bench-top power supplies with power transformers are heavy and expensive, especially when designed to supply a few hundred watts. Furthermore, such designs are old school. Today, using switch-mode configurations it is possible to design 250-watt PSUs running from anything from 85 up to 265 VAC line voltage, yet weigh less than 500 grams. The design showcased here is even open source.

@ Elektor Labs: www.elektormagazine.com/labs/1820

Build a robust continuity tester with adjustable threshold

Where many homegrown continuity testers are limited to a battery and a buzzer or maybe just an LED, this design uses a Wheatstone bridge instead. The advantage is that you can adjust the detection threshold from 1 Ω up to 20 Ω. But the design doesn't stop there, it even allows for lead resistance compensation and it is protected against people who forget to switch off the power of the DUT.

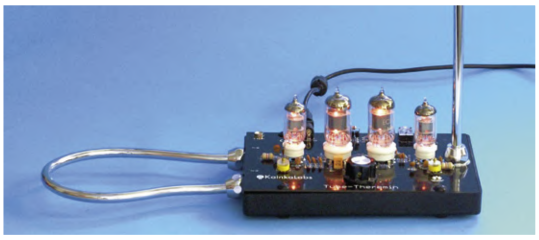


@ Elektor Labs: www.elektormagazine.com/labs/1823

Build a Theremin with tubes like Leon did a century ago

When Leon Theremin built his now oh so famous musical instrument, he only had tubes at his disposition. Today we can choose between analogue and digital technologies, and between BJTs, FETs, and tubes. We can even combine them all. This special 99th-anniversary version Theremin design only uses tubes and should please the purists among us. ⏮

180571-E-01



@ Elektor Labs: www.elektormagazine.com/labs/1828

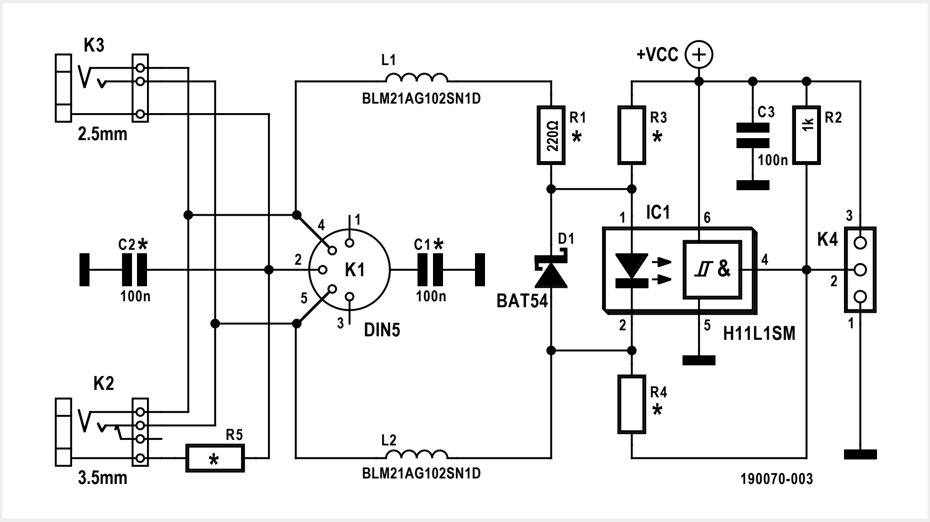
Err-lectronics

Corrections, Feedback and Updates to published articles

MIDI I/O Breakout Board

ElektorLabs 4/2019, p. 26 (190070)

CORRECTION. A mixup occurred in the DIN socket pin numbering on the circuit diagram. The correct pin assignment is: tip to DIN5 pin 5; ring to DIN5 pin 4; shield to DIN5 pin 2.
The manufacturer's part number of IC1 should read: H11L1SM.
See the corrected circuit diagram.
The Breakout-Board PCB is correct and does not require any changes.



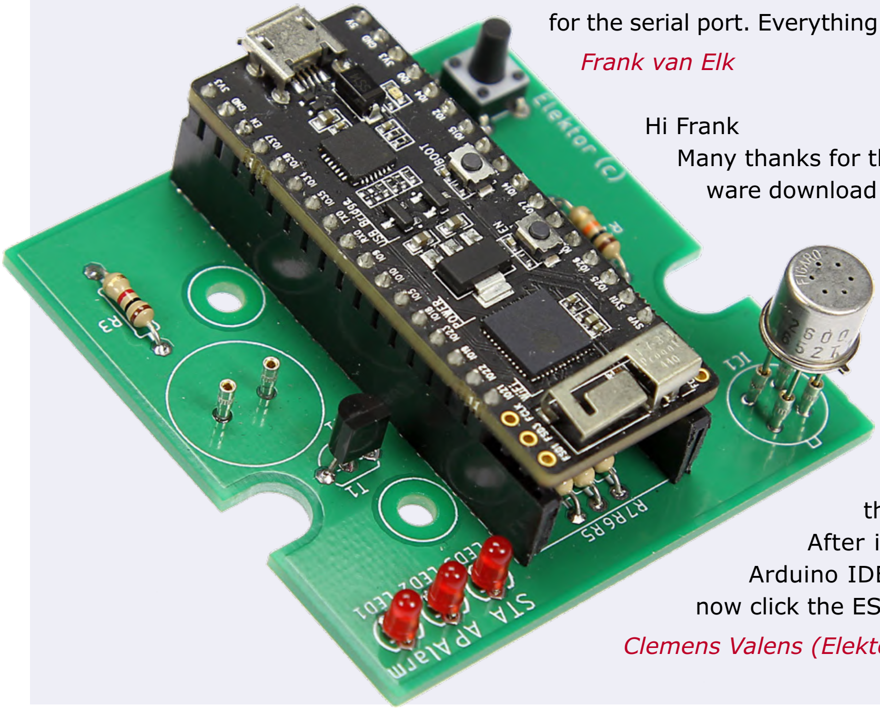
Air Pollution Monitor

ElektorLabs 2/2019, p. 90 (170182)

FEEDBACK. Download 170182-11.zip (*www.elektormagazine.com/170182-03*) contains various sketches, a data folder and a PDF document. Unfortunately, I am not too sure how to get them, can you help me out here please?

I have installed Arduino 1.8.9 with the ESP32 libraries as well as the drivers for the serial port. Everything is prepared for the *ESP32 Sketch Data Upload*.

Frank van Elk



Hi Frank

Many thanks for the feedback. We have now organised the software download procedure more clearly. It is now available for download on the project page.

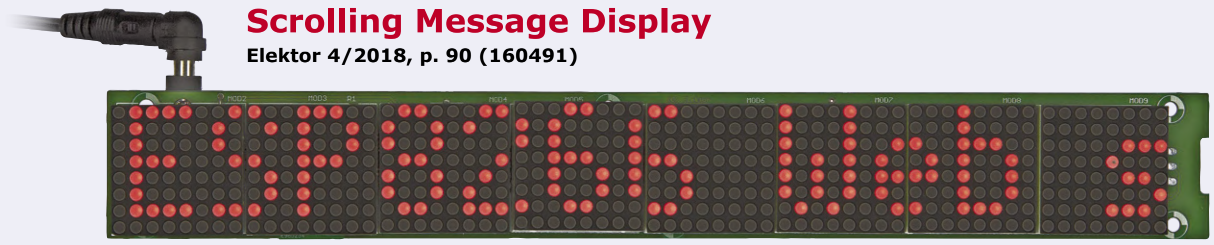
Extract the folder in the Sketch folder of the Arduino IDE (in Windows 10 it will normally be organised in *[...]\Documents\Arduino\sketchbook*). After that, the folder should look like this: *[...]\Documents\Arduino\sketchbook\tgs2600\*.The Arduino IDE needs to be restarted, otherwise it will not find the sketch. Now you can open the sketch and load it into the ESP32.

After installing the SPIFFS uploader, rebooting the Arduino IDE, and opening the TGS2600 sketch, you can now click the ESP32 Sketch *Data Upload* option.

Clemens Valens (Elektor Labs)

Scrolling Message Display

Elektor 4/2018, p. 90 (160491)



FEEDBACK. I have posted an update to the 'Scrolling Message Display' project featured in *www.elektormagazine.com/labs/scrolling-text-display-160491*. An ESP32 controller module has been now incorporated into the design which significantly broadens the capabilities of the original project.

Peter Tschulik

ESP32 Arduino Library 1.0.2

UPDATE. Anyone working with the latest version of the ESP32 Arduino board software (version 1.0.2 in the board manager) may stumble across an annoying bug. Should the code refer to AsyncUDP, an error may occur in the package buffers and the chip will throw an exception, which results in a reboot.

Libraries using this function include the *NtpClientLib* from *https://github.com/gmag11/ntpClient*. If you encounter this problem just switch to version 1.0.1 or version 1.0.3.-rc1 in the board manager options.

Mathias Claußen (Elektor Labs)

DIY Temperature Controlled Solder Station

ElektorLabs 1/2019, p. 14 (180348)

FEEDBACK. The new edition of the soldering station introduced in ElektorLabs issue 1/2019 is a great design. I already built the Platino-based station which is its predecessor and have been using it successfully ever since. Its successor is much smaller and more manageable and the additional features are a real bonus. Not everything described in the article is correct however: The output of the setpoint and the measured temperature values as well as the complete status of the soldering station via the USB interface is not implemented in the firmware. A quick check of the Arduino sketch revealed that the serial interface initialisation process is in fact missing and also that the values are not sent to the interface. Is a firmware upgrade planned to take care of these functions?

Hans Schneider

Hi Hans
Your comments are much appreciated. For all kits for this project delivered after the 15 March 2019 the update for the serial interface is available (from firmware version 1.2).
When you connect the soldering station to your computer, you can access the serial port using the terminal from the Arduino IDE or another serial terminal program. If you are requested to set a baud rate, choose 115200. Communication with the device is via USB so the baud rate setting in the device is irrelevant. NB: Do not select 1200 baud as this may trigger the bootloader.
These are the commands supported by the console:
set/get set/get setpoint[xxx]: this will set or recall the setpoint;
*get temperature: t*his will get the current temperature;;
clear error: if an error is displayed, this deletes it;
help: Help is available here.
Finally, a general note: At Github (*https://github.com/ElektorLabs/)* you will find the firmware log and software versions for various Elektor lab projects. It's worthwhile checking the files available here if you have built an Elektor Labs project. In the case of the soldering station, the web link is *https://github.com/ElektorLabs/180348-DIY-Soldering-Station*.

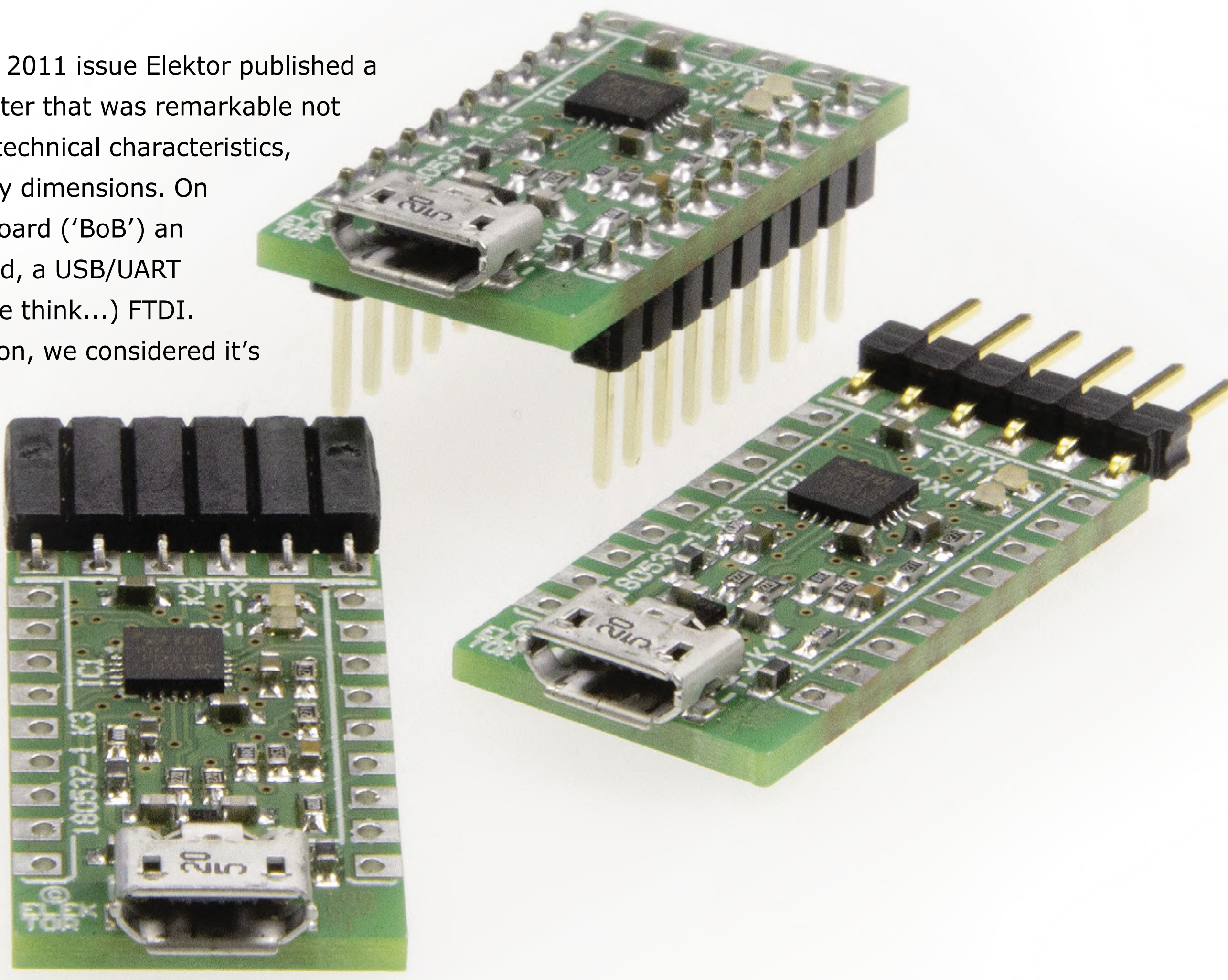
Mathias Claußen (Elektor Labs)

USB/RS232 Breakout Board

a.k.a. BoB

By **Eric Bogers** (text) and **Ton Giesberts** (Elektor Labs)

In the September 2011 issue Elektor published a USB/serial converter that was remarkable not only for its great technical characteristics, but also for its tiny dimensions. On thats break-out-board ('BoB') an FT232RQ was used, a USB/UART chip from (what'ye think...) FTDI. Now, eight years on, we considered it's time for an update, which we present to you here.



Quick Specifications

- micro-USB-connector
- fully USB 2.0 compatible
- VCCIO +1.8 to +3.3 V (4 V absolute max.)
- regulated 3.3-V output (50 mA max.)
- data rate 300 baud to 3 Mbaud
- compatible with RS232, RS485 and RS422
- full handshake USB-to-UART
- 4 configurable I/O pins

A few changes have been made though compared to the earlier version [1]. In the first instance we have used a cheaper version of the USB/UART converter: the FT231XQ. This version also allows normal pinheaders to be used, because they are a little closer together. This has the consequence that the new BoB with the connectors attached, is a little smaller than the old version; the circuit board is nevertheless a little larger (29.2 mm instead of 27 mm). With the micro-USB connector fitted and using a right-angle header at the other end (the RS232 connection) we arrive at a total length of just under 4 cm. That's right, we are now using a micro-USB connector instead of

the 'mini' version, since the first has now become much more common. Furthermore, we changed the order of the signals on the RS232 connector and added the RTS signal to it. This is for compatibility with the ready-made USB/RS232 cables from FTDI. And finally we added ESD protection in the shape of a special suppressor.
Anyone who is interested in what goes on inside the small package of the FTDI chip can cast an eye over the datasheet [2] and the accompanying technical note TN140 with errata [3].

The schematic

High time to take a look at the schematic of **Figure 1**. This may appear a little intimidating, but turns out to be much simpler after a closer look. The heart is, of course, formed by IC1, the actual converter IC. The connections from this IC are brought out to a set of four connectors. K1 is the micro-USB connector and K2 is the RS232 connector. The remaining connections from the IC go to the other two headers, where K3 gives access to all the UART signals and the four configurable I/O lines can be found on K4 (as well as the power supply rails). The two LEDs indicate activity on the data lines. These LEDs are connected to CBUS1 and CBUS2. By default these I/O lines are configured to indicate transmission (LED1/TX) and reception (LED2/RX) via the USB connection. It is, however, also possible to configure the chip so that these LEDs indicate the transmission and reception of RS232 data. (As an aside: on the small circuit board there was not enough space next to LED D1 for the text 'TX', this is therefore a little above and to the left of the LED. This text has therefore nothing to do with connector K2!)
A final component that is worth mentioning, is D1: an ESD suppressor that protects the circuit against electrostatic discharges. The selected type excels for its extremely small capacitance.
There are two 0-Ω resistors (R1 and R8) on the circuit board. These are related to the power supply, and we will look at that now.

Power supply

A few things can be said about this, especially because mistakes can turn out to be fatal for little BoBby.
In the first instance the chip can be powered via the USB connection (pin 1). Our small BoB then obtains +5 V from the device where the break-out-board, via USB, is connected to. In this case R1 has to be fitted. It is

PROJECT DECODER

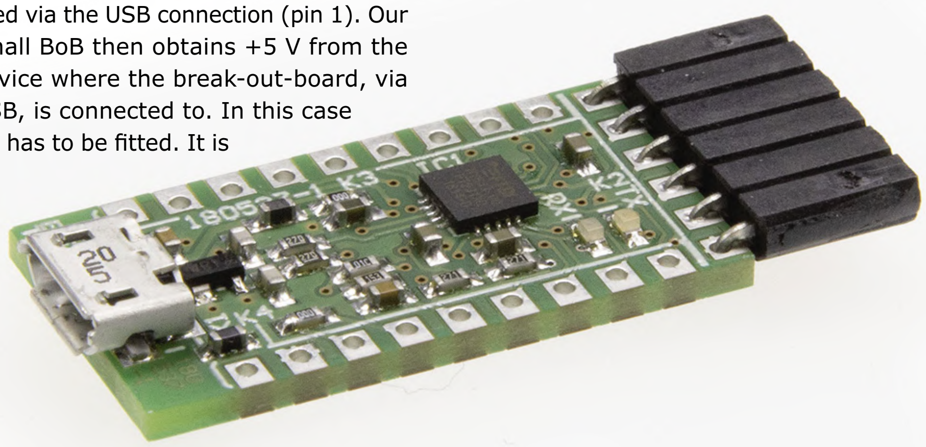
USB
RS232
FT231

entry level
intermediate level
➡ expert level

2 hours approx.

SMD tools,
hot-air soldering station

€25 / £ 22 / $30 approx.



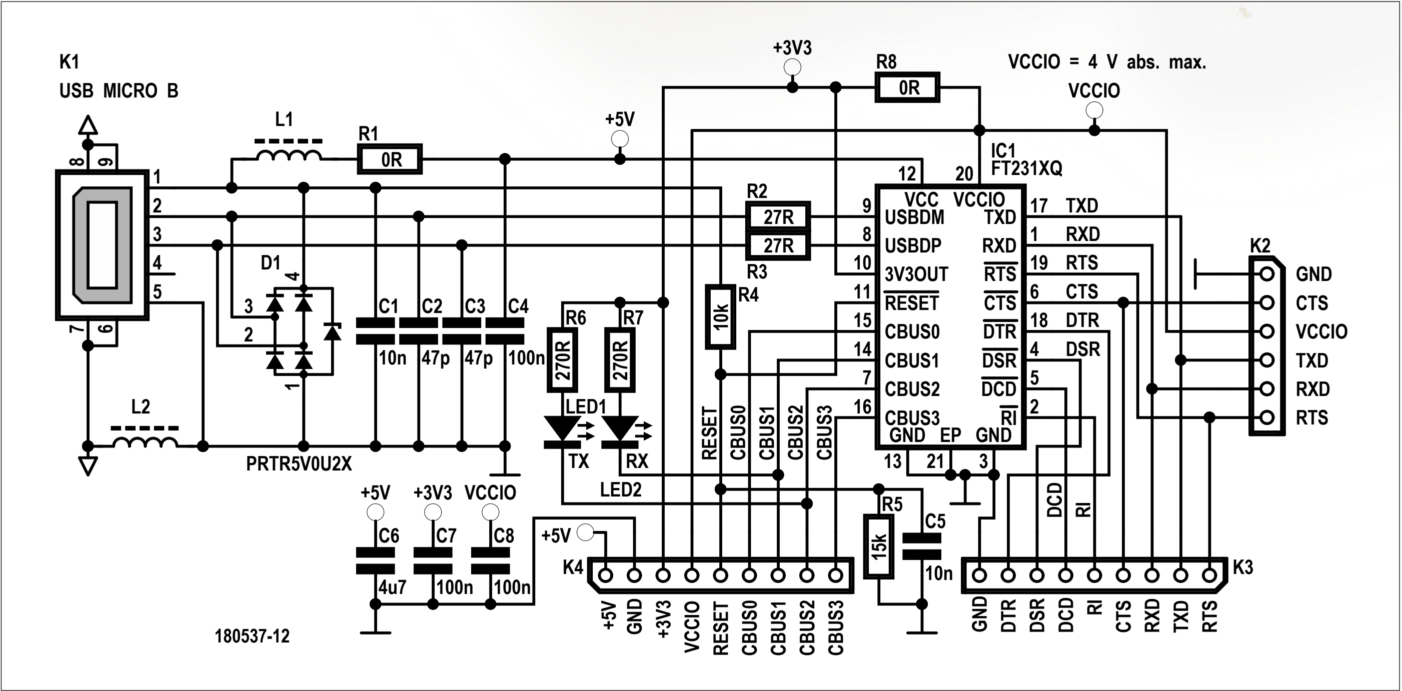


Figure 1: The schematic for the break-out-board. It is much less complicated than it appears at first glance.

self-evident that in this case no external power supply of 5 V may be connected to pin 1 of K4.
Of course, the reverse is also possible, powering the circuit from a separate 5-V power supply voltage; this is then connected to pin 1 of K4. Naturally, R1 has to be removed in this case. In this configuration the circuit draws practically no current from the connected device — a niggling 0.2 mA via voltage divider R4/R5; this serves the purpose of detecting whether there is a (switched-on) device connected to K1 or not. But please take care: pin 1 of K4 is directly connected to the chip!
Then we arrive at VCCIO: this is the power supply for the I/O cells in the chip. Here too we have two possibilities. When 0-Ω resistor R8 is fitted, the VCCIO pin (pin 20) is connected to the internal 3.3-V regulator in the chip (pin 10). This can supply a maximum of 50 mA; this voltage is available externally at pin 3 of K4. When R8 is not fitted, VCCIO is powered via pin 4 of K4. Pay very close attention to this: VCCIO is specified for a voltage range of +1.8 to +3.3 V with an absolute maximum of +4 V. Higher voltages will destroy the chip!
As an aside: the RS232 connections are 5-V resistant; external UART logic that operates at 5 V can be connected without any problems.

DIY assembly requires a steady hand!

Configuration

As we already mentioned in passing earlier, it is possible to configure the four I/O pins from IC1 (CBUS1 through CBUS4), that are available externally on K4, to your own requirements. The manufacturer of the chip makes the free tool, called FTPROG [4], available for this purpose. From the same location you can also download an extensive manual for this, so that we will not go into this further here, more so because the circuit in its default configuration does exactly what it needs to do. A word of warning: take care when experimenting with this software tool: if you mess around too much it is possible that the chip becomes completely non responsive...

Assembly

We have designed a double-sided)circuit board for this circuit that you can order from our Store — see **Figure 2**. What will be immediately obvious is the large number of vias — but not all that surprising on such a small circuit board with four connectors.
If finicky work with SMD parts does not appeal to you, then you can buy a partially assembled board from our shop — here all the ‘difficult’ parts are already mounted. The inveterate DIY-er requires, in addition to suitable SMD tools (a hot-air soldering station, for example), mainly a steady hand. To clearly see which components have to go where (for space reasons we have had to put the part designators within the outline of the components) it is recommended to use an enlarged version of the component overlay. You can download this from the project page for this article [5]. Incidentally, this also clearly indicates which signals are on which pins of the various connectors.
Take care when mounting the micro-USB connector K1 and ESD suppressor diode

COMPONENT LIST

Resistors
Default: SMD 0603
R1,R8 = 0Ω, 75V/100mW
R2,R3 = 27Ω, 75V/100mW
R4 = 10kΩ, 50V/100mW
R5 = 15kΩ, 50V/100mW
R6,R7 = 270Ω, 50V/100 mW

Capacitors
Default: SMD 0603
C1,C5 = 10nF, 50V, 20%, X7R
C2,C3 = 47pF, 50V, 2%, C0G/NP0
C4,C7,C8 = 100nF, 50V, 10 %, X7R
C6 = 4.7µF, 6.3V, 10%, X5R

Inductors
L1,L2 = 330Ω @ 100MHz, 1.7A, 0.08Ω
(Murata BLM18KG331SN1D)

Semiconductors
D1 = PRTR5V0U2X, ESD suppressor,
SMD SOT-143B
LED1 (TX) = HSMG-C190, green, SMD 0603
LED2 (RX) = HSMS-C190, red, SMD 0603
IC1 = FT231XQ, SMD QFN-20

Miscellaneous
K1 = micro-USB type B, female, SMD
K2 = 6-pin pinheader, 0.1” pitch, right angled
K3,K4 = 9-pin pinheader, 0.1” pitch, straight
PCB no. 180537-1 v1.0, from the Elektor Store

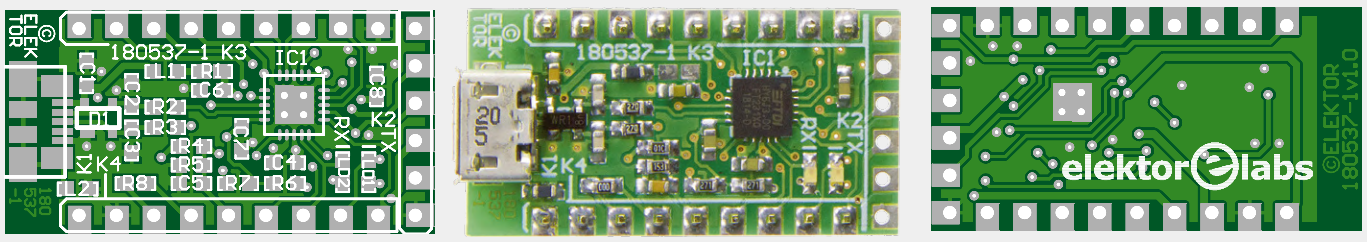


Figure 2: If the finicky work with SMD parts does not appeal, then a partially assembled circuit board is available in the Elektor Store.

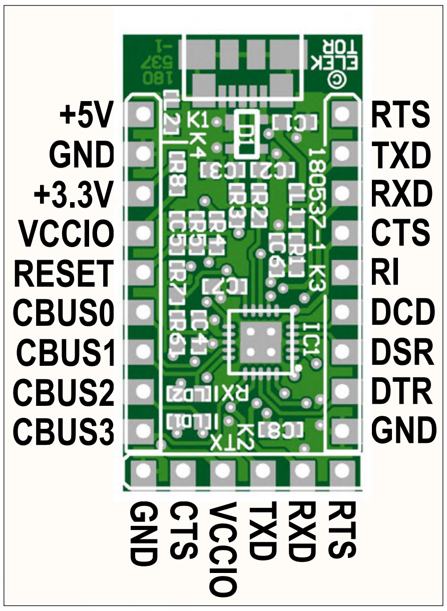
D1: solder K1 first and then D1. Reason: when D1 is already soldered in place, it is practically impossible to fix any potential short circuits between the pins of K1.

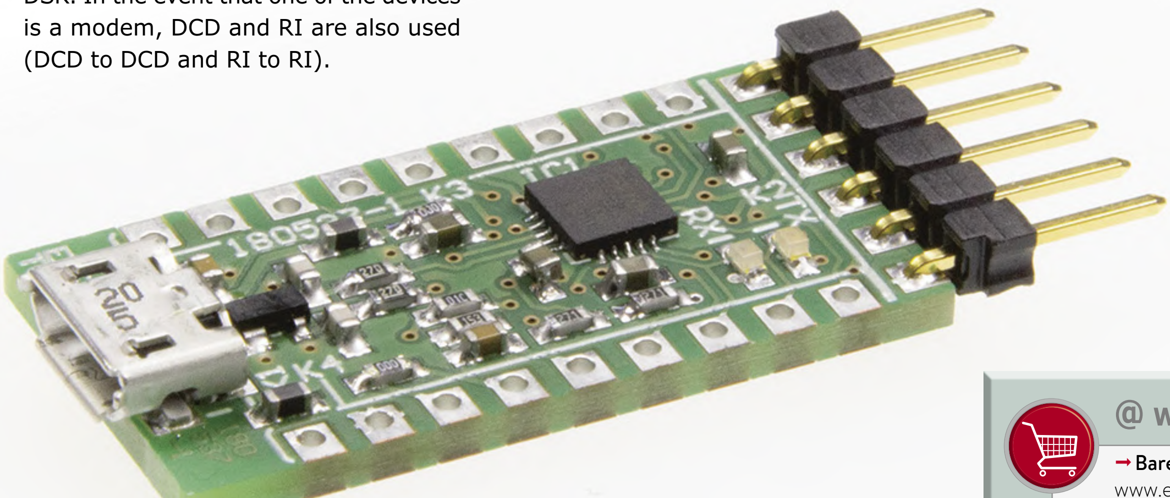
Conclusion

The simplest RS232 connection between two devices is the null-modem connection (also sometimes called the crossover connection). Here the TxD of one device is connected to the RxD of the other (and *vice versa*, and of course the Ground lines of both devices are connected together). Besides the 5 to 8 data bits, communications uses a start bit, up to two stop bits and optionally a parity bit. A safer method for transferring data is to use a buffer that stores the data temporarily. Handshaking (hardware or software wise) prevents 'overflow' of this buffer. For hardware handshaking the control signals DTR, DSR, RTS and CTS are used. CTS is cross-connected with RTS, while DTR is cross-connected with DSR. In the event that one of the devices is a modem, DCD and RI are also used (DCD to DCD and RI to RI).

With software handshaking, two special characters are used for start and stop: XON and XOFF (decimal 17 and 19 respectively). This is called software flow control. This method has a disadvantage however: when transferring binary data these characters are not allowed to occur, otherwise it will go wrong.
More information about RS232 can be found at [6] and [7], among other places for sure.
After you have finished assembling your BoB (and after a very careful check of the soldering work!), you can test it with the aid of a terminal program, such as HTerm. The simplest method is of course to use two computers and two BoBs. With full handshaking the connection can be verified in HTerm: green indicator circles indicate whether DTR and/or RTS are active at the other end. ⏮

(180537-B-02)





@ WWW.ELEKTOR.COM

→ Bare circuit board
www.elektor.com/18878

→ Pre-assembled circuit board (you need to mount the supplied headers)
www.elektor.com/18895

Web Links

[1] USB-FT232R breakout board: www.elektormagazine.nl/magazine/elektor-201109/16422

[2] FT232XQ Datasheet: www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT231X.pdf

[3] TN140: www.ftdichip.com/Support/Documents/TechnicalNotes/TN_140_FT231X%20Errata%20Technical%20Note.pdf

[4] FTPROG tool: www.ftdichip.com/Support/Utilities.htm#FT_PROG

[5] Project support page: www.elektormagazine.com/180537-02

[6] More about RS232: www.commfront.com/pages/3-easy-steps-to-understand-and-control-your-rs232-devices

[7] Even more about RS232: www.codrey.com/embedded-systems/rs232-serial-communication/

[8] FTDI driver: www.ftdichip.com/Drivers/VCP.htm

[9] FTDI driver: www.ftdichip.com/Drivers/D2XX.htm

HomeLab Helicopter

Wondrous Things Electronic Spotted From Above

By **Clemens Valens** (Elektor Labs)

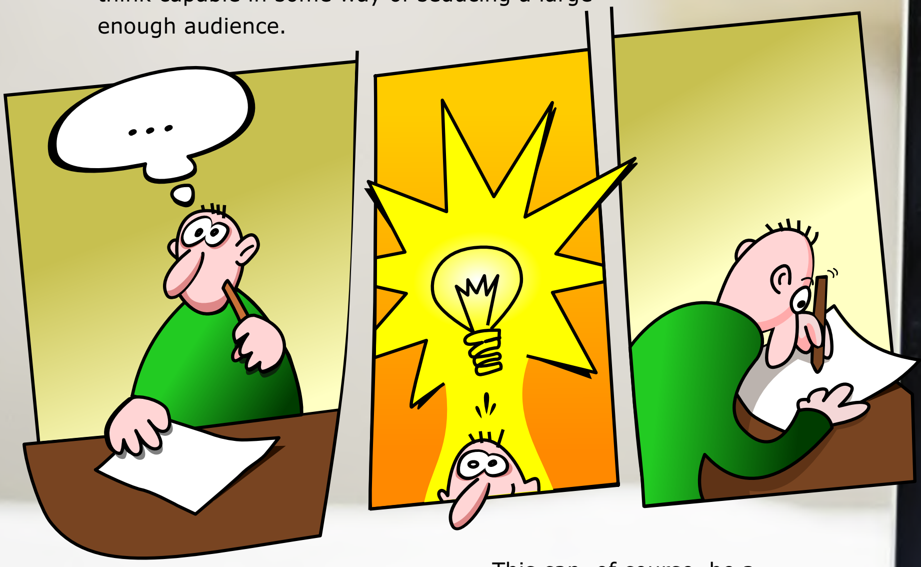
FROM IDEA TO PRODUCT PART 1

Sometimes people ask us: *How do you create a "product"*? Before we can dive into this interesting but complicated subject, another question should be answered first: *Why* develop a product? There is no point in developing a product just for the sake of it. Without a realistic objective, simply whipping up a prototype is usually enough. A product's life really begins when it is ready to be sold. This implies that developing a product also involves developing one or more sales channels for it, without forgetting details like packaging, shipping, invoicing, marketing and aftersales. A sales channel can be anything from selling out of the trunk of your car, eBay or your own online web shop to getting it on the shelves of a nationwide supermarket chain, and upwards of that to Amazon and/or Alibaba. But, before a product can be sold, it must be produced, tested and certified. A user manual should be written too. Production is only possible after the creation of a fully working and validated prototype that was designed with production in mind; i.e. it must be industrialized.

As you can see, creating a product is much more than just having a good idea.

FIND A GOOD IDEA

You need to come up with an idea for a product (or service, but we will restrict ourselves here to tangible products) that you think capable in some way of seducing a large enough audience.



This can, of course, be a completely new product, but it can also be something you already developed for some other reason. Indeed, many projects done for some particular purpose can easily be adapted to fit a much wider or different field of application, and therefore make for good product candidates.

NEW PR
Research
A careful study or a systematic study in order to establish facts or to discover new information.
Analysis
A process to examine something in detail in detail in order to explon it as a basis of discussion or interpretation.

WRITE A BUSINESS PLAN

Once you have an interesting product idea, with engineer's blood running through your veins you will be tempted to start developing the product right away. But hold your horses. Start by investigating the market in an objective way. You may think it's a great idea, but what do other people think of it? How many others have had the same idea? Does the product exist or not? If not, is that because your idea is truly original or because all the other people with the same idea gave up on it when they discovered that there was no market for it? If it does exist, is the market large enough for competitors? And what will you need to develop your product? How much time, money, tools, people, and other resources? How are you going to market your product? In short, what you should do first is write a business plan. That's is not an easy exercise. Do not make the mistake of assuming that everybody views the world like you do. Also, try to develop a plan covering several years. Where would you like to be in, say, five years' time? And how would you get there? To help you get started with your business plan, search the Internet for templates and examples; there are many online resources for this. You can also get help from your local Chamber of Commerce.

DO YOU REALLY WANT TO BE A CEO?

Now suppose that you managed to produce a business plan that holds up to scrutiny. The next step then is to execute it. Quit your daytime job, invest all your savings, maybe prepare for a divorce; are you willing to do that? Or is it enough to spend your free evenings and most weekends on the project? Also keep in mind that when your business becomes viable, your engineering job may very well turn into managing people, prospecting clients, looking for money and other 'things' businessmen do; as the engineering part you liked so much is to be left to other people. Is that what you want? If so, do you think you will be any good at it? Maybe you need one or more partners to help you. Asking friends to become partners is a good way to kill friendships, so you may want to look elsewhere. Avoid 50-50 partnerships at all cost; best is to keep a majority part or you may get kicked out of your own business.

RODUCT
Ideas
Any thoughts, opinions, creation, suggestions or conception that is existing in the mind as to a possible cause of action.
Function
An action or acticty proper to a person or thing the purpose for something which is designed for.

WHAT'S NEXT?

If, after reading this article, you are still motivated enough to develop a product, then get ready for the next instalments to learn how to turn an idea into a real-life, i.e. producible product. We will discuss electronics products, because that is what Elektor is all about, but parts of it will also be valid for non-electronics products. If, on the other hand, you feel that being an entrepreneur is nothing for you, you can fire up the soldering iron and start working on your idea right away. You never know what may come out of it.

Alias is mounted on top of the smart assistant.
Photo courtesy of Bjørn Karmann

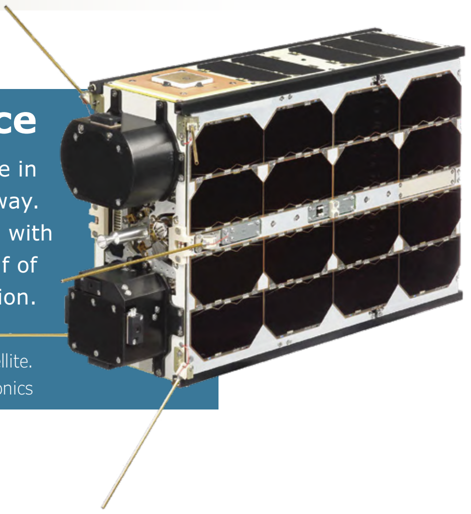
Outsmart Your Smart Assistant

Personal or smart assistants like Google Home and Amazon Alexa continuously listen to the sounds around them in order to pick up trigger words and commands. Since these devices are connected to the Internet, they present huge privacy issues. If this is a concern to you, you should not install them in your home and 'sadly' you cannot profit from the services they provide. Wouldn't it be nice if you could render these assistants less intrusive? Inspired by *cordyceps fungus* and viruses that take control of their hosts, this is where Project Alias by Danish designer Bjørn Karmann comes in. Simply put, Alias is a device that replaces the smart assistants' ears by less nosy ones. It does so by deafening the assistant's microphone with a constant interfering noise. Alias itself only listens for its own trigger words; it is not connected to the Internet. When it detects a valid trigger, it switches off the noise and passes the command on to the smart assistant that will then execute it normally. Alias can be programmed to send all sorts of speech commands to the smart assistant, enabling many new possibilities. Source code and building instructions can be found at GitHub.

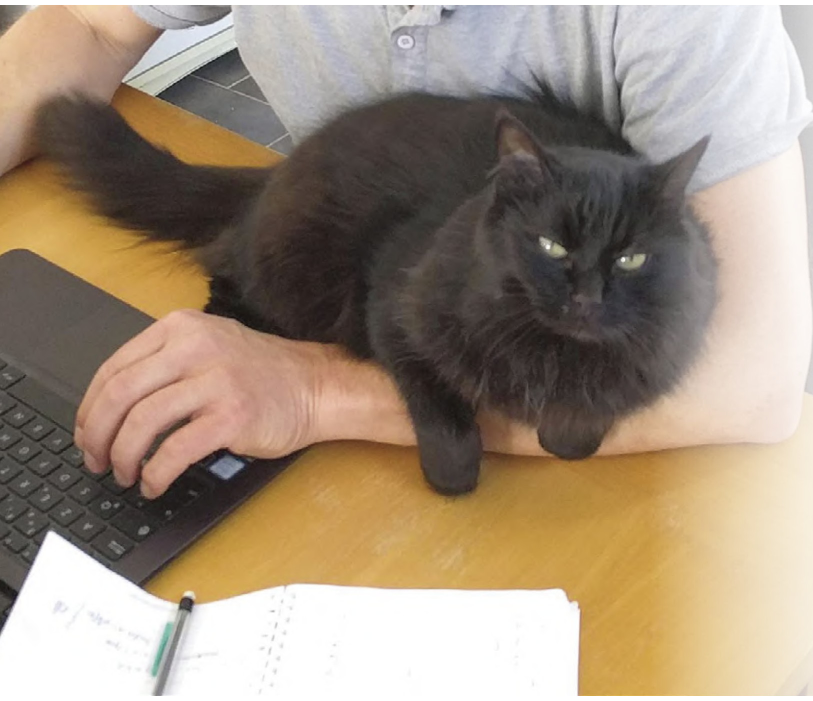
http://bjoernkarmann.dk/project_alias

LoRa extends into space

After launching a satellite back n April 2019 from the Satish Dhawan Space Centre in India, Lacuna Space has completed first phase testing of its space-based LoRa gateway. Test systems deployed around the world have been able to communicate effectively with the LoRa Space Gateway. A further three satellites are set for launch in the second half of 2019 to complete the initial demonstration constellation.


The LoRa-based Space Gateway was hosted on a 6U cubesat satellite.
Photo courtesy of Nano Avionics

Must-have Homelab Tool



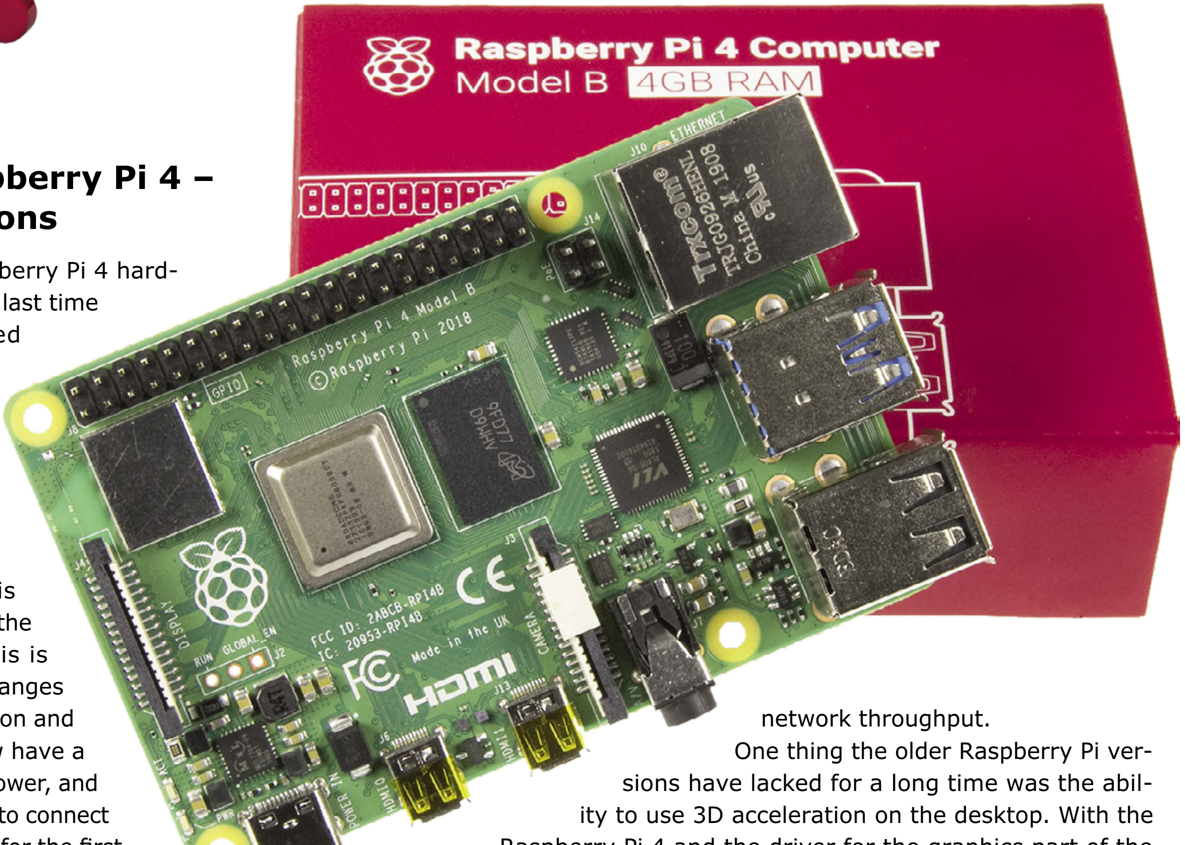
Cats can relieve stress and anxiety in humans; they help to lower blood pressure and reduce the heart rate, which in turn diminishes the risk of heart disease. Scientific research supports this. A study from 2008 even found that people without cats have 30% more chance of dying from a heart attack than cat owners. In an attempt to reduce employee stress, and to increase productivity, a Japanese IT company has adopted cats, and the employees are encouraged to take care of them. They even get financial help if they adopt a cat at home too. Another interesting example is this office building in Las Cruces, New Mexico, where they have set up a library-style system where people can sign out kittens to relax with. I have had cats all my life and I can only confirm their de-stressing qualities; therefore, I highly recommend them for any lab or office. The only thing to watch out for is cat pee. Not only does it smell terrible, it is also highly corrosive and will render circuit boards completely unusable in no time. Of course, for people who really can't stand cats, their effects may be less beneficial.

Japanese IT cats: https://youtu.be/vd21SH6zAX4

Kitten library: https://youtu.be/OKq22oF4FiQ

ELEKTOR STORE HIGHLIGHTS

RPI 4 – SDR Hands-On – DIY Geiger Counter



The new Raspberry Pi 4 – in three versions

At last, the new Raspberry Pi 4 hardware has arrived. The last time new hardware arrived the steps forward were modest if not small. The new RPi4 though has new things to offer. The first thing you notice if you grab the new Raspberry Pi 4 is that it won't fit any of the usual enclosures. This is due to some major changes in the display connection and power supply. We now have a USB-C connector for power, and two micro-HDMI ports to connect our 4K screens to. So, for the first time we have hi-res dual screen support through a digital interface, and powering from your universal USB-C supply.

You will also note the two blue USB ports. And yes, we finally got USB 3.0 onboard as well as native Ethernet, effectively ending all troubles while transferring data from our mass storage device to the network. It's all thanks to the new BMC2711 SoC at the heart of Raspberry Pi 4, which sports four Cortex-A72 cores with up to 1.5 GHz speed and a maximum of 4 GB RAM. Some trusted features that remained on the new Raspberry Pi 4 are the RCA port, display connector and the camera interface. Also untouched and functionally backwards compatible is the 40-pin PCB header. So we can attach HATs like we used to, at least hardwarewise.

Since we now have two micro-HDMI ports, we hooked up two Full HD screens to the Raspberry Pi 4.

At last we have what many of us have sorely missed: proper USB ports and not just a USB 2.0 with an ugly uplink to the SoC. The two blue ports and the new USB Hub are here to say: we entered the USB 3.0 world. Also, Ethernet is no longer attached to just one of the USB ports of the Hub — instead it is connected directly to the SoC, meaning more USB bandwidth for our devices and offering a big step forward towards network throughput.

One thing the older Raspberry Pi versions have lacked for a long time was the ability to use 3D acceleration on the desktop. With the Raspberry Pi 4 and the driver for the graphics part of the VidoeCore VI, it's back finally. As a result, on the desktop we now have 3D acceleration in 'windowed' mode, and it's working! For many of you who already own some Raspberry Pi or are looking to start with some existing extension, there is the inevitable question: will add-ons still attach to Raspberry Pi 4? Yes they will, but if you are using one of these TFT Displays that connect to HDMI you need at least a new cable for it.

@ WWW.ELEKTOR.COM

→ Raspberry Pi 4 with 1 GB RAM
www.elektor.com/18966

→ Raspberry Pi 4 with 2 GB RAM
www.elektor.com/18965

→ Raspberry Pi 4 with 4 GB RAM
www.elektor.com/18964

→ Official case for the RPi 4
www.elektor.com/18963

→ Official Power Supply
www.elektor.com/18962

Eventually we got the upgrades we waited for, faster USB, faster network, a faster CPU and finally more RAM. This comes at a price, first expressed in currency and secondly in current! With the entry models with 1 GB RAM roughly at the same price level as Raspberry Pi 3 B+, you'll see an increase in cost as you increase the amount of RAM you wish to have. At the same time, you get an increase in power consumption and heat dissipation.

@ WWW.ELEKTOR.COM

→ SDR Hands-on Book (Burkhard Kainka, DK7JD)
Book: www.elektor.com/18914

E-book: www.elektor.com/18915

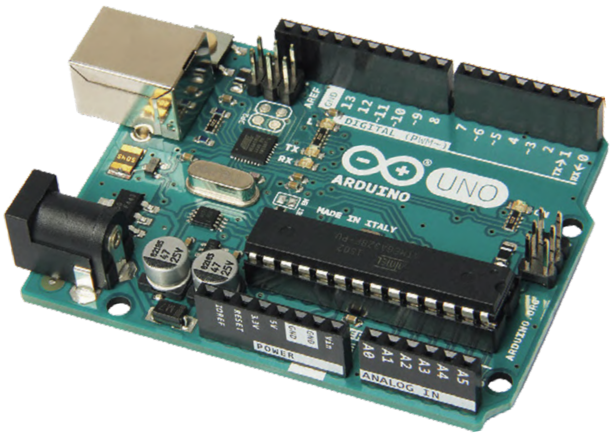
→ Arduino Uno R3
www.elektor.com/15877

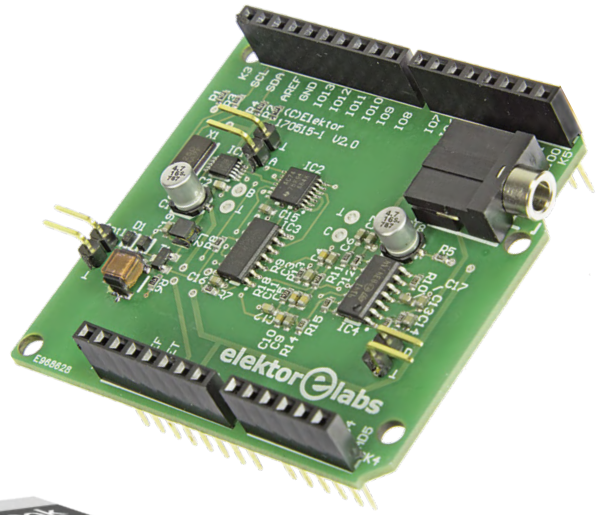
→ SDR Shield 2.0
www.elektor.com/18515

SDR Hands-on Book

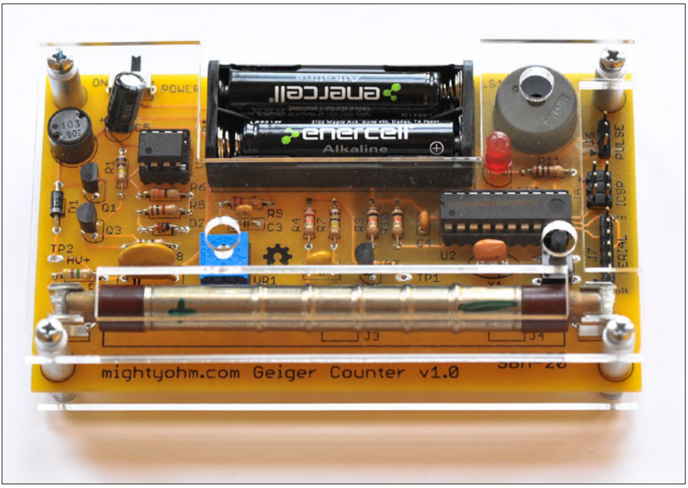
Author Burkhard Kainka's inspiring *Foreword* in the book I'm reviewing here has important clues as to why software defined radio (SDR) is a hit not only among existing radio amateurs. SDR has proven to be a great promotion tool for the declining radio hobby in general. Since the tuning and decoding of shortwave radio signals is now done with a minimum of hardware and powerful (free) software, it's easy to see why newcomers to the radio hobby have been totally captured by the concept of SDR

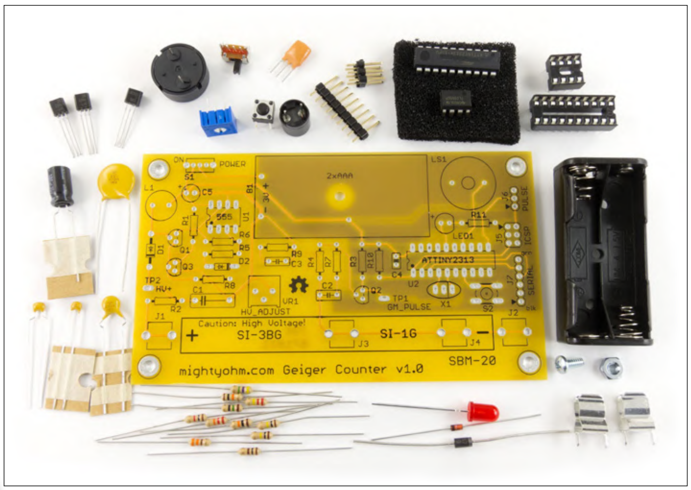
The days are gone when shortwave radio building was black magic with complex coils, RF parts and specialized construction techniques — these days all you need is a simple RF frontend board with an I-Q mixer, and... yes... software, usually running



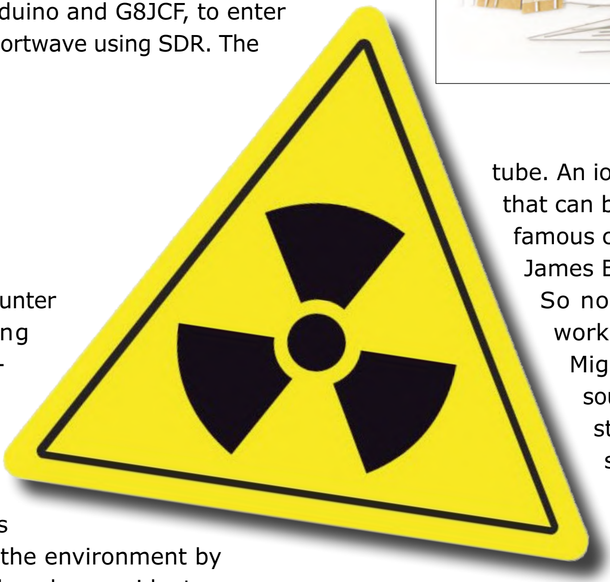


on a laptop with a quality sound card built in. All of the 'brains' of SDR is vested in software.
The correct order of working your way into SDR is: (1) do the hardware and understand it, (2) run the software. The book therefore first presents a shield (plug-on board) for the Arduino Uno.
The author discusses the schematic and theory of operation of the SDR shield in some detail.
Optionally the Elektor LCD shield can be plugged on top of the SDR shield. This configuration can be used to display the current SDR frequency, and also doubles as a measuring instrument. Cheap and handy, and it gives the Arduino a little extra to do!
Chapter 2 then continues with the installation procedure for one of the oldest, free, and still best of SDR software packages: G8JCF. The book has a step by step guide on how to link the G8JCF SDR software to the Arduino with the SDR shield on it and running the author's firmware. The guide takes the form of a little get-u-going exercise to tune the complete setup to 7000 kHz in the 40-m band. Next comes the installation of SDRsharp — also written as SDR# — for the frequency range 24 MHz to 1800 MHz.
The chapter on antennas is a highly successful way of promoting SDR to a new audience: start at the beginning, with that one radio device you simply cannot replace by a piece of software: the antenna. Through his ser by step discussion of SW reception antenna properties and receiver matching, the author succeeds in curbing the expectations of 'apartment dwellers' to pick up signals from the other end of the globe with no more than a piece of wire
The book is a must for everyone wanting to use popular, low-cost elements like Arduino and G8JCF, to enter the fascinating world of shortwave using SDR. The title is spot on.





Build Your Own Geiger Counter



The MightyOhm Geiger Counter is a device for detecting beta and gamma radiation. This kind of radiation is emitted by caesium-134 (^{134}Cs) and caesium-137 (^{137}Cs), i.e. radioactive isotopes of caesium released into the environment by nuclear weapon tests and nuclear accidents, Uranium (U) and thorium (Th), two of the most common radioactive elements on earth also produce such radiation. Their so-called decay chains pass through a phase of radon (Rn) which under normal circumstances is gaseous and easily inhaled, and therefore is considered a health hazard. Because radiation is so dangerous, you may want to keep an eye on the environment you live in and invest in a radiation detector like the MightyOhm Geiger Counter.
The detector produces a beep (a 'click') and a light flash every time the gas inside its sensor, a Geiger-Müller (G-M) tube, is ionized by passing radiation. To make this process work, a voltage of several hundreds of volts is required to bias the G-M tube. An ionization event results in a short current pulse that can be measured and made audible; these are the famous clicks you have probably heard in movies (e.g. James Bond).
So now we understand how the Geiger counter works and what it is for, let's build one using the MightyOhm Geiger Counter kit. It's 100% open source and open hardware, meaning that you can study it in detail. It features an LED and a piezo speaker to 'view' and 'hear' the radioactivity levels, respectively.
Headers are available for serial communication (3.3 V signal levels), in-circuit serial programming (ICSP) of the AVR microcontroller, and pulse output
The MightyOhm Geiger Counter is an easy-to-build kit that quite accurately detects radiation. Its logging capabilities make it a very usable instrument.

180689-E-03

@ WWW.ELEKTOR.COM
→ MightyOhm Geiger Counter Kit++ Bundle
www.elektor.com/18509

ICL7660 Output Noise Reduction

using a second-order Pi filter

By **Hesam Moshiri**

Some designs such as opamp-based amplifiers are sensitive to supply rail disturbances, forcing designers to keep noise levels down to their minimum. In this article, we look at suppressing the ICL7660's output noise efficiently and alongside learn how to connect the voltage converter part to the main circuit for best EMC compliance. The aim of of all is to preclude the '7660's switching currents affecting the main circuit in any way.

To provide a clean, negative supply rail is an unexpected challenge in many single-battery powered applications. The ICL7660/MAX1044 is a well-known chip to "drop in" and easily build a negative supply rail. Though the single chip and a few passive components look like a good option, some design challenges are inherent. The chip, a monolithic switched-capacitor voltage converter for sure has an efficient voltage conversion rate, but unfortunately it does add switching noise to the output voltage. Here we present one way of getting a really clean output voltage from the '7660.

Passive analogue filter

An analogue filter is an important electronic block with wide applications in signal processing. It is also used as a pre-filtering block for many analogue to digital converters (ADCs).
Three main types of filter exist, basically: high-pass, low-pass and band-pass/band-reject. In our application, the filter has to be simple, cheap and not limiting the output current in any way. Furthermore, it should not introduce instability or significant voltage loss at the output.

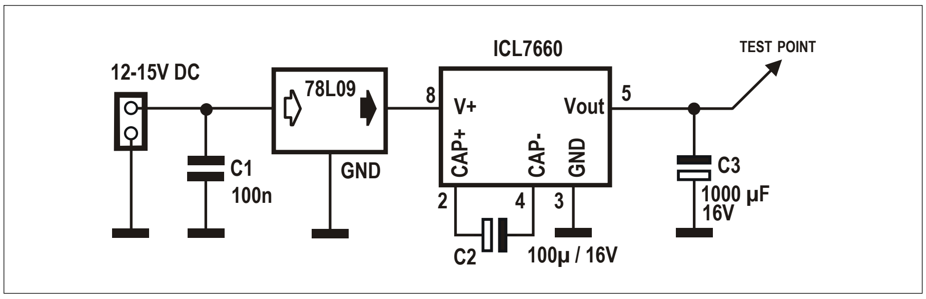


Figure 1: Supply-inverting configuration of the ICL7660 chip.

Noise from the ICL7660

Figure 1 shows the ICL7660 in a typical "inverter" configuration. Here it is configured to generate -9 V at the output. The converter's capacitor values (C2 and C3) got selected as optimal for a practical application.
Before connecting the oscilloscope to the circuit, let's first check the oscilloscope's internal noise in the laboratory setting. **Figure 2** shows the oscilloscope's internal noise floor in single-channel mode, with 2 mV/division scale and 20 MHz limited bandwidth. It demonstrates around 2 mV DC offset.
With the probe connected, the oscilloscope picks up a significant amount of hum (50 Hz), which his shown in **Figure 3**.
One method to eliminate hum-type noise and evaluate environmental noise levels with a connected probe is to connect a battery to the probe. The battery

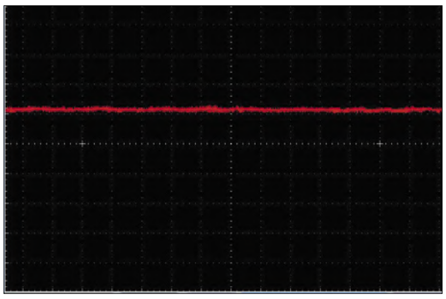


Figure 2: Measured noise floor of the oscilloscope (no probe connected, 2 mV/div).

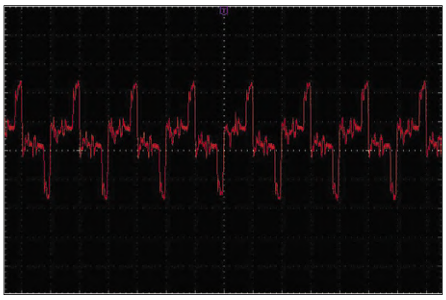


Figure 3: Mains-induced 50-Hz noise inevitably gets picked up by the oscilloscope probe.

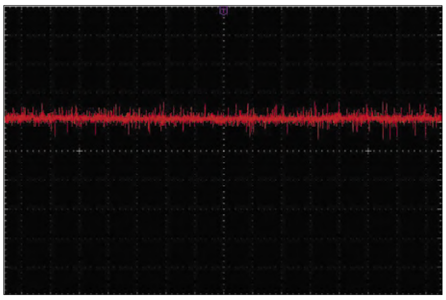


Figure 4: Environment noise with a battery connected; X1 probe setting; AC-in on scope.

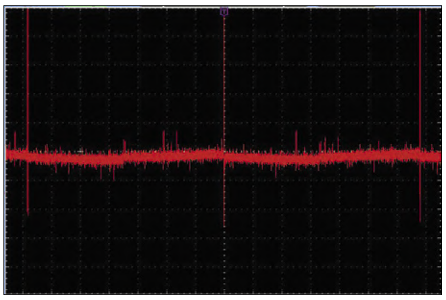


Figure 5: ICL7660 output voltage without additional filtering; 2 mV/div.

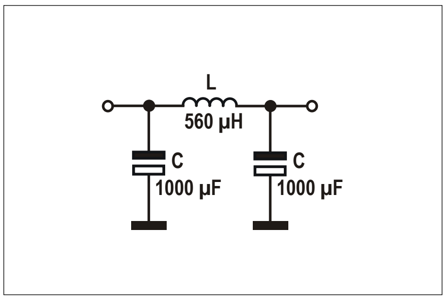


Figure 6: A first-order low-pass pi filter, to get started.

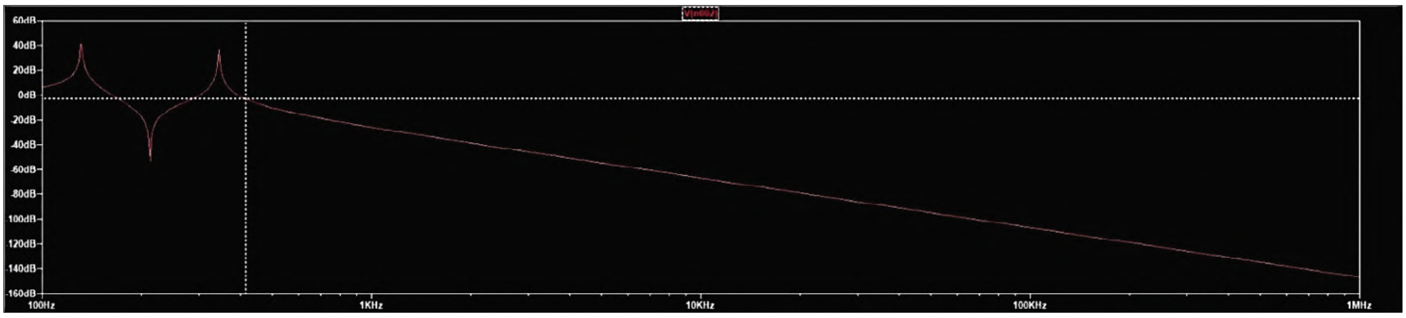


Figure 7: Frequency response of the first-order filter.

is a pure source of DC voltage, so the observed noise on the AC input would be the disturbances existing in the test environment, and naturally appear in an alternating shape. Therefore, the oscilloscope input must be set to AC-in mode. **Figure 4** demonstrates actual noise present when the battery is connected to the probe.

After this step we are ready to build the first phase of the ICL7660 voltage converting circuit and connect the oscilloscope probe to the “test point”. The circuit must be powered from a battery because we want to be dead sure that the circuit is supplied by a noise-free source, and what is measured over and above the noise line in Figure 4 is the switching noise generated by the circuit. **Figure** 5 shows the “test point” signal on the scope (2 mV/div). Although we have used a 1000-uF capacitor at the output affording serious filtering on its own, the supply rail is still infested with noise. The oscilloscope measurements indicate that the frequency of the main switching noise is around 7.3 kHz and has a varying amplitude reaching up to to 15 $m_V$pp. In order to make a pure DC voltage (as much as possible), we have to use a low-pass filter and block all noie, aand overow the observed noise frequency.

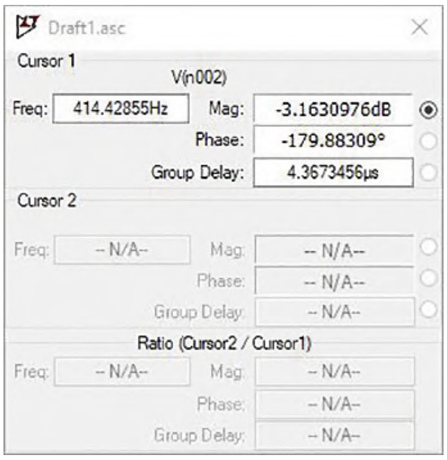
RC and LC filters are our options here. I do not plan to use RC filters because the resistor element inevitably limits the current and introduces a voltage drop, especially in higher-order filters. Therefore, the LC type filter is preferred. In practice, the 3-element filter behaves more efficiently, so we keep the capacitor C3 and add an LC filter stage to the output as shown in **Figure** 6. This is called a CLC or a ’pi’ filter.

A feature of almost any filter is its a cutoff frequency. For instance, a low-pass filter blocks signals above its cutoff frequency while passing signals that remain below this ’threshold’. We can use a circuit simulator such as LTSpice to predict the behaviour of the filter. Also, we can examine the filter’s efficiency by modifying the component values, L and C. For this purpose, we should select values creating a low cutoff frequency and do the “sharper” filtering right in front of the noise.

The output capacitor (C3) already got selected as 1000 µF, so let’s continue with this value. You’re not tied to this value though. Especially if the size of the PCB is important, you can select your own desired capacitor values and simulate the filter’s response before proceeding to PCB design and construction. **Figure** 7 shows the frequency response of the filter. The inductor value was set to 560 µH. The AC sweep frequency was set to 100 Hz to 1 MHz.

The cutoff frequency of the filter is the frequency on the X scale, where it meets the -3 dB level on the Y scale.

LTspice cheerfully performs the calculations which are shown in the **Figure** 8. The cutoff frequency is around 414 Hz and the phase shift is almost -180 degrees. It should be noted that these calculations result from ideal the-


Draft1.asc

Cursor 1

V(n002)

Freq: 414.42855Hz Mag: -3.1630976dB
Phase: -179.88309°
Group Delay: 4.3673456µs

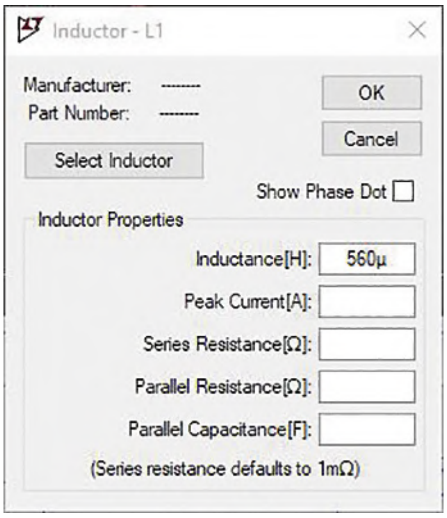
Cursor 2

Freq: --N/A-- Mag: --N/A--
Phase: --N/A--
Group Delay: --N/A--

Ratio (Cursor2 / Cursor1)

Freq: --N/A-- Mag: --N/A--
Phase: --N/A--
Group Delay: --N/A--

Figure 8: The filter’s cutoff frequency and phase shift according to LTSpice.



Inductor - L1

Manufacturer: --------
Part Number: --------

OK
Cancel
Select Inductor
Show Phase Dot ☐

Inductor Properties

Inductance[H]:	560µ
Peak Current[A]:	
Series Resistance[Ω]:	
Parallel Resistance[Ω]:	
Parallel Capacitance[F]:	

(Series resistance defaults to 1mΩ)

Figure 9: Aan inductor's modifiable parameters in the LTspice.

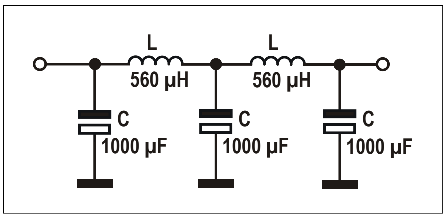


Figure 10: Second-order low-pass pi filter, with the selected values.

oretical components. If you intend to do more realistic simulations, you can modify the component values with the datasheet parameters. **Figure** 9 shows the LTSpice parameters provided for the inductor, such as the series resistance, peak current and so on.

This is a first-order CLC or pi filter. We can make the filter more powerful by increasing the filter order. So, we can build a second-order pi filter with the same component values. **Figure** 10 shows the circuit diagram.

If we examine the filter behaviour in LTspice, it is clear that the second-order filter gives a sharper and more stable response. In simple engineering terms it does the filtering job better. The frequency sweep in **Figure 11** shows the second order filter response (green) in comparison with the first-order filter response (red).

In reality

Now, it's the time to test the ICL7660 converter circuit with the finalized filter.

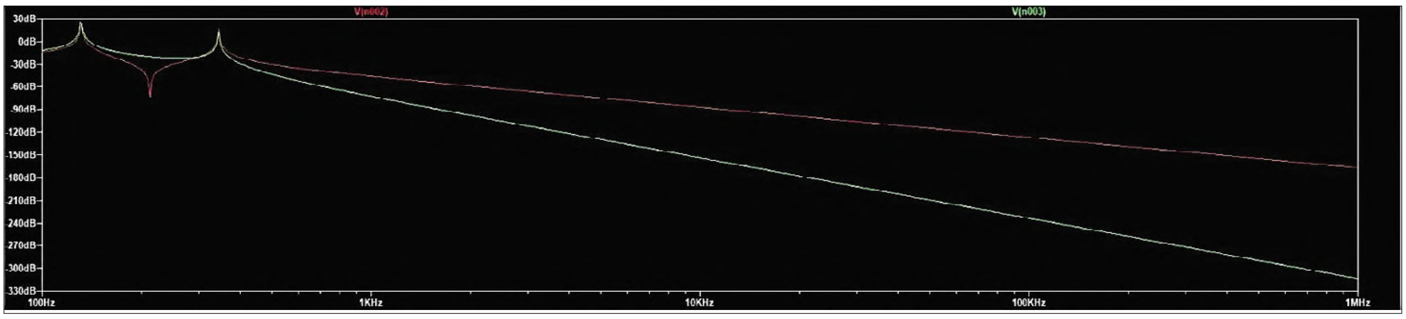


Figure-11: Second-order pi filter response (green line).

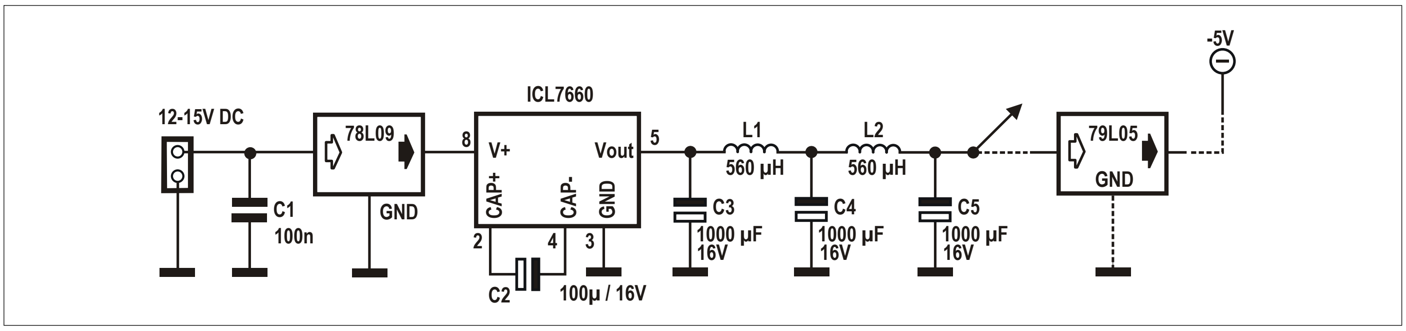


Figure 12: Complete circuit diagram of the 7660 voltage inverter, including an optional negative voltage regulator.

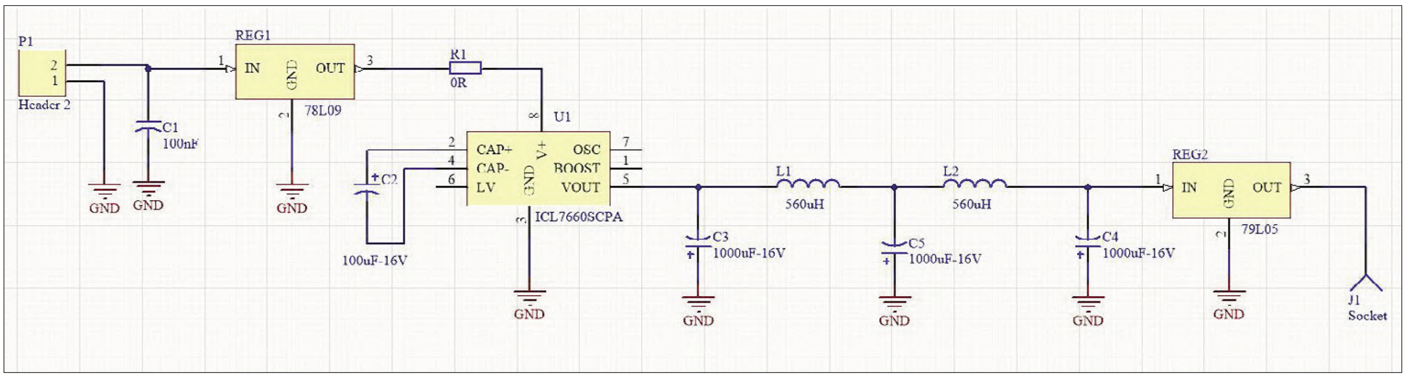


Figure 13: Schematic of the low-noise supply converter circuit as done in Altium Designer.

Figure 12 shows the complete circuit. It should be mentioned that the output voltage of the ICL7660 is not regulated. Therefore if you need a stable regulated voltage, such as -5 V, it is always wise to use a negative voltage regulator at the filter's output, as shown.

The correct way to build and test the circuit is to design a proper PCB for it. I used Altium Designer for this purpose. Also, I used the SamacSys component search engine/Altium plugin to find the components footprints. Using this service reduces the PCB design time significantly because it eliminates the repetitive schematic library/PCB footprint design task. **Figure** 13 shows the Altium-style schematic and **Figure** 14 the PCB respectively. The PCB design files are a free download from the ElektorLabs website [1]. The first PCB design tip is to use a proper copper-pour ground plane i.e. fill the free areas. The ground plane helps to reduce noise and transients. Resistor R1 is a zero-ohm resistor used here to avoid any unnecessary PCB ground plane cut-off below capacitor C2. The solid ground plane maximizes the low impedance current path area in the circuit.

According to EMC guidelines, a switch-mode power supply (SMPS) should not share a common ground with the rest of the circuit. Consequently, the connection between the SMPS ground and the main circuit ground plane should exist at a single point. Although I have used a ground plane for the SMPS circuit, the ground connection with the external circuit could only be made by a single header connection.

Figure 15 shows the built up PCB. And now, for the measured noise at the circuit output (at 2 mV/div), it is shown in **Figu**re 16. Comparing it to Figure 4, it's obvious how close it is to the oscilloscope's noise floor! ⏮

190127-01

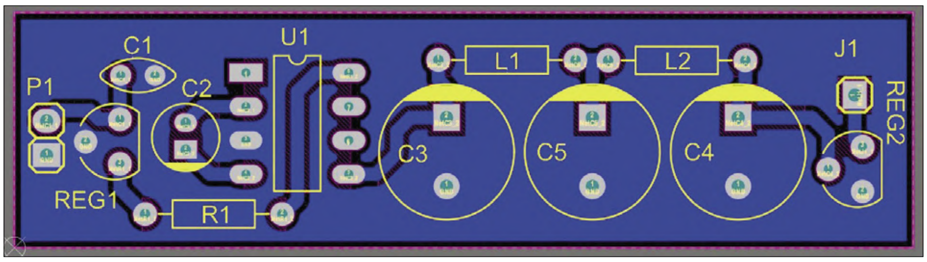


Figure 14: PCB designed for the converter circuit using Altium Designer.

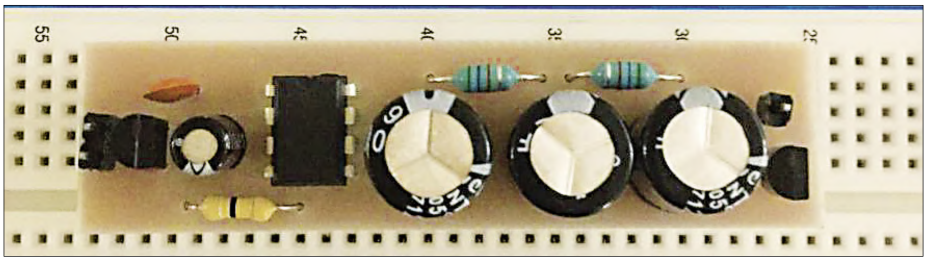


Figure 15: The assembled PCB board.

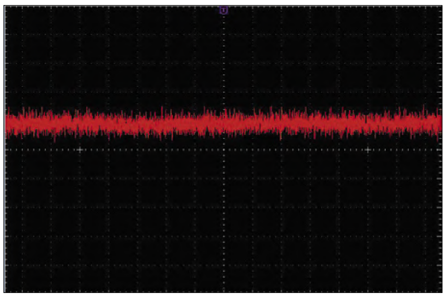


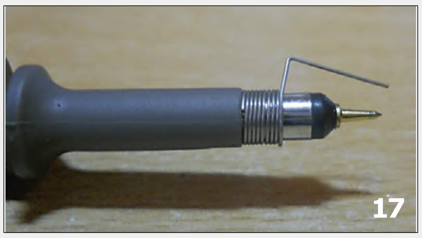
Figure 16: Converter output noise using the finalized auxiliary filter (2 mV division; X1), final results.

Web Link

[1] PCB design files (Altium): www.elektormagazine.com/190127-01

"Oscilloscope Best Practices"

For really precise measurements, the ground connection wire of the oscilloscope probe must be removed and the connection with the probe's body should be made through a ground spring. **Figure 17** demonstrates this type of connection. This way, you can be sure that the signal noise seen on the oscilloscope screen is from the test point in the circuit only, and environmental noise effects remain at their minimum level.


17

LoRa’s Big Event in Berlin

Three-day meeting showcases the latest IoT apps and technologies

By **Jens Nickel** (Editor-in-Chief, ElektorLabs Germany)

The Internet of Things is designed to make our lives more comfortable, safer and, above all, more resource-efficient. It thrives on a host of sensors that can transmit data wirelessly for years without changing batteries. Applications range from car park and warehouse management to the broad expanse of environmental sensors and the tracking of goods and animals. The demands on range, good building penetration and extremely energy-efficient operation mean that traditional wireless technologies such as WLAN and Bluetooth are no longer viable. On the other hand, new standards such as LoRa, Sigfox and NB-IoT support efficient radio data transfer with low bit rates, each competing with one another in this constantly growing marketplace. Base stations (also called gateways) receive the sensor data and then forward it to the Internet using WLAN or Ethernet. The ecosystems of each of these wireless technologies also include servers that make the data available to the owners of the sensor nodes (via a Web API, MQTT, e-mails and so on). For developers, there are low-cost development boards, free use of the infrastructure for evaluation purposes, software libraries and sample applications.

LoRa has secured a respectable position on the market. In physical terms, the radio standard is based on a CSS modulation process developed by semiconductor manufacturer Semtech, which also produces and licenses matching transceiver chips and SoCs. In contrast to its competitor Sigfox, however, there is not one single network, but a vibrant array of smaller and larger, both private and public networks. In order for sensor nodes to be able to register flexibly with the various networks, it is of course not only necessary for transceivers to understand each other on the physical level in order to transmit data back and forth as-is (this does also work, as we prove in this issue with the Wireless Automotive Battery Voltmeter project). Also necessary are registration procedures, collision management, timing and security features. The entirety of this is assured by the LoRaWAN standard, which is driven by a large and growing alliance of companies [1][2]. In addition to Semtech, the LoRa Alliance [3] embraces not only other semiconductor manufacturers but also telecommunications providers, software foundries and many user organisations. Numbering more than 500 members, the Alliance meets annually in a succession of cities — and in 2019 it was Berlin’s turn to host the event [4]. Throughout the first two days, meetings were held in closed circles for finessing the documentation and sharing insights; on the third day, representatives of the press and (potential) customers were also invited along. Time was also found for a little self-congratulation of course. Following the keynote speech by Alliance leader Donna Moore, who highlighted the consortium’s growth during the past year, awards were given to experts who had excelled in protocol development and support. Then the sponsors of the event had their say. Naturally, the presentations also discussed differentiating oneself from the competition. The market research company ABI Research has published a study that concentrates on examining the differences between LoRa and NB-IoT (**Figure 1**) [5]. Inevitably, LoRa partners are likely to suffer from the fact that many European telecommunications providers are not represented in the Alliance and are promoting their own standard with NB-IoT. Weighty arguments for the latter emphasise the pre-existing network infrastructure and specially reserved frequency ranges, whereas LoRa in Europe operates in the less structured 868 MHz band. In Berlin, speakers nevertheless stressed how LoRa sensor nodes can be more energy-efficient, cheaper and more compact. Underscoring this point,

Figure 1. On stage in Berlin: a study comparing LoRa with NB-IoT.

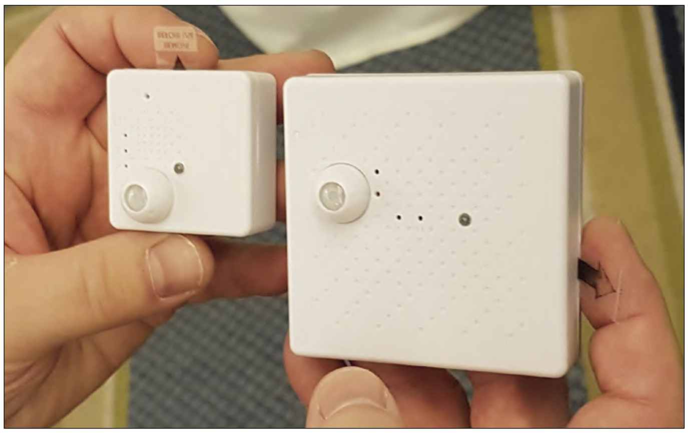
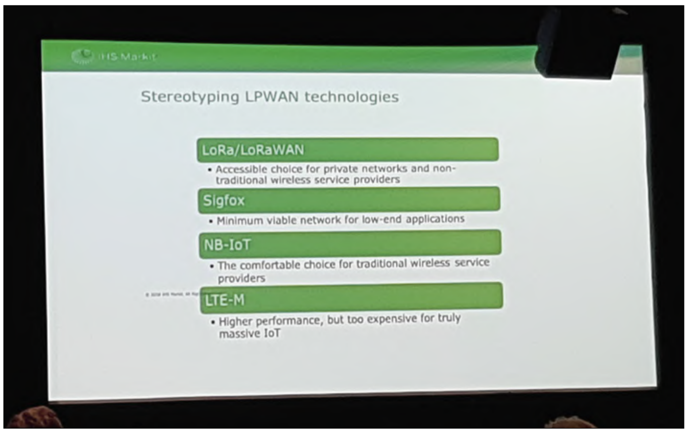


Figure 2. Two identical multi-sensors side by side: one using LoRa and the other for NB-IoT data radio (www.tektelic.com).



Stereotyping LPWAN technologies

LoRa/LoRaWAN
- Accessible choice for private networks and non-traditional wireless service providers

Sigfox
- Minimum viable network for low-end applications

NB-IoT
- The comfortable choice for traditional wireless service providers

LTE-M
- Higher performance, but too expensive for truly massive IoT

Figure 3. LoRa contrasted with NB-IoT and Sigfox.

LoRaWAN
Creating Valuable
Polygravity
Enabling Monetization of LoRaWAN® Roaming
Introducing a real-time, blockchain-based, billing engine

Figure 4. Startup company Polygravity set out their solution for roaming communication and trans-network payment.

the associated product exhibition provided a demonstration of a multi-sensor node (**Figure 2**).

At the other end of the application spectrum, Berlin rated the competing Sigfox technology as being suitable only for low-end applications in which very few bytes per hour or day are transmitted (**Figure 3**). The Sigfox package has, for sure, an undoubted simplicity: one single company licenses the chips, operates the base stations and the server platform with which the users can manage the sensor nodes and receive the measurement data. LoRa, on the other hand, is much more diverse. Users must either set up their own network (gateways plus server) or register the sensor nodes with a commercial network operator. The expansion of these networks is usually limited, although roaming solutions are already available for applications that require a high degree of mobility, such as tracking goods in long-distance traffic (**Figure 4**).

A free LoRa network striving for global coverage is TheThingsNetwork (TTN) [6], which is organised by volunteers, who each operate one or more gateways. This also includes a network platform from which the data from one's own sensors can be retrieved using an API. TTN founder Wienke Giezeman was likewise present in Berlin. LoRaServer.io [7] also operates an open network server. A workshop at the event demonstrated how a LoRaWAN gateway can be built with a Raspberry Pi, a connected gateway module and open source software. This can forward data to TheThingsNetwork as well as to LoRa Server (**Figure 5**). If you discount the time that the Raspberry Pi spends downloading the Github data and installing it, the whole setup is done in just minutes.

In this article we can only touch on much of what went on at the expo but rest assured that we're already working on further Elektor Labs articles on the subject, as well as some tasty LoRaWAN projects!

190340-02

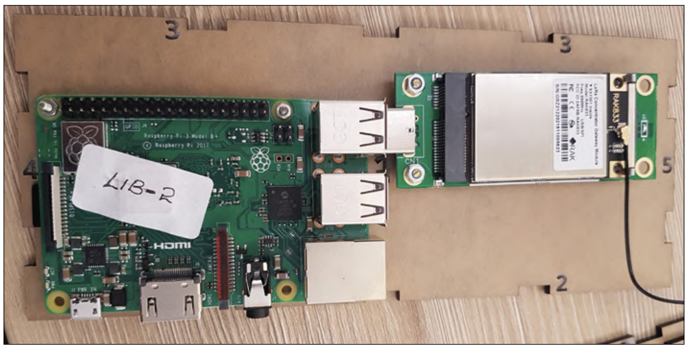


Figure 5. In its 'LoRaWAN in a box' workshop ICFOSS from India showcased exactly what could be created with just a Raspberry Pi and a gateway module.

@ WWW.ELEKTOR.COM

→ Dragino LoRa IoT Development Kit
www.elektor.com/18335

→ LoRa Nexus Board
www.elektor.com/lora-nexus-board-arduino-mini-shape

Web Links

[1] LoRaWAN Specs: https://lora-alliance.org/sites/default/files/2018-07/lorawan1.0.3.pdf

[2] Software elements: https://lora-developers.semtech.com/resources/tools/basic-mac/welcome-basic-mac/

[3] LoRa Alliance: https://lora-alliance.org

[4] LoRa Alliance expo in Berlin: https://lora-alliance.org/events/berlin-annual-members-meeting-2019

[5] LoRaWAN and NB-IoT comparison : https://lora-alliance.org/resource-hub/lorawan-and-nb-iot-competitors-or-complementary

[6] The Things Network: www.thethingsnetwork.org/

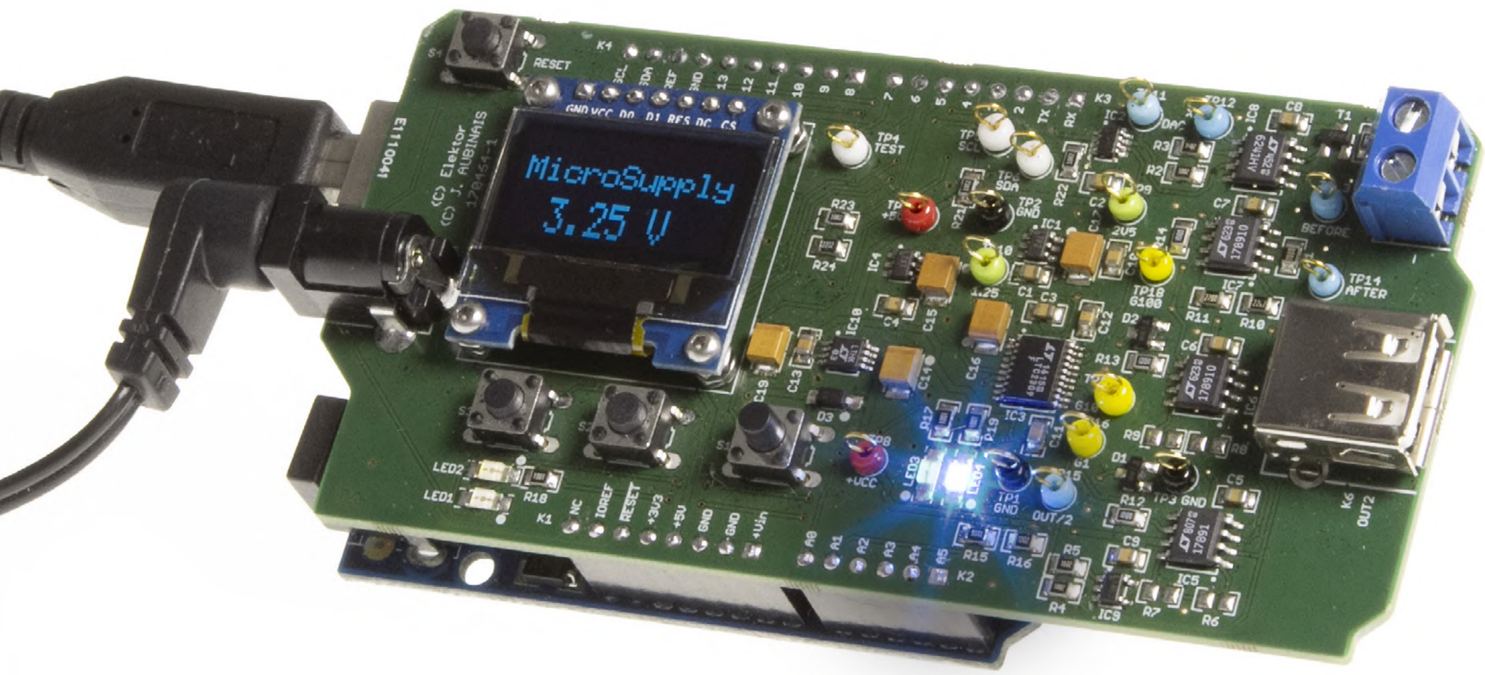
[7] LoRa Server: https://www.loraserver.io/loraserver/overview/

MicroSupply

A lab power supply for connected devices

By **Jennifer Aubinais** (France), thanks to **Jean-Luc** and **Philippe**

How much power does my connected device use? Does its sleep mode conform to the manufacturer's documentation? How long will the CR2032 battery last? If you're asking yourself these sorts of questions, here is a regulated power supply, dubbed MicroSupply, in the form of an Arduino shield, which measures very small currents. Thanks to a link with a PC application, you can see and log the power consumption of your device.



Quick Specifications

- Shield for Arduino Uno.
- Output voltage regulated from 1.5 to 5 V in steps of 0.05 V.
- Output current measured and displayed, from 1 µA to 40 mA.
- Measurement frequency: 1 kHz.
- PC software to display measurements received via the serial port.
- Software circuit breaker at 40 mA.
- Software circuit breaker (Output voltage reduced by >100 mV).
- Requires external 10-12 V supply

This is a project with a long history - it's already up to version 5! The project has evolved because of the numerous technical difficulties encountered since its conception, but also thanks to the author's exchanges at seminars and shows. Improvements will still be possible, but one must know when to stop...

Overview

Before looking at the electronics in detail, here are the different modules which make up the project (**Figure 1**):

- The supply voltage for our connected device is set using a digital to analogue converter (DAC).
- As the converter itself cannot supply enough current, a small power amplifier, with a maximum of 40 mA, is used.
- The 'shunt resistor' as its name implies, allows the measurement of current through it by measuring the voltage across it. As this resistance will often be referred to, it is referred to as a module.
- Before converting the voltage across our shunt resistor, several instrumentation amplifiers are used. Why several? The measured current may vary from 1 μA to 40 mA, a range of 1 to 40.000. It is not easy to cover such an amplitude range with a single amplification. That's why three instrumentation amplifiers cover the necessary range: ×1, ×10 and ×100.
- An analogue to digital converter (ADC) measures the different voltages (which are measures of current) at the outputs of the three instrumentation amplifiers.
- Finally, an OLED screen shows the output voltage and current. The screen is also used to show numerous messages like the steps to verify the whole project during testing.

The first version...

... was a simple Arduino shield which delivered an output voltage of 0 to 5 V and which measured the current at the ends of a small shunt resistor. A 0.96-inch OLED display periodically showed the measurement.

After that it became apparent that unfortunately the measurements were not being transmitted to a PC for inspection, and, who knows, processing or logging. The object was not to make a digital oscilloscope, but to have a decent sampling frequency. Too low, and it would not be very useful, as we're not interested in the consumption every hour or every day; too high a rate, every microsecond for example, would be a very complex project. The eventual choice was a sampling frequency of 1 kHz, that's to say 1000 measurements per second. The ADC initially chosen was not fast enough for a conversion every millisecond, so it had to be replaced.

The shunt resistor

Another problem was posed by the desired measurement range, from 1 μA to 40 mA. In the first version, the shunt resistor had a value of 10 Ω. A current of 40 mA thus produced a voltage of 400 mV across this resistor, for 1 μA it was 10 μV. That's not much. A higher shunt value, for example 100 Ω, partly fixed this problem, but the input voltage of the ADC could not be exceeded. In the last case, if the supply voltage was 5 V and it consumed a current of 40 mA, with a shunt of 100 Ω, it would need a voltage of 9 V (i.e. 5 + (0.04 × 100)) to supply the whole circuit. Briefly, the higher the value of the shunt resistor, the higher the voltage across it for low currents, but the higher the supply output voltage needs to be.

PROJECT DECODER

Internet of Things
Power supply
Arduino

entry level
intermediate level
➡ expert level

4 hours approx.

SMD soldering,
Arduino EDI,
lab power supply

£65 / $80 / €70 approx.

Three instrumentation amplifiers

An amplifier with variable gain is the ideal solution. For example, a gain of 1 for the top of the range, a gain of 10 for the middle and a gain of 100 for low currents, allows us to measure the current with the same precision whether it is low or high. To measure the voltage across the shunt, we need a differential amplifier, preferably of good quality, and this is why we opted for an instrumenta-

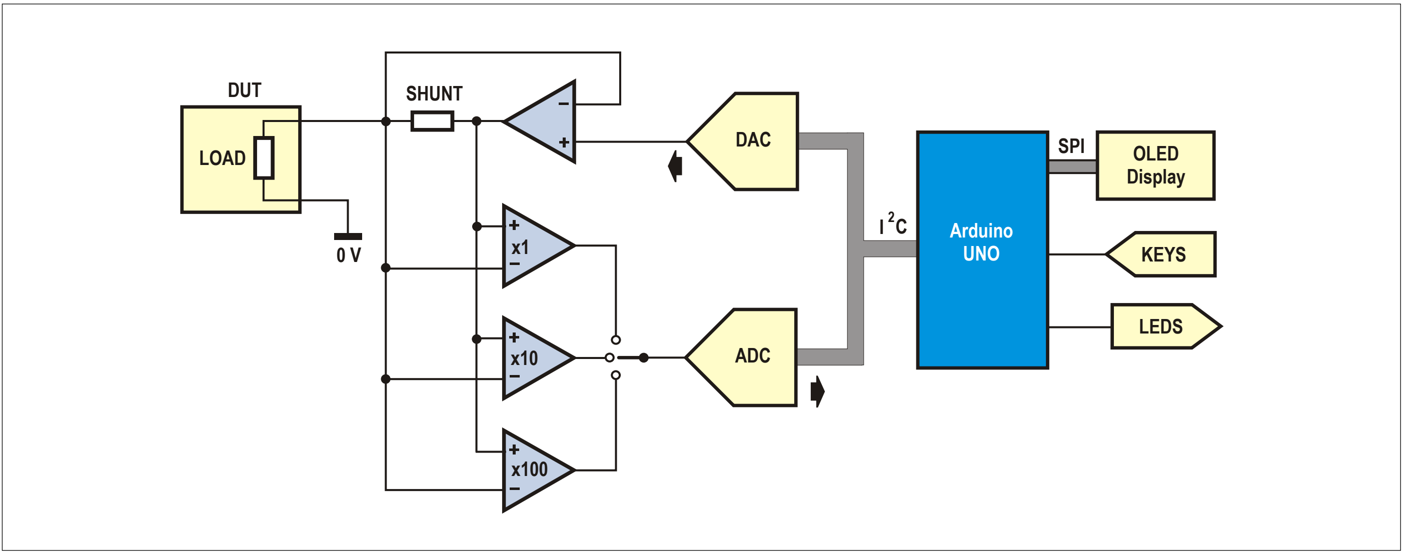


Figure 1. The block diagram of the MicroSupply.

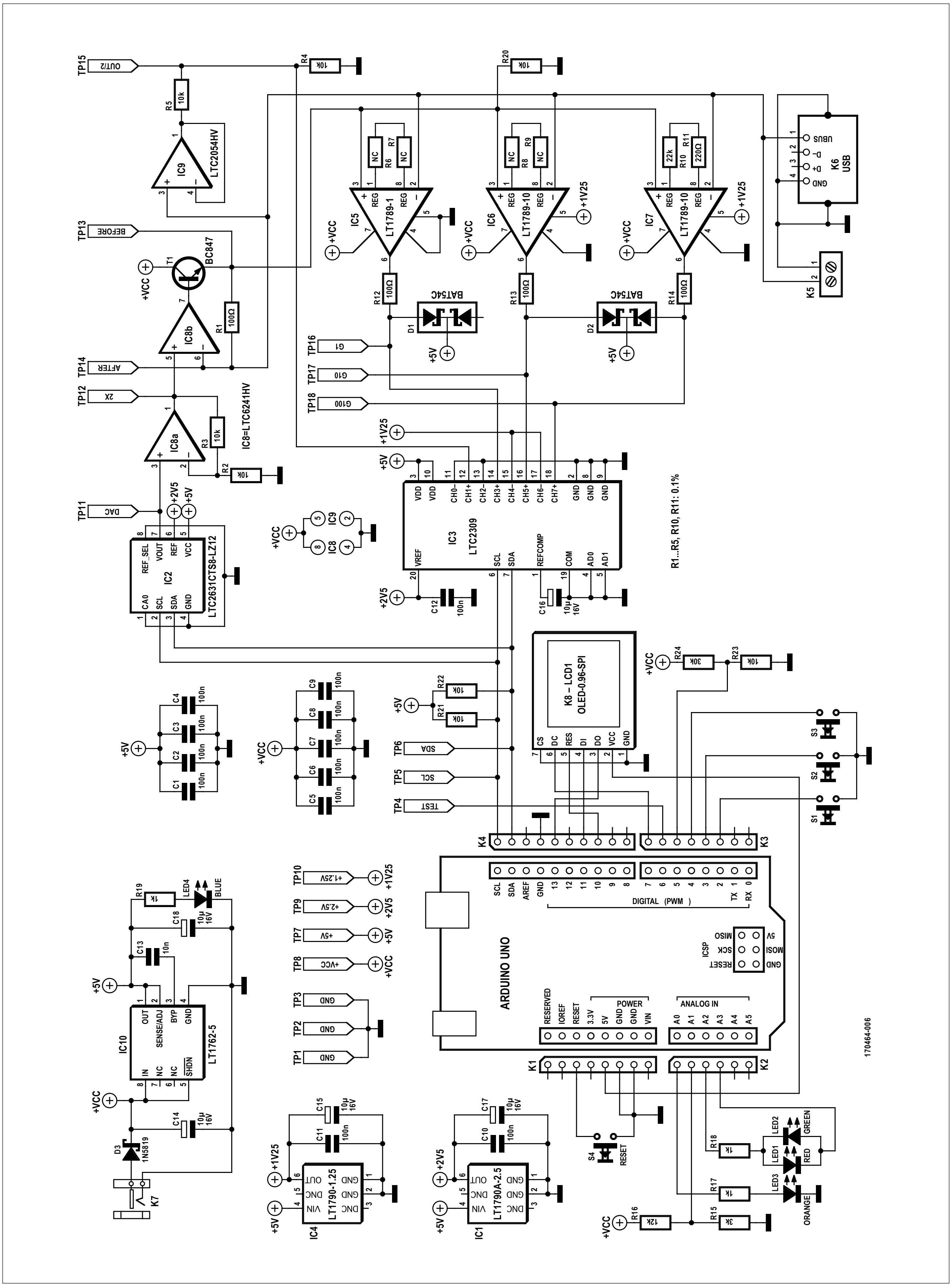


Figure 2. The circuit diagram of the shield.

tion amplifier. Unfortunately, this type of amplifier has a fixed gain. It is possible to change the gain by switching some resistors, but that needs several switches and the signals to change them (plus a bit more software). The solution finally adopted was to use three amplifiers with different gains in parallel.

The output voltage
A digital to analogue converter (DAC) is used to create a voltage adjustable from 0 to 5 V, which is, in the final version of the project, from 1.5 V (a higher value than the 1.25 V reference voltage of the instrumentation amplifiers, more on this below) at 5 V. With a reference voltage of 2.5 V and a precision of 12 bits, the DAC gives an output voltage of 0 to 2.5 V in steps of 0.61 mV (i.e. 2.5 / (2^{12}−1) = 2.5 / 4095). To get an output of 5 V, we use an operational amplifier with a gain of 2. So the steps are then 1.22 mV.

Reducing the power supply output resistance
One remaining problem is the elevated series resistance of the power supply due to the value of the shunt. The objective is to supply a constant output voltage, independent of the load current, so this resistance has to be compensated for. This is done by creating a feedback loop around the shunt. An op-amp with buffered output (by T1, to supply an output of 40 mA max) ensures that the output voltage remains constant, even if the current varies between 1 µA and 40 mA. Other than the fact that we need an output current of 40 mA, this corresponds to the maximum current thorough the 100 Ω shunt resistor without exceeding the maximum input voltage of the ADC (4.096 V).

It needs an external supply
We now have a system capable of supplying a stable voltage from 1.5 to 5 V and measuring the current supplied. The price to pay, apart from the complexity of the circuit, is that a higher input voltage of at least 9 V is needed (see above). The first version used for this a voltage doubler, the LTC660, which transformed the +5 V available from the Arduino board to +10 V. However, the voltage thus obtained carried a lot of noise that we did not succeed in reducing sufficiently. Alas, we ended up removing the doubler and replacing it with an external supply. The decision was justified by the fact that the MicroSupply is a laboratory instrument, and any self respecting lab should have a lab supply worthy of the name.
A short note — when we started we used an external supply voltage of 22 V which allowed us to use a shunt resistor of 400 Ω,but this is not the typical way of doing this, it's a relatively high value and is considered too restrictive, so we reduced the voltage to 10 V minimum (9 V plus the voltage across D3 and T1), 12 V maximum, because of the supply limitations of IC8.
Note that the Arduino board is powered either by its USB port (whether you're using the PC to program or not) or by an external supply which can be the same used for the shield.

The circuit in detail
For once we will start with the power supply (**Figure 2**). Because this is supposed to be provided by a good quality lab power supply, the handling on the board is limited to regulation (by IC10) and some capacitors, to produce the 5 V needed for the DAC and ADC. The amplifiers are powered directly from the input voltage (there is however a protection against reverse polarity: D3). The OLED display is powered from the 5 V from the Arduino board. The power supplies for the analogue and digital sections are thus separated without much effort.

Digital to Analogue Converter
IC2 is the DAC which produces the output voltage of our MicroSupply. This IC offers a resolution of 12 bits and has an internal reference voltage of 2.5 V. The DAC gives an output voltage ('visible' on TP11, labelled 'DAC') of 0 to 2.5 V in steps of 0.61 mV (i.e. 2.5 / (2^{12} − 1)). to get an output voltage of up to 5 V, we use an op-amp, IC8, with a gain of 2 (TP1, '2×'). The resolution thus changes to 1.22 mV.

As pointed out above, the DAC IC2 has an internal reference, but the datasheet does not give a lot of information on the quality of this reference. For this reason we have added IC1, a high precision and well documented reference source. So we know what we are dealing with. You can observe this on test point TP9 ('2.5V').

'Power' amplifier
The other part of IC8 drives a transistor to supply the maximum output current of 40 mA. As it's an op-amp, it will try to adjust its output so that the voltages across its inputs, pins 5 and 6, are identical. Because of the connection of the inverting input (pin 6) to the output of the MicroSupply (TP14, 'AFTER'), this mechanism ensures that the output voltage is identical to that on the non-inverting input (pin 5), independent of the current through R1 and T1. If the op-amp is fast enough compared to the variations in current taken by the load, the load will see a source with a very low output impedance, which is our goal. IC9 divides the output voltage by two (TP15, 'OUT/2') for purposes of displaying it and also activating the software circuit breaker.

The shunt resistor and its amplifiers
R1 is the shunt resistor referred to above. The current taken by the load passes through R1 and produces a voltage across it (TP13, 'BEFORE' and TP14, 'AFTER') a voltage that we can measure. For this, we use three instrumentation amplifiers in parallel: one with a gain of 1 (IC5), a second with a gain of 10 (IC6) and a third with a gain of 100 (IC7). Pay attention to the suffixes of these components when you order them and solder them, because they are not exactly the same. IC5 has the le suffix '-1' while IC6 and IC7 have the suffix '-10'. The suffix indicates the pre-programmed gain. It's possible to change it using a resistor, which is what we have done to IC7 with R10 and R11 to get a gain of 100. It would be possible to use just one amplifier and switch this gain resistance to achieve our ends, but it's not sure whether we would gain anything in the end.
TP16 ('G1'), TP17 ('G10') and TP18 ('G100') are test points for the three amplifiers. Note: for the 'G10' and 'G100' points, they are referenced to the 1.25-V reference voltage, TP10.

Digitizing the measurements
The signals at the output of the amplifiers, plus that produced by IC9, are digitised by the Analogue to Digital Converter (ADC) IC3. The inputs can be used as differential pairs or single inputs. We use them in single mode for the outputs of IC9 and IC5. By comparison, the output signals of IC6 and IC7 are measured in differential mode relative to the reference voltage of 1.25 V, produced by IC4

COMPONENT LIST

Resistors

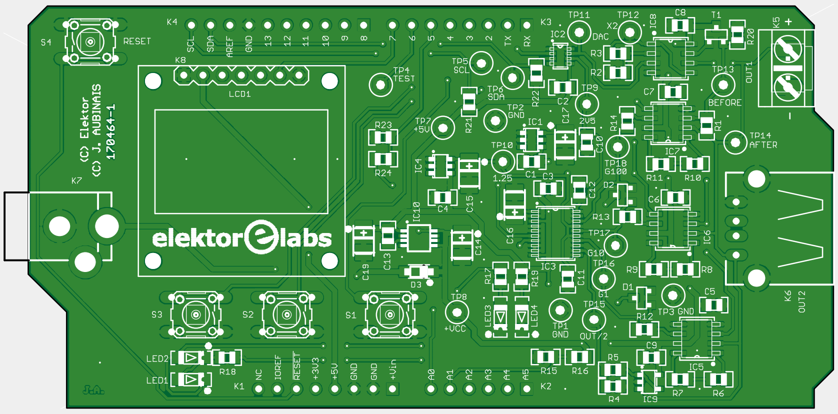
Default: SMD 0805
R6,R7,R8,R9 = not fitted
R12,R13,R14 = 100Ω
R1 = 100Ω 0.1%
R11 = 220Ω 0.1%
R17,R18,R19 = 1kΩ
R15 = 3kΩ
R20,R21,R22,R23 = 10kΩ
R2,R3,R4,R5 = 10kΩ 0.1%
R16 = 12kΩ
R10 = 22kΩ 0.1%
R24 = 22kΩ

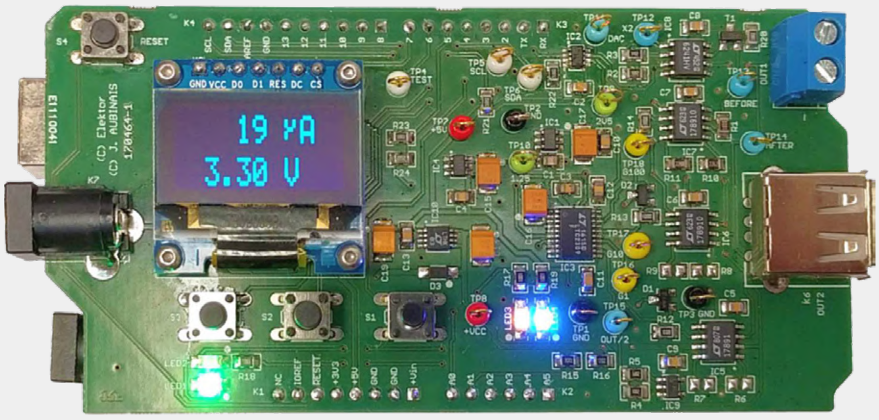
Capacitors

Default: SMD 0805
C13 = 10nF
C1,C2,C3,C4,C5,C6,C7,C8,C9,C10,C11,C12 = 100nF
C14,C15,C16,C17,C18 = 10µF 16V, tantalum, SMD 1411

Semiconductors

D3 = 1N5819HW-7-F
D1,D2 = BAT54C
T1 = BC847C
LED4 = blue, 1206
LED2 = green, 1206
LED3 = yellow, 1206
LED1 = red, 1206
IC10 = LT1762EMS8-5
IC5 = LT1789CS8-1
IC6,IC7 = LT1789CS8-10
IC4 = LT1790BCS6-1.25
IC1 = LT1790ACS-2.5
IC9 = LTC2054HVCS5
IC3 = LTC2309CF
IC2 = LTC2631CTS8
IC8 = LTC6241HVCS8





Miscellaneous

K8 = socket, 7-way, 0.1" pitch, for LCD1
K2 = pinheader, 6-pin, 0.1" pitch
K1,K3 = pinheader, 8-pin, 0.1" spacing
K4 = pinheader, 10-pin, 0.1" spacing
S1,S2,S3,S4 = tactile switch, 6x6 mm
K6 = USB socket, type A, angled
K5 = 2-way PCB screw terminal block, 0.2" pitch
K7 = DC Power connector, 1.95 mm central pin
LCD1 = OLED display, 0.96 inch, SPI, 7-pin
4x bolt, M2 x 8 mm
12x nut, M2
Precision resistors 1kΩ, 100kΩ and 1MΩ, 0.1%, for testing and calibration
Printed circuit board ref. 170464-1, from the Elektor Store (www.elektor.com)

(TP10, '1.25'). Why? Because the outputs of IC6 and IC7 cannot go below 110 mV, which is a bit inconvenient if we want to measure very small voltages close to 0 V. Happily, these ICs have an entry labelled 'reference' which allows us to raise the voltage at the output. Thanks to the differential inputs of the ADC we can then subtract the reference voltage to get a measurement relative to zero. We don't need to use this trick on IC5 because this lets us have a maximum voltage of 4.096 V at the output of the instrumentation amplifier when the current supplied is 40 mA.
Like the DAC, the ADC has a resolution of 12 bits, and, just like the DAC, it has an internal reference which we have preferred to replace with our own reference IC1. Finally, just like the DAC, (They're not from the same family, perchance?) the ADC communicates using the I²C but (with test points TP5, 'SCL' and TP6, 'SDA').

The display on the SPI bus

Then we come to the OLED screen which communicates by the SPI bus. This type of display is also available with I2C interface, but we wanted to separate the communication buses, not least because the SPI bus is better suited to graphical applications, because it permits much higher rates of data transfer.

What about the microcontroller?

Have we seen everything now, is that the end of our tour of the diagram of the MicroSupply? No, not quite, we've missed the microcontroller, the brains behind our project! You'll have guessed (well, we didn't try to hide it) that it's an Arduino Uno on which is found an ATmega328.
Apart from controlling our shield, which amongst other tasks controls the DAC. recovers the samples and displays the data on the OLED screen, the µC also communicates with a PC via the USB port.
To ensure that the input voltage to the shield is sufficient to power the load (>10 V), the µC measures the input

voltage via its analogue input A1. It will not continue initialising unless the input voltage is greater than 10 V.
Pin 5 of the Uno also monitors the input voltage, but more crudely. If the µC reads a low logic level on this input, the input voltage is less than about 9.6 V and the system will halt.
The output voltage is also monitored by the µC (with the aid of IC9) to ensure that there is no voltage drop greater than 100 mV (limited in the program). If this is detected, the output is shut off immediately and an error message is displayed.

A few words on the software

The Arduino sketch is not in itself complex, but a few lines of code are worthy of attention:

- **The switch debouncing system**. The push buttons S2 and S3 (Down and Up) are used to set the output voltage between 1.5 and 5 V in steps of 0.05 V. They are processed using a library from Jack Christensen [3], slightly modified, which debounces them and gives an acceleration of the rate of increase or decrease during a longer press.
- **The 1-ms interrupts (1 kHz)**, the time between each measurement. During the initial tests, the system was subjected to a false load made with a sine wave signal at 1 Hz protected by resistors. Because 1 kHz is an exact multiple of 1 Hz, we expected a perfectly stable signal. However, the PC display showed a sine wave which shifted slowly. After some searching, it was found that the actual period between our interrupts was 1.024 ms. This was due to the Arduino board using the same counter to produce pulse width modulation signals (PWM [4]). To correct this, it is necessary to configure the register OCR0A to 249: 16,000,000 Hz (crystal frequency) / 64 (internal divisor) / (249+1) = 1.000 Hz.
- **The character fonts**. In addition to the libraries available on the internet, we had to create a new character font to improve the display on the OLED screen.
- **The I²C library**. Given that the project only used integrated circuits from Analog Devices (Linear Technology), we used the 'LT_I2C' library [5].

The Loop function

As its name indicates, the Loop function runs the µC in a continuous loop. This has three modes:

- **Stand-by**. This mode is activated/deactivated by S1 (furthest to the right). In this mode, S2 and S3, under the display, are used to adjust the output voltage.
- **Local**. The measurements are displayed on the OLED screen, and are not transmitted to the serial port. The reception of an 'E' character on the serial port will activate this mode (the PC program is responsible for this).
- **Remote**. The measurements are transmitted to the serial port, and are not displayed on the OLED screen (which then displays 'SERIAL'). The reception of a 'B' character on the serial port will activate this mode (the PC program is responsible for this).

Integrated diagnostics

If one of the buttons is pressed during the start-up of the Arduino, a small program to test the shield is run. The user needs to connect a 100 kΩ (0.1%) resistor, then a 1 kΩ (0.1%) resistor on the output and follow the instructions on the OLED screen.
It is also possible to calibrate the shield. To start this procedure, buttons S2 and S3 ('Down' and 'Up') must be pressed during the startup of the Arduino. Connect a 1-MΩ, 0.1% resistor to the output and follow the instructions on the display.

PC Application

As the MicroSupply sends its measurements at a rate of 1 kHz, the PC should be able to display them in real time. The author, who programmes primarily in C#, first looked for a solution using ChartControl, but the refresh rate and especially the automatic scaling were too slow. After a lot of research, she discovered the Nebula library. This library responds well to our needs, but needs development in Eclipse/Java. First tests were concluded, apart from the serial communication with our Arduino. So what to do to avoid opting for a costly solution? Well, use IKVM [6], a a virtual Java machine for .NET (Visual Studio) runtime environments. The IKVM.NET tool (developed by Jeroen Frijters) can migrate an existing Java database to .NET. As Elektor is not an IT magazine, we'll spare you the details of this formidable tool, but to summarise, it produces a DLL written in Java (*DLL MicroSupply_JA.dll*) which is easy to use with C# in Visual Studio. Our PC application thus uses two different programming languages.
The application stores 4000 measurements, that's 4 s, which can be displayed in slices of 1, 2 or 4 s. Note: The application only displays 1000 points, so for the 2 et 4 s choices some measurements do not appear, which may entail an error in the analysis of the workings of your connected device.

Windows 10 64-bit required

For the programme to run on your PC you will need Windows 10, 64-bit and up to date, plus Java (64-bit) version 8, update 191 or better. Download the archive file [1] or [2] and expand it in a directory of your choice. You will find the following:

- The programme *MicroSupply.exe*
- The two DLLs *IKVM.OpenJDK.Core.dll* and *ikvm-native-win32-x64.dll*
- The DLLS for the SWT function (directory 'filesSWT' from download [1]) with directory names '.swt\lib\win32\x86_64' should be copied into directory 'C:\Users\YourUserName'.
- The DLL *MicroSupply_JA.dll* (it's not necessary to copy it, as it is integrated in the programme MicroSupply.exe with numerous other IKVM DLLs).

The first time, using a command line, it is advisable to run the programme with the argument '*debug*'. This will display a lot more information if you encounter any problems with the configuration of your PC for running the programme.

User Manual

Here is a quick guide to what the various buttons and cursors of the PC application are used for:

- **Refresh**: refresh PC serial port list.
- **Max Value and Min Value**: values of maximum and minimum current measured during the second displayed.
- **1 Second, 2 Seconds and 4 Seconds**: display mode, display data for 1, 2 or 4 s.

- **Cursor Long (range 0 to 9)**: choose a period of data to display (max. 4 s). In mode 1 Second in steps of 0.25 s; in mode 2 Seconds, the step is 0.5 s.
- **Cursor Short (range 0 to 1 or 0 to 3)**: select the measurements from memory. In mode 2 Seconds, the cursor selects to extract one of 2 measurements, either the first or the second. In mode 4 Seconds, it's one of four measurements: the first, the second, the third or the fourth.
- **Save**: save all the measurements in a CSV file (values in µA), up to a maximum of 4000 measurements.

The LEDs and Pushbuttons

- LED1 (green): The connected device is powered.
- LED2 (red): The software circuit breaker has been tripped, the connected device is no longer powered.
- LED3 (Yellow): Flashes if the connected device is powered. In case of programme malfunction, it will remain lit or unlit.
- LED4 (blue): The shield is powered from external power. This does not necessarily mean that the external power is sufficient.
- S1: Activation / deactivation of the output.
- S2: Up / Next / Yes.
- S3: Down / No.
- S4: Reset.

Conclusion

The MicroSupply is capable of powering a load at 5 V maximum and supplying a maximum current of 40 mA. At the same time, it measures the current taken by the load and displays the value on its OLED screen. The captured data may also be transmitted on a serial port, to a PC for example. If the receiver of the data is a PC running Windows 10, a specially developed application allows the data to be displayed in the form of a graph. Here then is a very practical little tool to verify the power consumption of your connected devices (now or in the future).

(170464-02)

Web Links

[1] Article support page : www.elektormagazine.com/170464-02

[2] Source code : https://github.com/jenniferaubinais/MicroSupply

[3] Button Library: https://github.com/JChristensen/Button

[4] Book 'Mastering Microcontrollers Helped by Arduino ' : www.elektor.com/mastering-microcontrollers-helped-by-arduino-edition-3

[5] LT_I2C & Linduino : https://github.com/analogdevicesinc/Linduino

[6] IKVM : www.ikvm.net/

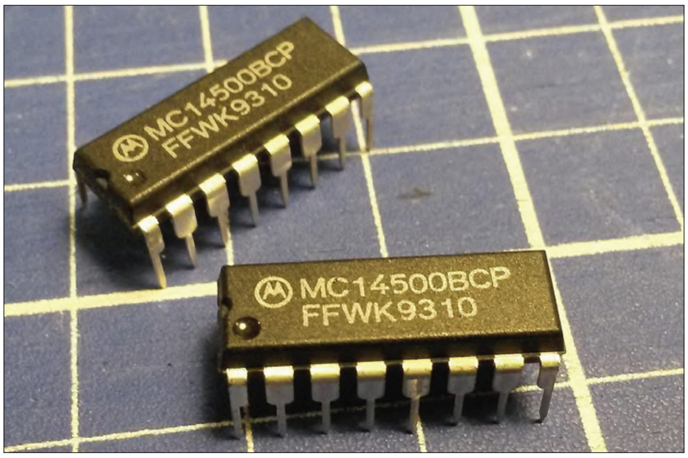
Motorola MC14500 Industrial Control Unit

Peculiar Parts, the series

By **Neil Gruending** (Canada)

The relay logic used to control industrial machinery was eventually replaced with discrete digital logic but it was still difficult to modify and maintain. A programmable solution was needed but late 1970's computers were expensive and complex so Motorola introduced the MC14500 industrial control unit. It has an interesting 1-bit architecture that is worthy of being a *Peculiar Part*.
Released in 1977, the MC14500 was designed to implement the ladder logic that was used in programmable logic controllers (PLC). It's an interesting design because it only contains enough logic to implement 16 instructions with a 1-bit register as shown in **Figure 1** [1]. The data line is either an input or an output depending on the instruction being executed and there are dedicated JMP/RTN lines for jumping over instructions. There are also instructions for basic logic calculations between the data register and the data input/output. Unfortunately there aren't any numeric instructions since they need more than 1 bit of storage. But what about the missing program counter and memory?
Well that's because the MC14500 requires an external program counter and memory just like in **Figure 2** [1]. The counter controls the active memory address which contains the instruction and other control signals for things like data latches like in the figure depending on what the application requires. The latches allow the MC14500 to calculate as many inputs and outputs as necessary 1 bit at a time.
The block diagram in Figure 2 should look familiar as it's all of the basic building blocks for a simple discrete computer. In fact, the DATANorf Hard and Software company in Germany made and sold the WDR-1 training educational kit that used a MC14500 as simple CPU [2]. The kit added LEDs for the outputs and had a keyboard to enter instructions. It's even possible to implement a MC14500 using standard logic like Dieter Mueller did [3].
The MC14500 was manufactured by Motorola for about 20 years (!) and there's a lot of information available about it including the datasheet and applications handbook. It shouldn't be too hard to find one in the used market, or maybe you want to try implementing it yourself in programmable logic. Either way I think a 1-bit processor is definitely a *Peculiar Part*! ⏮

180575-E-01



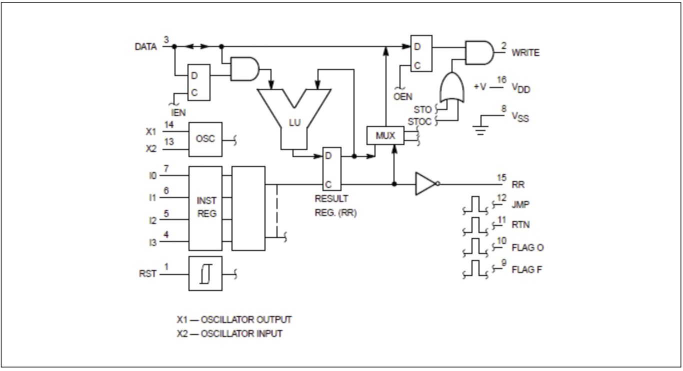


Figure 1: MC14500 block diagram [1].

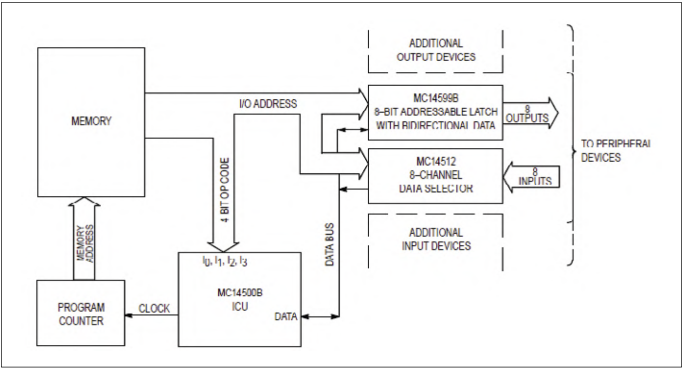


Figure 2: Overall MC14500 design [1].

Web Links

[1] MC14500 diagrams: www.brouhaha.com/~eric/retrocomputing/motorola/mc14500b/mc14500brev3.pdf

[2] WDR-1 training educational kit: www.old-computers.com/museum/computer.asp?st=1&c=834

[3] Dieter Mueller's website: www.6502.org/users/dieter/m14500/m14500.htm

Hardware Design using (V)HDL (4)

Reaction time game

By **Jörg Zollmann** (Germany)

In the previous instalment of this series [1] we saw how to drive an 8-by-8 LED matrix to display the time. In this article we will be using a seven-segment display instead, and rather than making a simple clock we will look at the rather more complex process sequencing involved in implementing a reaction time game in VHDL.

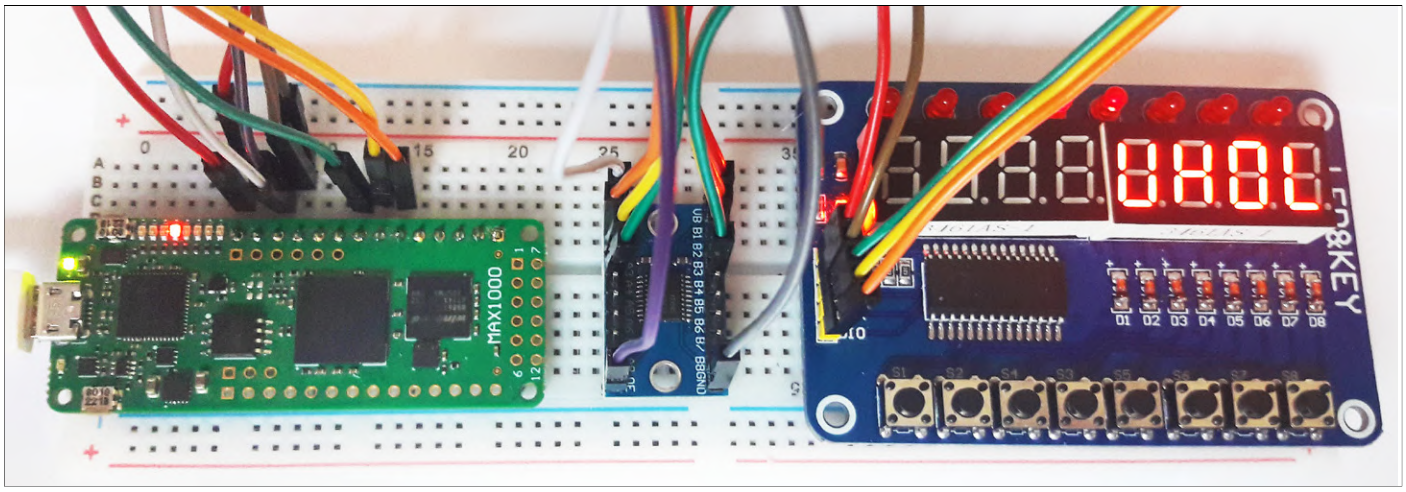


Figure 1. Assembly of the reaction timer game on a breadboard.

As always the source code for the project can be downloaded from the Elektor Magazine website, in this case, at [2]. The hardware supporting this VHDL demo will be either the CPLD board that will be familiar from the previous instalment or the MAX1000 platform from the SCCC project (see the **@ www.elektor.com text box**). We will also be using an expansion board carrying a TM1638 driver for the eight 7-segment LED displays, the displays themselves, and eight pushbuttons [3] (see **Figure 1**). Communication between the FPGA and the TM1638 uses a serial protocol; and because the FPGA uses 3.3-V logic levels and the TM1638 expects 5-V signals, a TXS0108E level shifter also forms part of the design (see **Figure 2**).

Game construction

We will describe the hardware elements required for the reaction time game one by one and briefly outline how to go about doing a timing analysis, an essential part of digital circuit design that we have not previously covered.

The game consists of several functional blocks which are described in various entities and packages. The functions are the overall game sequence controller, a running light, the display controller including the SPI bus master driver, a random number generator, a delay timer and a stopwatch. The game is started by pressing KEY0 on the expansion board. This causes the running LED light on the board to start. After a random time period the running light stops and the reaction time measurement begins. The player must now as quickly as possible press the button in the position where the running light has stopped. So, if for example the running light stops with LED3 lit continuously, the player has to press KEY3 to stop the time measurement. Button KEY0 can now be pressed again to have another go.

Stopwatch and running light

The stopwatch logic consists of several counters of the sort familiar from previous parts of this VHDL series. The configuration of the counter is specified in the package `count_pkg`. One counter is configured to act as a prescaler: note that different prescale factors are required depending on which board you are using. After the prescaler comes a counter for hundredths of a second (`csec`), one for tenths of a second (`dsec`) and two seconds counters. Since we have to display the counter states from the stopwatch logic on the seven-segment displays, it makes sense to use BCD counters here. Each counter is thus responsible for one of the significant digits required for the display, and is configured to count from 0 to 9. And when, for

example, the hundredths of a second counter counts past 9, the overflow signal that is generated is used to enable the clock to the tenths of a second counter, causing it to increment by 1. If the running light is to be clearly visible it is important that it should not change state too fast or too slowly. The solution used here is to arrange for the running light to change states every 10 ms, and the 'tick' enable signal required to implement this is taken from the stopwatch logic via the higher-level part of the design as signal `t10ms`. The `led_chaser` process is responsible for implementing the running light function itself (see **Listing 1**). This process makes use of variables.
VHDL variables can always be used when a signal is only used within a single process. Assignments to variables are expressed using just '=' rather than the '`<=`' used for signals. Variables are defined always to take on their newly-assigned value immediately: this only happens with signals when they are expressed in terms of purely combinatorial logic; otherwise signals only take on their new values at the end of a clock cycle.
The running light process stores the number of the LED that is currently lit in the unsigned signal `led_r`. The variables `dir` and `pos_led` determine whether the next LED that should be lit is to the left or the right of the currently-lit LED. In order to see the difference between a variable and a signal, it is instructive to experiment here with the effect of replacing the variables with signals.

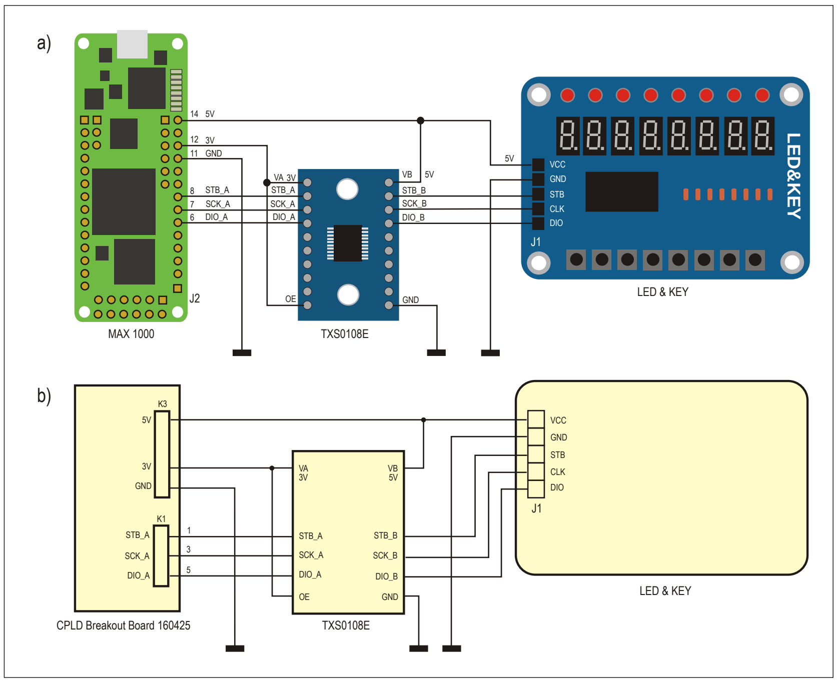


Figure 2. A level-shifter converts the 3.3 V signals of the FPGA to 5 V.

Random number generator

In order to ensure that the running light does not always come to a halt in the same position, the duration of the effect is chosen at random. There are two parts to this. The first is a delay counter, which is told whenever a new game starts how far it should count. The second part is a random number generator. When a new reaction time measurement is started the current value from the random number generator is loaded into the counter as its initial value and its reset signal is deasserted. In contrast the the counters we have looked at before, this implementation has only a synchronous reset, and the entity also has an input port named `reset_val`. The random value itself is created using a PRBS (pseudo-random binary sequence) generator. This consists of a seven-stage shift register with feedback (see **Listing 2**). The shift register continuously shifts data one stage at a time from its input (which is at bit 0). This input is obtained from the feedback logic, which is an exclusive-or gate connected to the MSB of the shift register (bit 6) and to bit 5. The pattern of connections to the gate is derived from what is called the 'generator polynomial'; depending on the choice of polynomial the PRBS generator can generate a sequence of up to $2n-1$ values, where n is the length of the shift register. In our reaction time game 127 different delay times are possible.
At this point we also have the opportunity to see how a shift register can be implemented. In one version we use two assignments, the first to explicitly store the 'old' values and the second to process the new value entering the shift register. This is the method used in Listing 2 as well as in the SPI master driver code. VHDL offers an alternative approach, however, implementing a shift register with the help of functions in the `IEEE.numeric_std` library, which defines the functions `shift_left`

Listing 1. Running light.

```
led_chaser: process (rst, clk)
  variable led_pos : unsigned(3 downto 0);
  variable dir      : bit;
begin
  if (rst = '1') then
    led_r                <= 8B"0000_0001";
    led_pos              := (others => '0');
    dir                  := '0';
  elsif rising_edge(clk) then
    if (t10ms) then
      if (run_chaser = '1') then
        if led_pos = 7 then
          dir := '1';
        elsif led_pos = 0 then
          dir := '0';
        end if;
        if dir = '0' then
          led_r    <= shift_
 left(unsigned(led_r), 1);
          led_pos  := led_pos +1;
        else
          led_r    <= shift_
 right(unsigned(led_r), 1);
          led_pos  := led_pos -1;
        end if;
      end if;
    end if;
  end if;
end process;
```

and shift_right. These functions can be passed a signed or unsigned value as their first parameter; the second parameter specifies the (integer) number of places by which the value is to be shifted. Depending on whether the first parameter is signed or unsigned, the function implements a logical shift (filling vacated bits with zero) or an arithmetic shift (sign-extending into vacated bits).

Listing 2. PRBS generator.

```
  prbs_lfsr: process (rst, clk)
  begin
    if (rst = '1') then
      d(6 downto 1) <= (others => '0');
      d(0)          <= '1';
    elsif rising_edge(clk) then
      d(6 downto 1)   <= d(5 downto 0);
      d(0)            <= d(5) xor d(6);
    end if;
  end process;
```

Listing 3. The rdata2slv function.

```
  function rdata2slv(
    rd : in std_logic_vector(31 downto 0)
  )
  return std_logic_vector is
    variable v_keys   : std_logic_vector(7 downto
   0);
  begin
    v_keys := rd(17)& rd(21) & rd(25)
   & rd(29) & rd(19) & rd(23)& rd(27)
   &                   rd(31);
    return std_logic_vector(v_keys);
  end;
```

Listing 4. SPI clock generation.

```
-- f_ena_int = 2 x f_sck
-- use sck_r1 in non delayed version -> CHPA = 0
-- use sck_r2 to delay sck --> CHPA = 1
-- use CPOL=1 to invert sck (switchable inverter)

  sck  <= ((sck_r1 and not CPHA) or (sck_r2 and
   CPHA)) xor CPOL;
  sck_proc : process (clk,rst) is
  begin
   if (rst = '1') then
     sck_r1 <= '0';
     sck_r2 <= '0';
   elsif rising_edge(clk) then
     if (ena_int) then
       if (frame_enable)then
         sck_r1  <= (not sck_r1);
         sck_r2  <= (sck_r1);
       end if;
     end if;
   end if;
  end process sck_proc;
```

The LED driver IC

The 'LED&KEY' board used in this example employs a type TM1638 driver device, and is available from Amazon and from far-eastern suppliers for a few pounds. The device originates from the Chinese company Titan Micro, which offers a range of LCD and LED display driver ICs. The TM1638 data sheet is not always a perfect model of clarity, owing in part to the lack of a good-quality translation from the original Chinese. However, since the board is very popular among the Arduino set, there are fortunately many blogs to be found on the Internet explaining various functions of the board rather more clearly: see [4], [5] and [6], for example. The TM1638 has 16 internal 8- and 2-bit writeable registers, as well as four further single-byte registers whose contents can be read back. In order to light up one of the LEDs or a seven-segment display it is first of all necessary to send a separate command to configure the brightness level, and then set up the mode for addressing the internal registers. Then you can write the desired value to the register and finally turn the display on.
The values stored in the sixteen writeable registers are continuously output by the TM1638 to the displays in a time-multiplex arrangement. The sixteen registers are organized into pairs, each pair comprising an 8-bit and a 2-bit register. During the display output phase the register pairs are addressed in turn and the values in the currently-active register pair are output on the segment output lines. At the same time the 'grid' pin corresponding to that register pair is pulled low. The duration of the pulse on the grid pin determines the brightness of the connected LEDs and, as mentioned above, can be separately configured. When one cycle of the output phase is completed the key scan phase begins, when the states of the connected buttons are read. The buttons are wired in a matrix between the SEG output lines of the chip and its 'K' inputs via protection diodes. If the chip detects that a button is pressed during the key scan phase then the corresponding bit of one of the four readable registers will be set to 1.

Functions

The LED&KEY board is driven using a simple state machine in the process called ctrl_fsm. The state machine passes commands, which are declared as constants in the package tm1638_pkg, to the lower-level hardware driver (SPI bus master) layer. The first step is to switch the TM1638's operating mode to direct addressing, which allows us to configure the display brightness. Then we switch to addressing the individual segments and send out the correct value to turn the LEDs on or off in sequence: this in turn depends on the value of the signal led_r, whose state is changed by the running light process. Finally the key scan registers are read out from the device and the values saved in the internal signal rdata. The function rdata2slv is used to determine which button has been pressed from the data in the four bytes returned by the TM1638. The function f_bcd_2_segment is called to convert the BCD counter values in the stopwatch logic into seven-segment representations. As in other programming languages, the use of VHDL functions in situations like this helps to promote the reuse of code fragments and leads to more compact and readable code. Like in C, functions are called by giving their name followed by a pair round brackets enclosing their (optional) parameters. Functions can only return a single value. The line

```
key_r <= unsigned(rdata2slv(rdata));
```

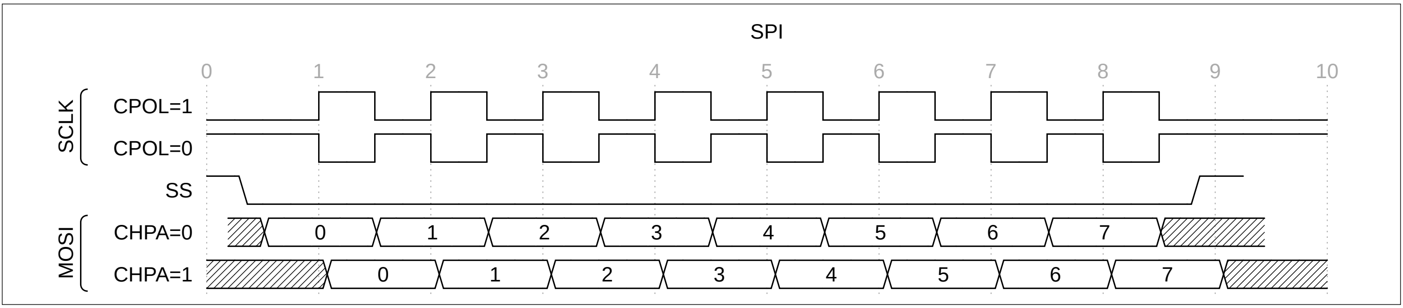


Figure 3. SPI timing.

calls the function rdat2slv (see **Listing 3**), converts the returned result (a std_logic_vector) into an unsigned value and assigns this value to key_r. The conversion to an unsigned value is necessary so that a comparison is possible with the current LED value, which is held in the unsigned signal led_r. Both std_logic_vector and unsigned are 'unconstrained arrays' of type std_logic. An equally satisfactory alternative approach to allow this comparison would have been to convert led_r to a std_logic_vector with the help of the function call std_logic_vector(led_r).

SPI bus master

The TM1638 is controlled using three signals: a clock line, a strobe line, and a bidirectional data line. A glance at the data sheet for the TM1638 shows that the communications protocol is essentially the same as SPI, with the exception of the possibility of transferring data in either direction on the data line. From a technical point of view the SPI standard is very straightforward. It only specifies how data are transferred at the lowest level, as the exchange of bytes between a master device and a slave device; the standard does not include an specification of signal voltage levels or other electrical characteristics. Normally the interface includes two control signals, SCK and SS, and two data signals, MOSI (master out slave in) and MISO (master in slave out). However, there are some devices, including the TM1638, that combine the MOSI and MISO data lines into a single bidirectional open-drain signal. SPI is a synchronous serial protocol, which means that data transfer is under the control of the clock signal SCK, generated by the master. As a result of the looseness of the specification there are four 'modes' of operation, which differ in the quiescent state of the SCK signal and in its phase relationship to the data signals (see **Figure 3**). The information in the TM1638 data sheet implies that the mode with CPHA=1 and CPOL=0 is the one required.

The SPI bus master implementation in this project is very generic and can therefore be reused in many other projects. It consists of four main functional blocks. The frequency divider in the ckdiv_proc process divides down the system clock to generate two enable pulses per SPI bus clock. The SCK generation logic and the shift register finite state machine (ctrl_fsm) are triggered by these enable pulses. **Figure 4** (SPI_master_ctrl_fsm) shows the state diagram for ctrl_fsm. The main job of the state machine is to control two shift registers: on each enable pulse the state machine either causes the transmit shift register to output a new bit on the output data line, or causes the receive shift register to shift in a new bit from the input data line. The total number of bytes that will be transmitted is determined by the input signal frame_bytes, and the signal first_rx_byte determines how many bytes will be processed before switching the data direction from transmit mode to receive mode. The SCK signal generation is done in the process sck_proc (see **Listing 4**). The process describes a toggle flip-flop (sck_r1) whose state is inverted on each enable pulse, and a further flip-flop (sck_r2), which stores the state of the toggle flip-flop and so generates a clock delayed by half a period. The SPI clock signal at the output of the SPI bus master driver is either the output of sck_r1 (if CPHA=0) or that of sck_r2

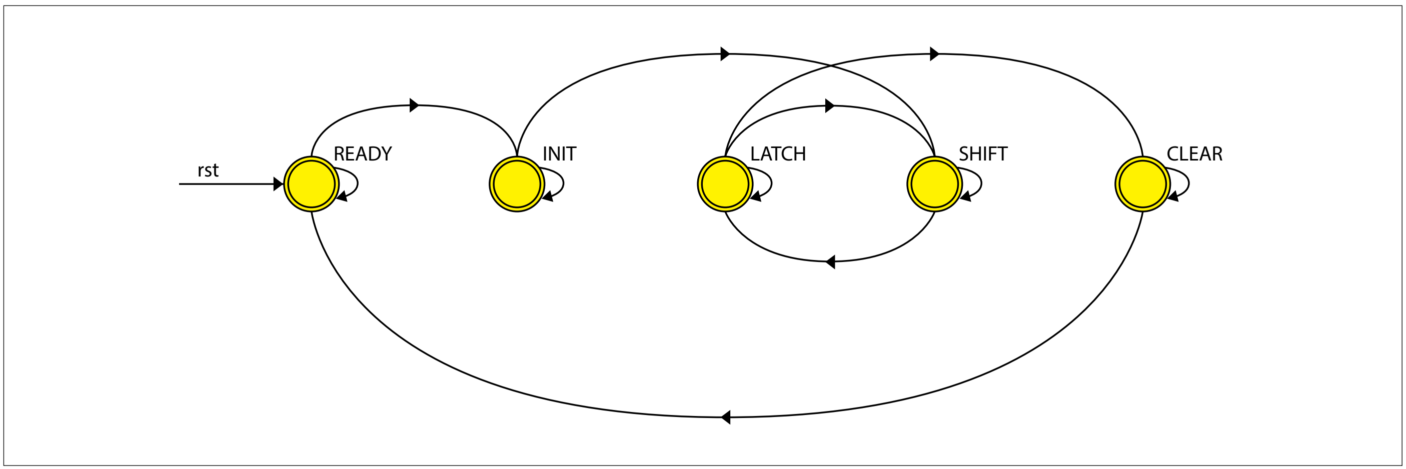


Figure 4. State diagram for the SPI controller state machine.

(if CPHA=1). The signal CPOL then determines if the chosen output is subsequently inverted. The last remaining important functional block is a process (doe_proc) that controls the switching of the function of the data line. This process uses the internal signal d_oe to make sure that, depending on the current state of the state machine and on the number of bits that have so far been transmitted, the direction of the DIO pin is switched at the correct moment (see **Listing 5**).

Listing 4. SPI clock generation.

```
-- f_ena_int = 2 x f_sck
-- use sck_r1 in non delayed version -> CHPA = 0
-- use sck_r2 to delay sck --> CHPA = 1
-- use CPOL=1 to invert sck (switchable inverter)

  sck  <= ((sck_r1 and not CPHA) or (sck_r2 and
   CPHA)) xor CPOL;
  sck_proc : process (clk,rst) is
  begin
   if (rst = '1') then
     sck_r1 <= '0';
     sck_r2 <= '0';
   elsif rising_edge(clk) then
     if (ena_int) then
       if (frame_enable)then
         sck_r1  <= (not sck_r1);
         sck_r2  <= (sck_r1);
       end if;
     end if;
   end if;
  end process sck_proc;
```

Listing 5. DIO direction control.

```
-------------------------------------------------
-- Output enable generation for DIO
-------------------------------------------------
din    <= dio  when d_oe = '0' else '0';
dio    <= dout when d_oe = '1' else 'Z';

doe_proc : process (clk,rst) is
variable rx_bit : integer := 0;
variable send_bits : integer := 0;
begin
 if (rst = '1') then
   d_oe      <= '0';
 elsif rising_edge(clk) then
   rx_bit    := to_integer(unsigned(first_rx_
   byte)& "000");
   send_bits := to_integer(unsigned(frame_bytes)&
   "000");
   if (ena_int) then
      if (fsm_state = READY) then
        d_oe      <= '0';
      elsif (fsm_state = INIT) then
        d_oe      <= '1';
      elsif (to_integer(send_bits-bit_cnt) = rx_
   bit and (fsm_state = SHIFT)) then
        d_oe      <= '0';
      end if;
   end if;
 end if;
end process doe_proc;
```

TimeQuest timing analysis

For Quartus to be able to create a correctly-operating digital circuit from VHDL source code, the tool needs not only the functional description as supplied by the VHDL but also various extra goals for the design. These are stored in a separate *.sdc* file: SDC stands for 'Synopsys Design Constraints' and is an industry standard based on Tcl. The most important constraints stored in this file relate to timing: note that pin allocation constraints in Quartus are stored in a *.qsf* file. **Figure 4** shows a simplified example that illustrates a signal within our design and its corresponding timing diagram. FF1 and FF2 might here be two registers forming part of the state storage for the state machine that controls the sequence of events in our game. During the 'place and route' phase of compiling the design Quartus must assign locations to the flip-flops and connect them up in such a way to ensure that a data change at the output of the first flip-flop (called the 'launch edge') arrives in time to be captured correctly by the second flip-flop on the next clock (the 'latch edge'). If there is a setup time violation ('negative slack', see below) then that means that the propagation delay of the signal might be too great and so the second flip-flop will capture the previous output of the first flip-flop rather than the new data. Conversely, a hold time violation means that data that should be captured on the next clock edge arrive so quickly relative to the clock that they are captured on the same edge as they are generated. Also, setup and hold times can be simultaneously violated, which means that the second flip-flop might end up in an undefined or metastable state [8]. Setup violations can only be avoided by making the clock frequency sufficiently low in relation to the data path propagation delays, or in other words when

$t_{clk} < t_{su} + t_h + t_{pd} + t_{co}$

holds, where

t_{su} (setup time) is the time for which the input data to a flip-flop must be stable before an active clock edge;

t_{pd} (propagation, or sometimes path, delay) is the time taken for the signal to get from A to B;

t_{co} (clock to output delay) is the time from an active edge on the clock input to a flip-flop to a change of state on its output; and

t_h (hold time) is the time for which the input data to a flip-flop must remain stable after an active clock edge.

To perform a 'setup timing check' the timing analysis tool must compute the time required for a signal to get from one flip-flip to the next for each possible internal path (the 'data arrival path'). This calculation takes into account both the propagation delay inherent in the chosen connection path and the delay to the clock signal (the 'clock arrival path'). The value that describes the extent to which the above condition is fulfilled is called the 'slack' (or, in this case, 'setup slack'). The value is positive when the condition is fulfilled and negative when there is a timing violation. To tell Quartus the anticipated clock frequency of the design, use the following command in the *.sdc* file.

```
create_clock -name  clk  -period  83
   [get_ports {clk12m}]
```

In this example we create a 'base clock' with a period of 83 ns, and specify that the name of the clock signal in the design is `clk12m`. This is appropriate only for the MAX1000 board; for the CPLD board the `period` value should be reduced to 25. This first step is enough to let us work with the internal logic of the device. The next step is to add constraints on the SPI inputs and outputs: in other words, to tell the tool that a certain timing relationship must exist between the SPI strobe, data and clock signals, and that this relationship must be preserved by the routing in the FPGA or CPLD.

To that end we define a further clock signal called `spi_clk` as follows:

```
create_clock -name  spi_clk -period  $spi_perr
   [get_ports {sck}]
```

Because the .sdc file format is based on Tcl, variables can be defined using the `set` keyword and their value can be accessed using a reference starting with `$`; as usual it is a good idea to use meaningful variable names. The next thing we need to specify is the time delay introduced by external circuitry connected to the input and output paths. This is done using the `set_output_delay` and `set_input_delay` commands. You can find out whether Quartus is happy with your specified timing constraints in the Compilation Report Window under TimeQuest Timing Analyzer, or directly in the timing analysis tool, which can be started with a click on Tools -> TimeQuest Timing Analyzer. We do not have space here to detail all the constraints that it is possible to specify: the ones relevant to this project can be found in the file *reactionTimer.sdc*. An important thing to take away from this exercise is that even when you have written your VHDL code you are still a long way from having a complete working circuit. There is a wealth of further information on timing constraints on the Internet, for example at [7], [9] and [10].

Conclusion

The reaction time game we have described in this part of our VHDL series is a good example to illustrate the advantages and disadvantages of using a hardware programming language. If we were to implement the game using an Arduino sketch surely only fifty or so lines of code would be needed. But fifty lines does not get you very far in VHDL, and in total this project runs to over 1200 lines. And, if one wanted to rustle up a quick and dirty measurement and control application, CPLDs and FPGAs are unlikely to be one's first choice. On the other hand, if the technical demands of a project include high-speed input and output, or deterministic behaviour, or low latencies, or a large number of parallel processes, then an FPGA is just the ticket. A wonderful combination of the advantages of both worlds is offered by soft processors, such as the SCCC project in this issue. And we welcome your ideas for further projects using VHDL!

180285-D-02

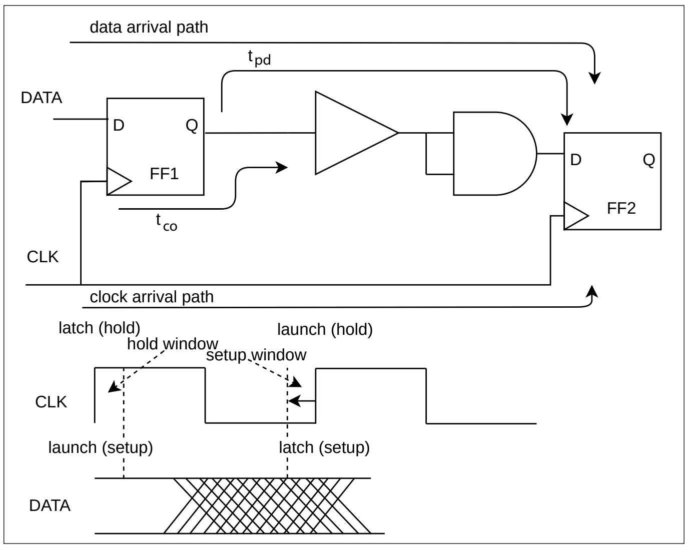


Figure 5. Example path of a signal in our design and the corresponding timing diagram.

Web Links

[1] Hardware Design using (V)HDL (3), Elektor 2/2019: www.elektormagazine.com/magazine/elektor-88/42446

[2] Article support page: www.elektormagazine.com/180285-D-01

[3] LED&KEY TM1638: www.amazon.com/s?k=led%26key+tm1638

[4] LED&KEY TM1638 demo 1: https://blog.3d-logic.com/2015/01/10/using-a-tm1638-based-board-with-arduino/

[5] LED&KEY TM1638 demo 2: www.reuk.co.uk/wordpress/arduino-code-for-displaying-numbers-on-tm1638-module-display/

[6] LED&KEY TM1638 demo 3: http://arduinolearning.com/learning/basics/arduino-tm1638-module.php

[7] TimeQuest User Guide: https://fpgawiki.intel.com/wiki/TimeQuest_User_Guide

[8] White Paper (Altera): Understanding Metastability in FPGAs: www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/wp/wp-01082-quartus-ii-metastability.pdf

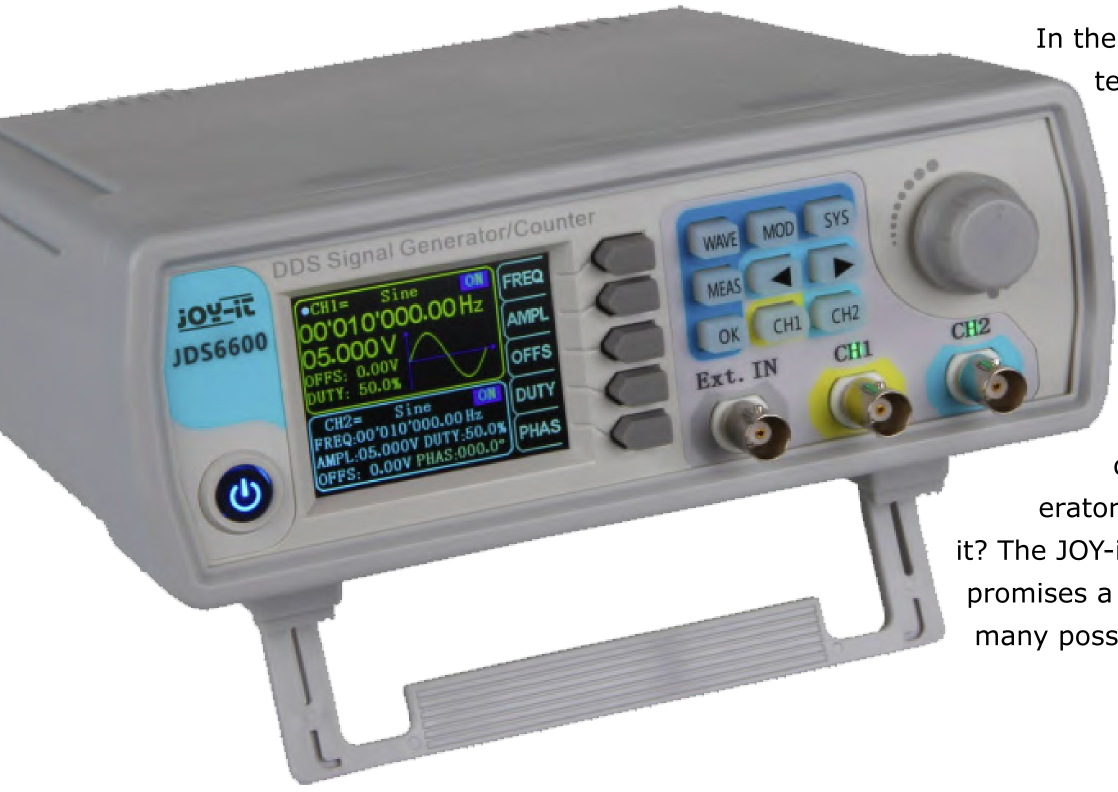
[9] Eli Billauer: Quartus Timing Analysis on set_input_delay and set_output_delay constraints: http://billauer.co.il/blog/2017/04/io-timing-quartus-calculation/

[10] Intel FPGA Wiki: Constrain SPI Core: https://fpgawiki.intel.com/wiki/Constrain_SPI_Core

JOY-iT JDS6600 DDS Signal Generator

lots of signal for little money

By **Harry Baggen** (Elektor Labs)



In the past, every electronics technician in his (hobby) lab had a function generator with an XR2206, but those days are long gone. Nowadays such a generator works with a DDS chip that generates frequencies digitally. Unfortunately, a decent DDS function generator isn't really cheap — or is it? The JOY-iT DDS function generator promises a wide frequency range and many possibilities at a low price.

Is there a snake in the frequency grass? We'll start with the bad news right away. And that's the casing of the JOY-iT JDS6600. It looks rather cheap, and the manufacturer could have invested more in it. But the good news is that the thing a large offers a wide frequency range for less than 140 euros, is quite accurate and has a whole zipper of settings. More on that in a moment.

Hardware

The JOY-iT generator is housed in a simple plastic box of modest dimensions. On the front there is a small but clear LC display that shows all settings, a number of operating buttons, a rotary knob and three BNC sockets, two outputs and an input for the internal frequency counter. The device can therefore provide two output signals that can be set independently of each other, or coupled. The supply voltage is provided by a included AC adapter. The box also contains two BNC cables with crocodile clips, a BNC-BNC cable and a USB cable, which is not what you would expect from a device with such a price. The main features of the JDS6600: frequency range for sine wave up to 60 MHz, for rectangle and triangle up to 15 MHz, and for the other waveforms up to 6 MHz. There is a choice of approximately 15 preprogrammed and up to 60 user-programmed waveforms. The maximum output voltage is 20 V_{pp} below 10 MHz. 10 V_{pp} (up to 30 MHz) or 5 V_{pp} (up to 60 MHz) are also available. The adjustable offset voltage has similar values. The JDS6600 can also generate signal bursts and frequency sweeps. The built-in frequency counter is suitable for signals up to 100 MHz and 2-20 V_{pp}.

Use

The controls of the JDS6600 are clearly arranged. There are four function keys next to the display, the actual function of which depends on the selected settings. The display shows the settings of both channels, the signal at the top also shows the selected signal shape. Most keys have multiple functions, which is a bit confusing in the beginning. For example, by pressing a CH button once, you can switch to the other channel to set this, while pressing once on an already selected channel will cause this channel to be switched off. A long press on the CH2 button causes channel 2 to be shown at the top of the display with the waveform. Press and hold CH1 to return to the top of channel 1. Clear? The keys work well, but when you press them you notice that the adjacent keys move along with them, that could have been solved mechanically a bit better.
The values on the display are set by first selecting a number with two cursor keys and then changing it with the dial. In frequency mode a lot of digits are displayed, which makes the

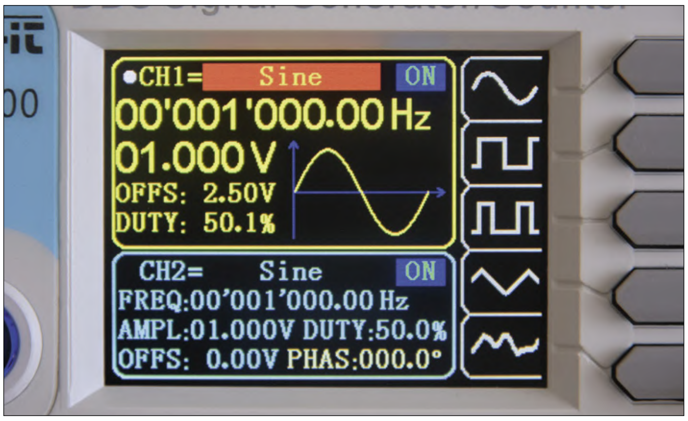


Figure 1. All settings for the two output signals are shown on the display.

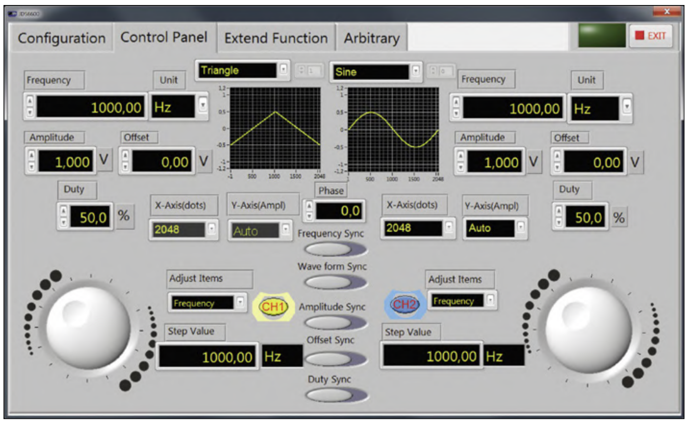


Figure 2. Control is also possible via the PC.

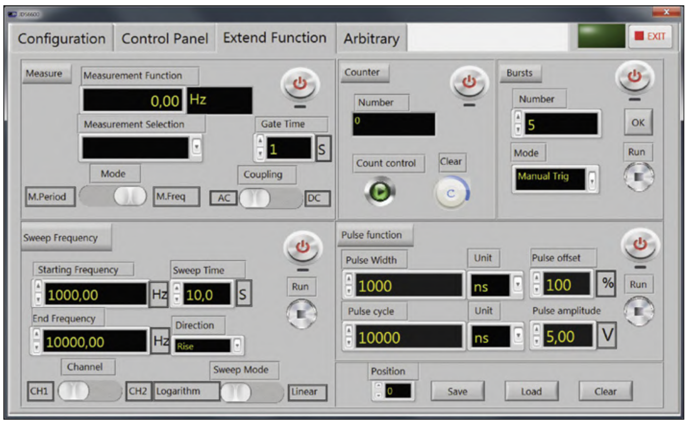


Figure 3. On this tab, all additional options are available, such as frequency sweeps.

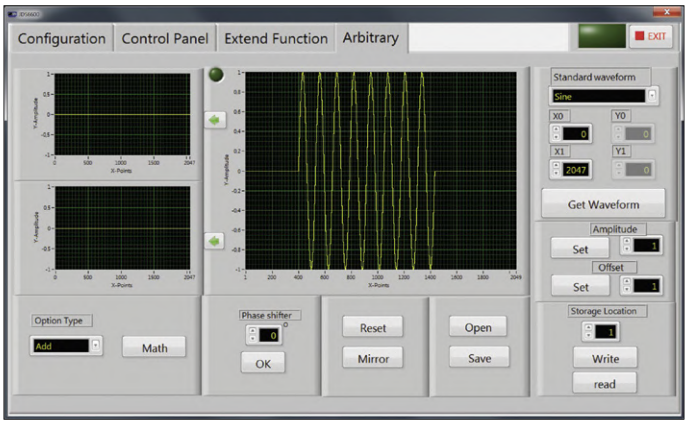


Figure 4. DIY programming a waveform is quite easy.

setting a bit difficult. The OK button switches both outputs on or off, a somewhat strange name for this function. And with the MOD button, you can set sweeps, bursts and pulse widths, but no modulation. It's just a trifle, all in all it's quite easy to work with if you're aware of these peculiarities.

The output signals look pretty good on the oscilloscope screen, although based on the specifications (sampling-rate 266 Msamples/s, waveform-length 2048 points, 14-bit resolution) I expected a little better waveforms. FFT measurements of a sine wave in the audio area showed distortion remnants of just over 1%. The manufacturer mentions less than 0.8%, so that's just not achieved with my review specimen. Square waves, on the other hand, look pretty good, with fairly steep edge and little overshoot.

The frequency response is quite linear: above 30 MHz, the output voltage increases by about 2 dB and then drops to about -1 dB at 60 MHz, not at all bad! On the JOY-iT website you can download a corresponding program [1] allowing the JDS6600 to be operated entirely from the PC via a USB connection. Like a lot of Chinese software, the design is not great, but everything works well. There are several tabs for the various functions, including one to compose your own waveforms. After trying it out for a while I made a sinewave burst for testing speakers in no time at all.

Conclusion

The JOY-iT JDS6600 has a lot to offer at a price of under 140 euros. There are a few points that can be improved, but all in all you get a lot for your money. The only thing you don't have is a modulation option, but you'll find almost everything else in signal possibilities that you occasionally need for a hobby lab. For that purpose you don't have to buy a professional device with a nice housing but a much higher price. ◀

190310-04

Web Link

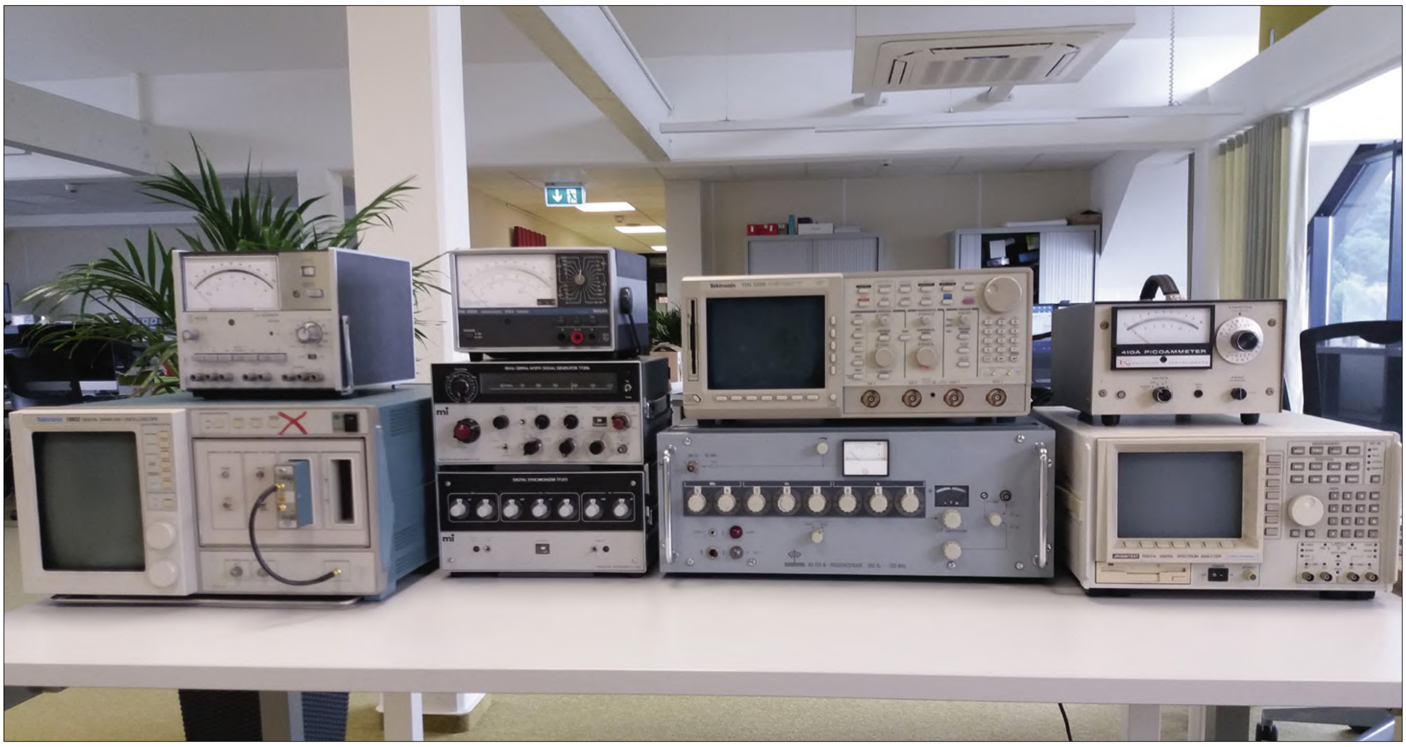
[1] Software for PC control
http://anleitung.joy-it.net/?goods=jds6600

Treasures or E-Debris?

your votes please

By **Jan Buiting** (Elektor Retronics Conservator)

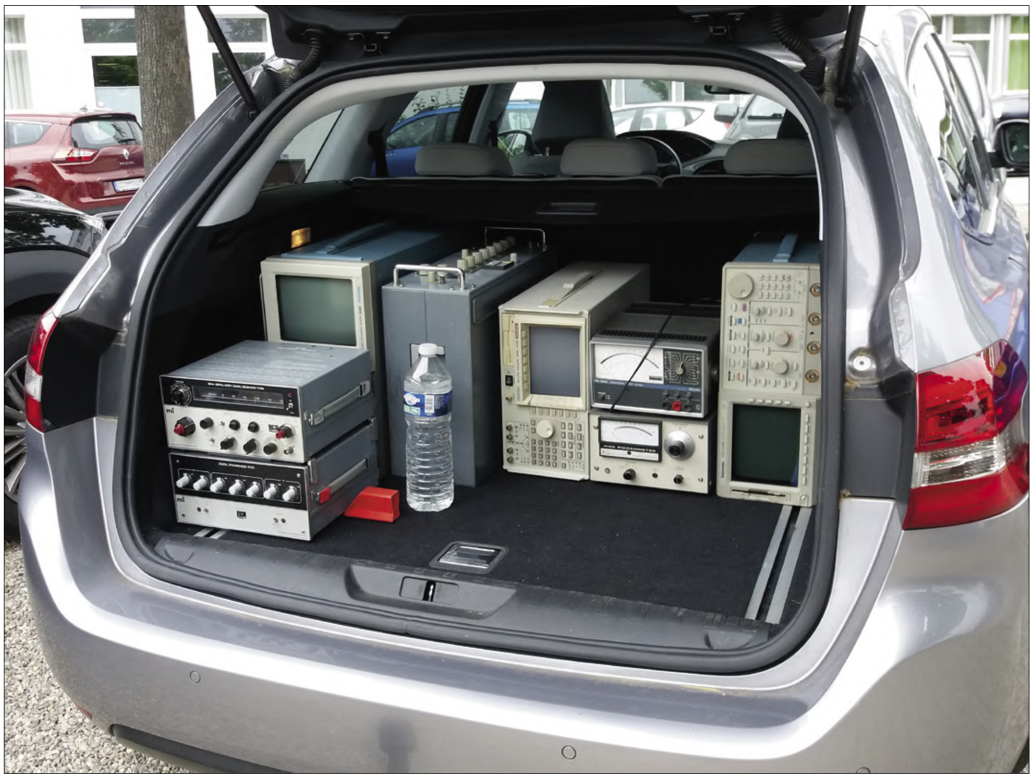
With the unstoppable move of electronic test equipment to all-digital/software/simulation and mostly plastic, top-notch gear from the old analogue and "early digitizing' periods is written off and left to gather dust in cellars. I emptied such a place recently and now ask the Retronics audience: which of the e-debris pictured here would you like to see a repair report on in the next instalment?



The equipment shown here is a random selection from a larger lot that filled the good part of the 1775-litre load space of my Peugeot 308 SW. Although on collection all equipment was said to be defective in some way, in practice it turned out some was out of calibration only. In many labs equipment calibration contracts expire after a few years, meaning the equipment is formally disqualified from scientific reporting and soon after, informally carted off to a dark place.
When flicking the on/off switch, the equipment discussed here either malfunctioned or appeared dead as a doornail. So far Retronics has had good feedback generally on boatanchor repair and restoration stories, so the aim of this instalment is to poll reader interest in a writeup on one of the instruments briefly described here. Which instrument is your favourite for an illustrated in-depth repair story? Let me know at:

- Email: jan.buiting@elektor.com;
- Subject: Treasures or E-Debris?

In your email, any thoughts or hints you might have on tackling the problems are highly appreciated.
All equipment exteriors got cleaned and a basic inspection was performed on electrical safety. No issues were found. As far as possible, documentation was obtained, and advice was sought from various experts in the field. Here come the equipment descriptions. With "educated guess" I am stating what I believe to be the top suspicious area or component in the equipment. Pile numbers: left to right. Equipment in piles: bottom to top.



Pile 1

Tektronix Type 11802 Digital Sampling Oscilloscope
This once high end and extremely expensive instrument is from the mid 1990s. This one dropped out of cal in 2008 and has one 24-GHz plug-in type SD24.
Status: powers up, screen good, self-test reports: "`error E1811, Subsys Executive`".
Educated guess: connectors; SMD electrolytics; NVRAM battery.

Philips PM 2436 DC-Micrometer
This instrument is from the famous "grey" line of PM 24xx test gear made by Philips in the early 1970s. Fantastic for sub-milliamp measurements and a standard item for decades in physics and chemistry labs.
Status: powers up, meter needle flies f.s.d. on all ranges, all modes.
Educated guess: solder joints; electrolyitics; semiconductors.

Tektronix TDS 520B Two-Channel Digitizing Oscilloscope
Status: powers up. Self-test passed successfully. Channel 2 has large distortion, AC mode equals DC.
Educated guess: capacitor in Ch. 2 amplifier, SMD electrolytics.

Pile 2

Marconi Instruments TF2173 Digital Synchronizer and TF2016 10 kHz – 120 MHz AM/FM Signal Generator
This set forms a wide range PLL, quartz-and-oven stabilized signal generator that was just affordable by larger labs in the 1980s.
Status: powers up. Initially heat, then light smoke from TF2016 rear side. System fails to lock. Output frequency unsteady, all ranges.
Educated guess: internal power supplies.

Philips PM 2504 electronic VAΩ Meter
From the 'black' PM 25xx series, this multimeter is portable, and battery operated. The supplied mains adaptor doubles as a supply and a charger.
Status: powers up. V an A readings make sense, DC and AC. Meter needle flies f.s.d. on ohm ranges.
Educated guess: unknown.

Pile 3

Schomandl ND 100 M Frequency Decade 300 Hz - 100 MHz
This heavyweight produced with German *Gründlichkei*t has an internal high stability quartz oven and promises 0.1 Hz resolution.
Status: powers up. Output frequency and amplitude unsteady. Alarm light remains off.
Educated guess: internal power supplies; no xtal activity.

Pile 4

Advantest R9211A Digital Spectrum Analyzer
This "portable" instrument covers 10 mHz (no typo) to 100 kHz for real-time FFT analysis. The -A version has a 'running zoom' function and is the one but cheapest version in the 9211 series.
Status: blows 2 A mains fuse.
Educated guess: internal power supplies. Help, no service documentation available.

Keithley Instruments 410A Picoammeter
A simple, rather popular instrument that with its pico-amps range (10^{-12} A) goes even 'lower' than the Philips PM 2436, and just scratches at those magic femto-amps out there.
Status: powers up. Meter action erratic in all ranges.
Educated guess: unknown.

180574-E-01

ESTD 2004

www.elektor.tv

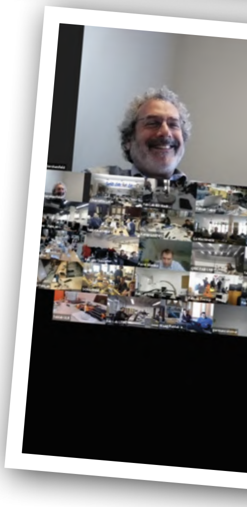
Retronics is a regular section covering vintage electronics including legendary Elektor designs. Contributions, suggestions and requests are welcome; please telegraph editor@elektor.com

ELEKTOR ETHICS

Fab Academy
a crash course in digital fabrication

By **Tessel Renzenbrink** (Elektor Ethics Correspondent)

A fully automatic cocktail mixer from France, a computer controlled calligraphy machine from Japan, a CNC machine from Peru. It's week 17 at the Fab Academy and students from around the world are showing the machines they have built in the online classroom. Each team has been allocated two minutes to present their project on the live stream. This is followed by some tips, compliments and occasionally some criticism ("it clearly needs more work") from Professor Neil Gershenfeld, the director of Fab Academy.



Everything at Fab Academy is open and can be found at https://fabacademy.org/. From the video lessons of Prof. Gershenfeld to the documentation of the projects of all students who have ever participated.Everything at the Fab Academy is open, and can be found at https://fabacademy.org/. This includes everything, from the video lessons by Professor Gershenfeld to the documentation for the projects of all the students who have ever joined.

I'm visiting the Fab Academy at De Waag in Amsterdam, The Netherlands [1]. The speed at which the lap around the world (as week 17 is called) takes place is typical of the Fab Academy [2]. In 20 weeks the students learn how to 'make (almost) anything'. Each week they learn a new skill, such as laser cutting, electronics design, embedded programming and 3D printing. And each week they have to apply that knowledge in an assignment: designing, implementing, testing, debugging and documenting. "The Fab Academy is very intense", says Henk Buursen, a mentor at the Fab Academy in Amsterdam. "We have 32 hours allocated per week, but in reality it is often more. You wake up with it and it will be on your mind until you go to bed". Sometimes it even goes beyond this: Rutger, a student, dreams about the lab in his sleep.

From Fab Lab to Fab Academy

The Fab Academy grew out of the Fab Lab (*Fab*rication *Lab*oratory). This workshop with computer-controlled tools, such as laser cutters, CNC machines and 3D printers, was established to make digital fabrication accessible to all. The idea for Fab Labs came about in 2001 at the Massachusetts Institute of Technology (MIT) in the Center for Bits and Atoms that is managed by Professor Gershenfeld. This time it wasn't a creation that escaped from the lab, it was the lab itself that escaped from the university. There are now more than a thousand Fab Labs around the world. In 2003 Professor Gershenfeld started a series of lectures called 'How To Make (almost) Anything' at the MIT campus, where students have just a single term to learn to use the instruments in the Fab Lab. In 2009 there was a trial whereby people from other Fab Labs could follow the lectures via a live stream. The result of that experiment turned into the Fab Academy.

The remote learning model, which used a combination of lecture podcasts and self-study, was gradually replaced with a new learning model: distributed learning. This facilitates the exchange of knowledge at both a global and local level. In local workshops the students learn from their mentors and from each other. Mr. Buursen and his students meet in the Amsterdam Fab Lab every Thursday to discuss their assignment and how to use the relevant machines. At a global level, the local academies are linked via interactive online lectures. There is a further global exchange of information because all groups document and share their projects online.

Open design

Am important aspect of the Fab Academy is the fact that you have to make your design available to others. This open design principle ensures that everybody in the world can learn from each other. Just as with open source software, the designs may be used, modified and improved.

Apart from facilitating an exchange of knowledge, this methodology has another purpose: it brings us closer to a production model that enables more things to be produced locally. Instead of sending container ships full of goods round the world, we can share designs via the Internet, which can then be fabricated in local Fab Labs.

Machines that make machines

"One aim of the Fab Academy is to personalise production", says Mr. Buursen. "We have become consumers and we no longer produce much anymore. The Fab Lab has a simple collection of machines that can be used to make almost anything. In the past these machines were large and expensive. The complete inventory of a Fab Lab would have cost around a million dollars in the past, but nowadays 100,000 dollars will go a long way [3]. In the future we expect this figure to get closer to 10,000 dollars. The ultimate aim is to have a Fab Lab that can create another Fab Lab, using machines to construct new machines. The Fab Academy from Peru has taken a step in the right direction, creating a CNC machine for their group project."

From consumer to producer

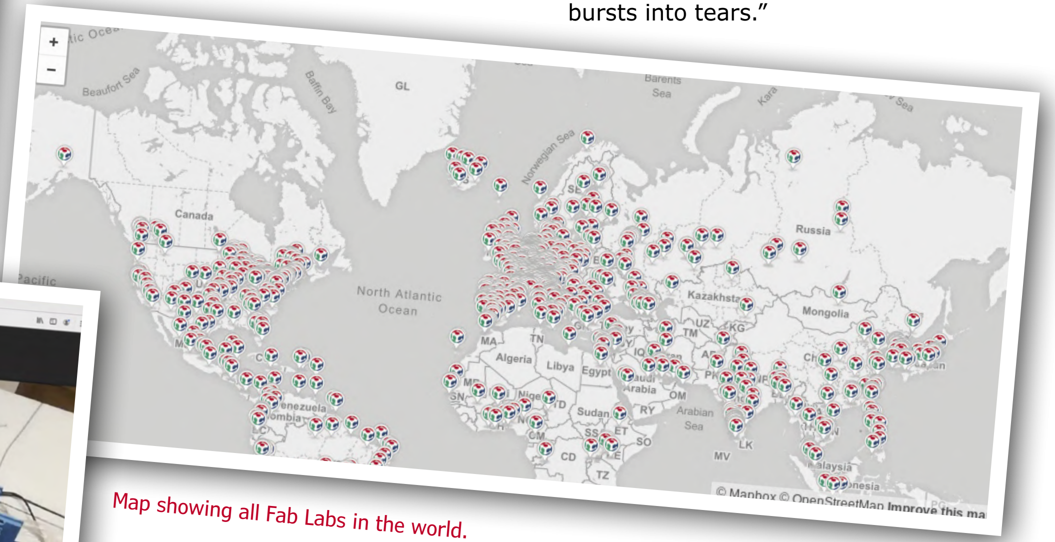
"At the Fab Academy you learn to work with the complete inventory in a course consisting of 20 modules", Mr. Buursen explains. "This results in a return to self-sufficiency. You learn that you can really make those things that are going round in your mind. This leads to an understanding of how things function. As an example, I find it important that younger students take a computer apart to give them an understanding of how it works. When everything is packaged inside a box you lose sight of what is there and what makes it tick.

But technology has an enormous influence on society. If you no longer understand how things work, you get more out of touch with the world around you. My motivation as Fab Academy mentor is to teach people how things really work. It is empowerment: giving people the knowledge and tools to understand the world around them and to make it their own.

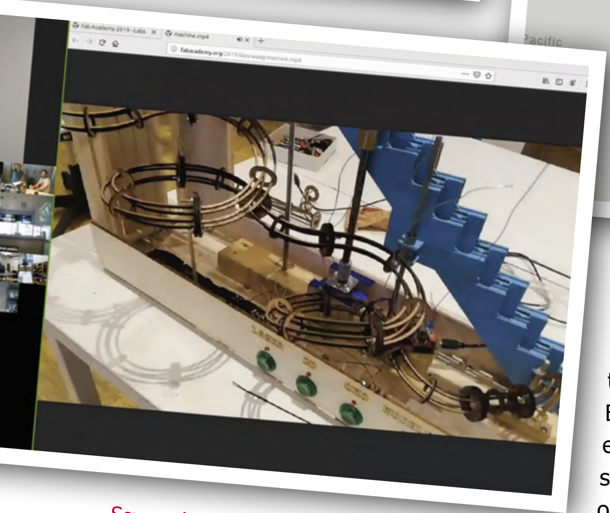
Crash course

The Fab Academy is intensive and demanding. Mr. Buursen elaborates: "I can explain to you how a machine works, or you could read a book about it. But what if it behaves differently from what you expect when you turn it on? You then have to figure out what the cause is. You have to solve problems, be inventive, and make decisions. At the Fab Academy you learn how to faultfind quickly. Because you are under a lot of pressure, this follows naturally. We all know about those projects that we started on and that didn't work properly. You put it aside for a while, and four months later it's still there. It doesn't matter what experience you have when you start at the Fab Academy. The most important aspect is that you learn new things to expand your current knowledge. And preferably exceeding even that. Everybody who joins the Fab Academy has at least one breakdown, a total panic attack, or just bursts into tears."

Rutger, a student, adds the final touches to the group project. The other students follow the lap around the world via the live stream.



Map showing all Fab Labs in the world.


Screenshot of the interactive lesson. Top left is Professor Gershenfeld. On the right is the presentation of the machine from De Waag. Bottom left are the live streams of the participating Fab Academies.

In the classroom of the Fab Academy in Amsterdam I can see that Professor Gershenfeld starts the lap around the world precisely at three o'clock: "Fab Lab Aachen, do you have a machine?" There is a sigh of relief in the classroom. "He does it in alphabetical order!", exclaims Rutger. It gives the students of Fab Academy De Waag almost two hours to put some final touches to their group project before they have to give a live demo to the whole world. ⏮

180572-E-03

Fab Academy

Everything at Fab Academy is open and can be found at https://fabacademy.org/.
From the video lessons of Prof. Gershenfeld to the documentation of the projects of all students who have ever participated.

Web Links

[1] Fab Academy at De Waag: https://waag.org/en/project/fab-academy

[2] Video of the 'lap around the world' lesson: https://vimeopro.com/academany/fab-2019/video/338245042

[3] Complete inventory of a Fab Lab: https://is.gd/XauCaK

welcome in your ONLINE STORE

EDITOR'S CHOICE

Velleman Earth Listener

The Earth Listener is a practical and nicely designed device, that will not look out of place in the living room or the office. It is completely ready for use, but also offers the possibility of tinkering with it yourself and think of your own extensions.

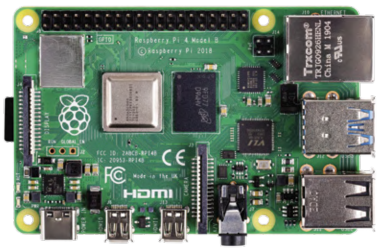
In addition to the classical measurement values of a thermometer, barometer and hygrometer, the gas sensor, with its measurement of eCO_2 and dust particles, gives a good indication of the air quality in a space and when ventilation is desired or even essential.

Luc Lemmens
(Elektor Labs)

www.elektor.com/velleman-earth-listener-kit

Elektor Bestsellers

1. Raspberry Pi 4 B
   www.elektor.com/rpi4



2. SDR Hands-on Book
   www.elektor.com/sdr-hands-on-book
3. PCBite Kit
   www.elektor.com/pcbite-kit
4. Elektor SDR Shield 2.0
   www.elektor.com/170515-91
5. Raspberry Pi Zero WH
   www.elektor.com/rpi-zero-wh
6. SmartScope USB Oscilloscope
   www.elektor.com/smartscope
7. Camera Projects Book
   www.elektor.com/
   camera-projects-book

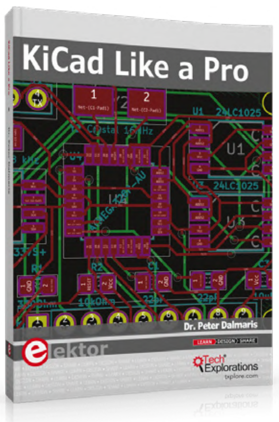
Volumio Primo

Volumio Primo runs on a software developed by the company itself. With this device, you can play music files stored on a local USB hard drive, network shares or networked media servers. The playback of music from Internet based radio and streaming services is possible as well. Depending on your preference, Volumio is capable of delivering the music in digital or analog form.

Member Price: £386.95 • €431.10 • $490.95

www.elektor.com/volumio-primo

KiCad Like a Pro

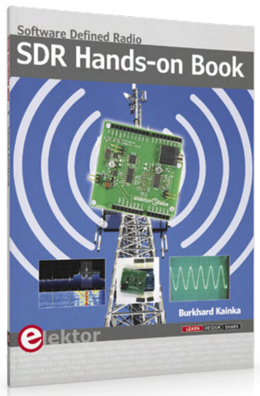


This book will teach you to use KiCad. Whether you are a hobbyist or an electronics engineer, this book will help you become productive quickly, and start designing your own boards. This book takes a practical approach to learning. It consists of four projects of incremental difficulty and recipes.

Member Price: £31.95 • €35.96 • $40.95

www.elektor.com/kicad-like-a-pro

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: £23.95 • €26.96 • $30.95

www.elektor.com/sdr-hands-on-book

NEW

Camera Projects Book

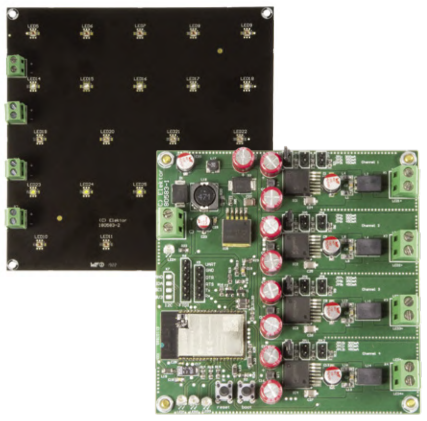
The book explains in simple terms and with tested and working example projects, how to configure and use a Raspberry Pi camera and USB based webcam in camera-based projects using a Raspberry Pi.

Example projects are given to capture images, create timelapse photography, record video, use the camera and Raspberry Pi in security and surveillance applications, post images to Twitter, record wildlife, stream live video to YouTube, use a night camera, send pictures to smartphones, face and eye detection, colour and shape recognition, number plate recognition, barcode recognition and many more.

Member Price: £23.95 • €26.96 • $30.95

www.elektor.com/camera-projects-book

Horticulture Box



Grow plants faster with this specialized horticulture LED lighting kit. The kit consists of two assembled circuit boards. An external 24 V, 50 W power supply is required to power the boards. Be careful not to look directly into the LEDs when they are on. Even at seemingly low light levels, the pulsed light intensity is very high and can damage your eyes.

Member Price: £132.95 • €148.46 • $168.95

www.elektor.com/horticulture-box

The Complete ESP32 Projects Guide

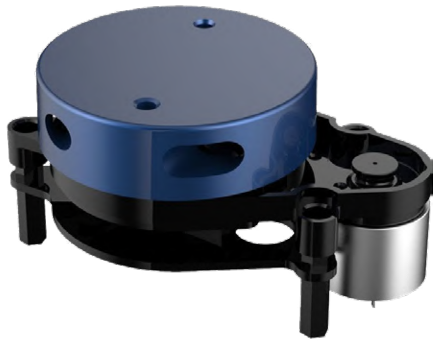
59 Experiments with Arduino IDE and Python
The Complete ESP32 Projects Guide
Dogan Ibrahim
elektor

The main aim of this book is to teach the Arduino IDE and MicroPython programming languages in ESP32 based projects, using the highly popular ESP32 DevKitC development board. Many simple, basic, and intermediate level projects are provided in the book using the Arduino IDE with ESP32 DevKitC. All projects have been tested and work. Block diagrams, circuit diagrams, and complete program listings of all projects are given with explanations.

Member Price: £31.95 • €35.96 • $40.95

www.elektor.com/esp32-projects-guide

YDLIDAR X2 – 360-degree laser range scanner



YDLIDAR X2 Lidar is a 360-degree two-dimensional distance measurement product. This product is based on the principle of triangulation, and is equipped with relevant optics, electricity, and algorithm design to realize high-frequency and high-precision distance measurement. While the distance is measured, 360 degrees of scanning distance measurement is achieved by continuously obtaining the angle information through the 360 degree rotation of the motor.

Member Price: £64.95 • €71.96 • $81.95

www.elektor.com/ydlidar-x2

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 September 16, 2019, supply your name, street address and the solution (the numbers in the gray boxes) by email to:
hexadoku@elektor.com

Prize Winners

The solution of Hexadoku in edition 4/2019 (July & August) is: **70DA8**.
The €50 / £40 / $70 book vouchers have been awarded to: Ulrich Schoor (Germany); Peter Maarse (The Netherlands); Michel Jamin (France); Peter Wäckerle (Switzerland); Vladimir Saric (Serbia).

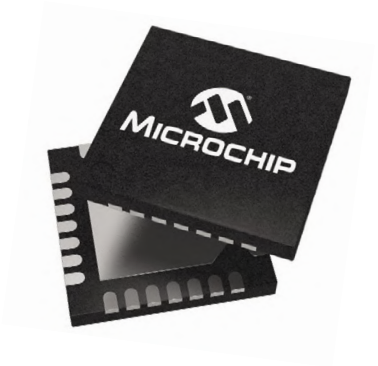
Congratulations everyone!

8	3		0			E			6			B		F	4
		5			2	3			D	1			8		
7	A			4							3			6	0
9		2				7	D	C	E				3		A
							6	B							
			9	8		F			A		E	6			
6		B	7	2	5					F	0	4	A		8
E		0	2	7	A					6	4	F	1		D
B	7		8		▨	▨	▨	▨	▨			5		E	1
	C													A	
1						D	9	A	0						7
3				F	E	1			4	8	6				2
	8	4	A		C	2			5	0		7	9	B	
	E				D					7				2	
			C	A	0					3	9	E			
	9		D		F	6			2	B		A		4	

6	9	0	D	3	8	A	E	5	B	1	2	7	F	C	4
7	A	B	C	F	D	2	4	E	3	8	9	0	1	5	6
3	E	F	4	C	5	1	6	7	0	D	A	8	2	9	B
5	1	2	8	7	0	9	B	6	4	C	F	A	D	3	E
8	B	6	2	E	4	3	C	9	7	5	0	F	A	D	1
9	0	3	1	8	A	F	7	4	D	B	6	2	5	E	C
A	5	D	7	2	6	0	1	8	C	F	E	3	B	4	9
C	F	4	E	9	B	D	5	A	1	2	3	6	7	8	0
1	6	7	F	4	2	E	9	B	5	0	D	C	8	A	3
B	2	8	9	5	F	6	A	C	E	3	4	1	0	7	D
D	3	E	0	B	1	C	8	F	A	6	7	9	4	2	5
4	C	5	A	D	3	7	0	1	2	9	8	B	E	6	F
E	8	1	3	A	C	5	F	D	6	7	B	4	9	0	2
2	4	C	B	6	7	8	D	0	9	E	1	5	3	F	A
F	D	A	6	0	9	B	3	2	8	4	5	E	C	1	7
0	7	9	5	1	E	4	2	3	F	A	C	D	6	B	8

Your Choice—Any Core, Any Performance, Any Feature Set

Scalable Performance for Changing Requirements



Are you facing changing design requirements–again? Let Microchip help you put an end to the frustrations and wasted time that result from these changes. Microchip is the only semiconductor supplier innovating across 8-, 16- and 32-bit microcontrollers, digital signal controllers and microprocessors. Our upward-compatible architectures preserve your time and resource investment in code development. In addition, our development ecosystem allows you to leverage a common ecosystem across multiple designs. Changing design requirements don't need to be painful; learn how Microchip can make it effortless.

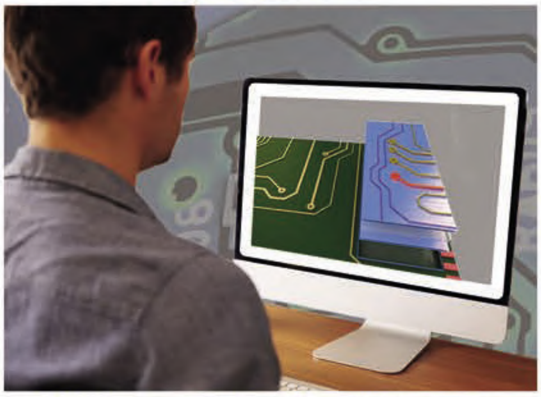
Simplify your life at www.microchip.com/Scalable

MICROCHIP

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 replace 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