Edition 1/2019 | January & February 2019 | www.elektormagazine.com

elektorlabs magazine

DESIGN ⊳ BUILD ⊳ SHARE **ELECTRONICS**

ESP32 Weather Station

Reading sensor data online

DIY Temperature Controlled Solder Station

SMD manual soldering for everyone

FM Radio with RDS

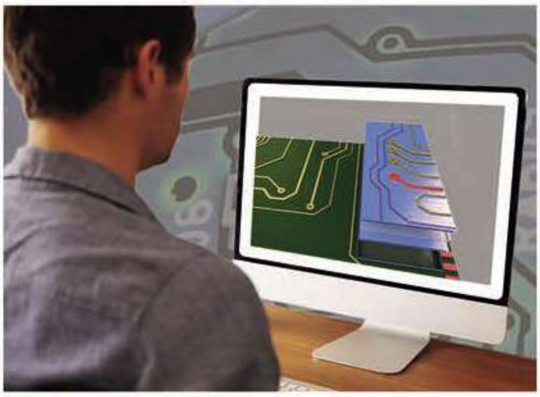
A top HAT project for the Raspberry Pi

Electric Motor Control ⊳ Snore Shield ⊳ Lighthouse 2.0 ⊳ One-time Pad Crypto Shield ⊳ 2.4 GHz Half-Duplex Telemetry ⊳ Old School AC Powerline Conditioning & Regulation ⊳ Verilog Basics ⊳ Personal Speedometer for Runners & Joggers ⊳ PobDuino Board ⊳ Opamp Schmitt Triggers ⊳ The SN76477 ⊳ Q&A: Enclosures ⊳ Low-cost SMD Test Adapter ⊳ Planned Obsolescence?

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

Elektorlabs Magazine

Edition 1/2019
Volume 45, No. 493
January & February 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.

Great electronics with small things

Back in mediaeval times many 'things invisible' like the vacuum, gravity, time, darkness, and static electricity were awe inspiring if not a cause of great fear to the uninitiated. I wonder if the same applies to such seemingly unrelated stuff we struggle with in today's electronics, like 0201-size surface mount components, electromagnetic radiation, microcontroller firmware and buried vias. All very real challenges in the Elektor Labs.

Both invisibility and tiny dimensions initially create a feeling of unrest. When the transistor took over from the valve, a frequently heard complaint was that "it sure is much more efficient but you can't see if the thing is alive or not" and "these things die with not so much as a whisper." Reportedly some radio & TV servicemen actually cut faulty transistors open to see if the clunkers could be fixed (and they could, in some rare cases).

Although far from being invisible, in the case of SMA components (a.k.a. SMDs) we heard reports from our readers like "more ended up in the vacuum cleaner that on my printed circuit board". Being able to view those tiny SMDs is a foremost requirement to using them — next come handling, positioning on a board, and only then, soldering. The art of soldering having been covered in many issues of Elektor magazine — including reflow techniques 'pizza style' — we figured a microscope is sure to open up a world to you if you still feel those minuscule parts are encroaching upon the hobby. The range of Andonstar USB microscopes in our Store are bestsellers and they have helped many of our readers to not just overcome their hesitations about using SMDs in their projects, but also move forward to advanced techniques like board reworking.

Personally, I am nearsighted to the degree of being able to scrutinise SMD solder joints with my nose almost touching the board surface, so an Andonstar 201 could save me the trouble of raising my expensive glasses to an insecure position on my forehead.

Jan Buiting, Editor-in-Chief

The Circuit

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

This Edition

Volume 45 – Edition 1/2019

No. 493 January & February 2019

Regulars

20 **electronica Fast Forward Award 2018**
a glimpse of the future as imagined by smart start-ups

33 **Peculiar Parts, the series**
The SN76477

34 **HomeLab Helicopter**

38 **Tips & Tricks**
Low-cost SMD Test Adapter

61 **Elektor Labs Pipeline**

67 **Questions & Answers**
(Nearly) everything you always wanted to know about... LoRa

80 **Questions & Answers**
(Nearly) Everything you always wanted to know about... Enclosures

99 **Elektor Store Highlights**
What electronicists dream of...

106 **Retronics**
Old School AC Powerline Conditioning & Regulation

110 **Elektor Ethics**
The Interplanetary File System

114 **Hexadoku**
The original Elektorized Sudoku

14 DIY Temperature Controlled Solder Station

SMD manual soldering for everyone

The Platino Solder Station presented back in 2015 received wide acclaim and not just from the Elektor readership. In good engineering tradition, the very fact that the original project worked off the bat was a key motivator to seek enhancements for the intrinsic design. The concerted effort and thought that went into this produced a completely new design which is described here.

Features

6 **Electric Motor Control**
By switch, simple PWM, full bridge ...

23 **Obsolete or „Allocated“?**
Problems in the Lab

68 **What's New in Android Things**
A boost from C++

76 **Making PCBs with your 3D Printer**
Creative solutions with your 3D printer

94 **Verilog Basics**
Fundamental features of the hardware description language

100 **Opamp Schmitt Trigger**
Circuits and calculations

Projects

13 **Frequency Divider with Adjustable Integer Division Factor**
on a (CMOS) stroestring

14 **DIY Temperature Controlled Solder Station**
SMD manual soldering for everyone

24 **Lighthouse 2.0**
With adjustable light characteristic

6 Electric Motor Control

By switch, simple PWM, full bridge ...

elektorlabs magazine

62 **PobDuino Board**
Flowcode and Arduino together fight obsolescence

74 **Automatic Traffic Light Controller**
For model railways

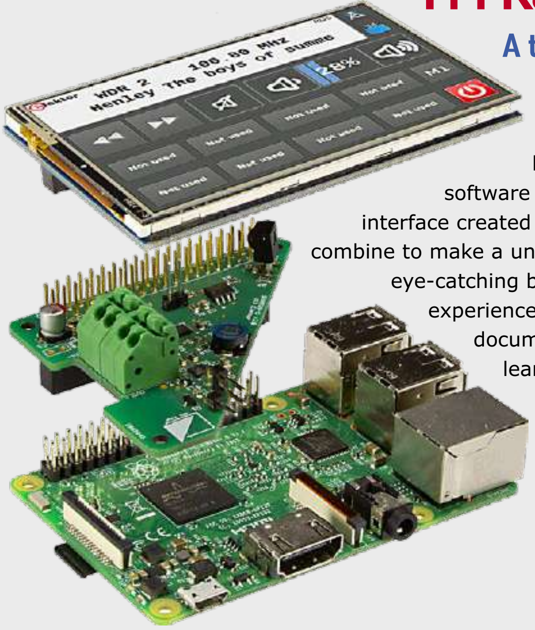
78 **My Little Amplifier**
LM386 gets a makeover

82 **Bad Circuit Howler**
It works... but not the way it should.

86 **FM Radio with RDS**
A top HAT project for the Raspberry Pi

FM Radio with RDS

A top HAT project for the Raspberry Pi



A Raspberry Pi plus a homebrewed HAT with driver software and a custom graphical user interface created specially for this application combine to make a unique FM radio that is not only eye-catching but a fascinating learning experience, thanks to extensively detailed documentation. Nobody can fail to learn something new from this wide-ranging project!

86

28 **ESP32 Weather Station**
Reading sensor data online

40 **Snore Shield**
helps to improve your night rest

45 **Personal Speedometer for Runners & Joggers**
With trip meter and logger

48 **One-time Pad Crypto Shield**
Secure communication through openness

56 **2.4 GHz Half-Duplex Telemetry**
Replace awkward remote-control cables with a license-free ISM radio link.

Next Editions

ElektorLabs Magazine Edition 2/2019

3-Axis CNC portal machine • Styrofoam Cutter • Cross Platform Basic Compilers • Harmonics/THD Measurements in the MHz range • RPI Ruler • Microcontroller Kits for Dummies • VHDL Programming Course (3) • Raspberry Pi IOTA • Audio Spectrum Meter • TCAM Low-cost Thermocam • Stepper Motors in Antiresonance Mode • All-analogue PWM Fan Drivers • Air Pollution Monitor • DIY CPU • Pinball Clock • Simple Function Generator • The Baristor • ADC with PLD.

ElektorLabs Magazine edition 2/2019 covering March and April 2019 is published on 21 February 2019. Delivery of printed copies to Elektor Gold Members is subject to transport.
Contents and article titles subject to change.

Elektor Industry Edition 1/2019

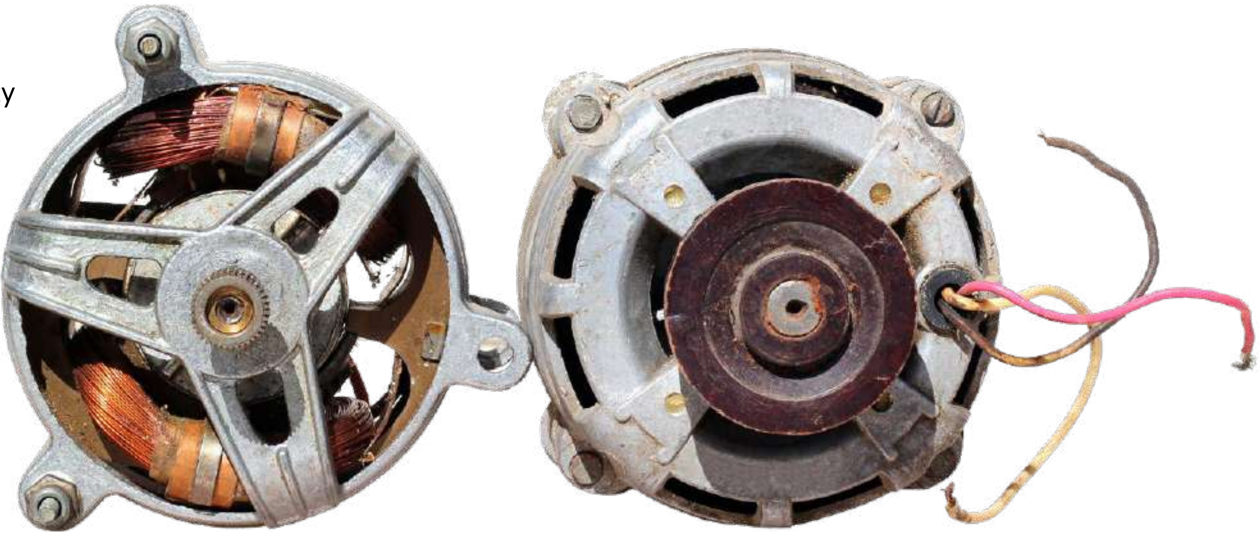
Edition 1/2019 of *Elektor Industry Magazine* will cover microcontrollers, embedded technology, programming and tools, with contributions from companies, industry specialists, Elektor editors, and free-lance authors. Plus, you'll find fresh instalments of regulars like Infographics and Industry Store. Edition 1/2019 has a special focus on the **Embedded World Exhibition & Conference** in Nuremberg, Germany, from 26 to 29 February 2019, in support of Elektor's presence at the show.

Elektor Industry magazine issues are available to ElektorLabs magazine Gold members in print, and Elektor Labs magazine Green members as a PDF download, as well as for purchase by anyone, at www.elektormagazine.com. Publication date: 20 February 2019.

Electric Motor Control

By switch, simple PWM, full bridge ...

By **Dr Thomas Scherer** (Germany)



Wherever electricity is converted into power, or electrical energy into mechanical energy (mostly rotating), it is about the good, old electric motor. Good for electronics engineers, because this type of engine has significantly less mechanical “disturbance” than steam or gasoline engine engines. “Well established” to electronic engineers because arguably all electric motors have long since been invented and hardly anything is happening with their technology. However there are many types of electric motors, and they all need to be controlled suitably for different purposes.

In contrast to thermomechanical motors, where something hot with a lot of noise and smells sets something in motion through crankshafts and the likes, there is much more variety in electric drives, and seeing real use too. Electric motors almost invariably rotate the shaft directly. Not only does that reduce the mechanical complexity, it lowers the price of the application and the overall weight. Thanks to the simplicity, the durability is also better, and precise control easier to implement. On the

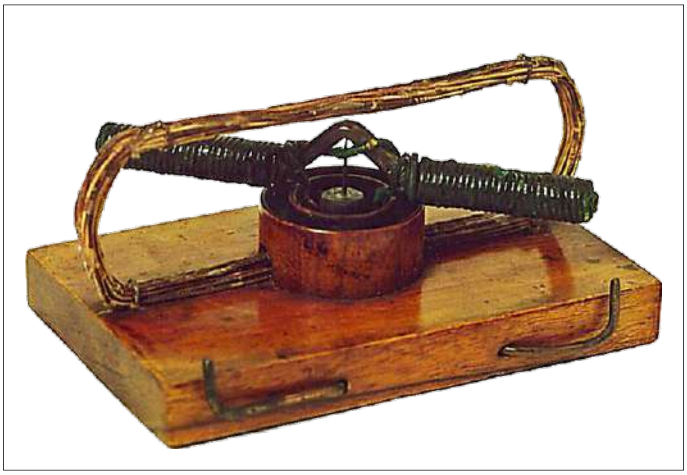


Figure 1. Jedlik engine from 1827 (Picture: *Wikimedia Commons* [2]).

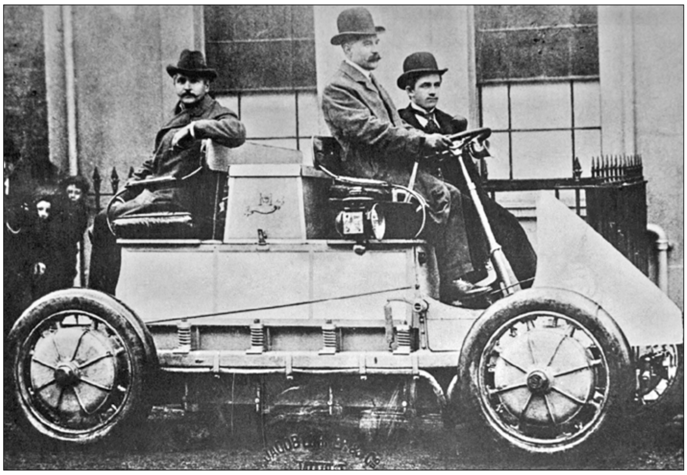
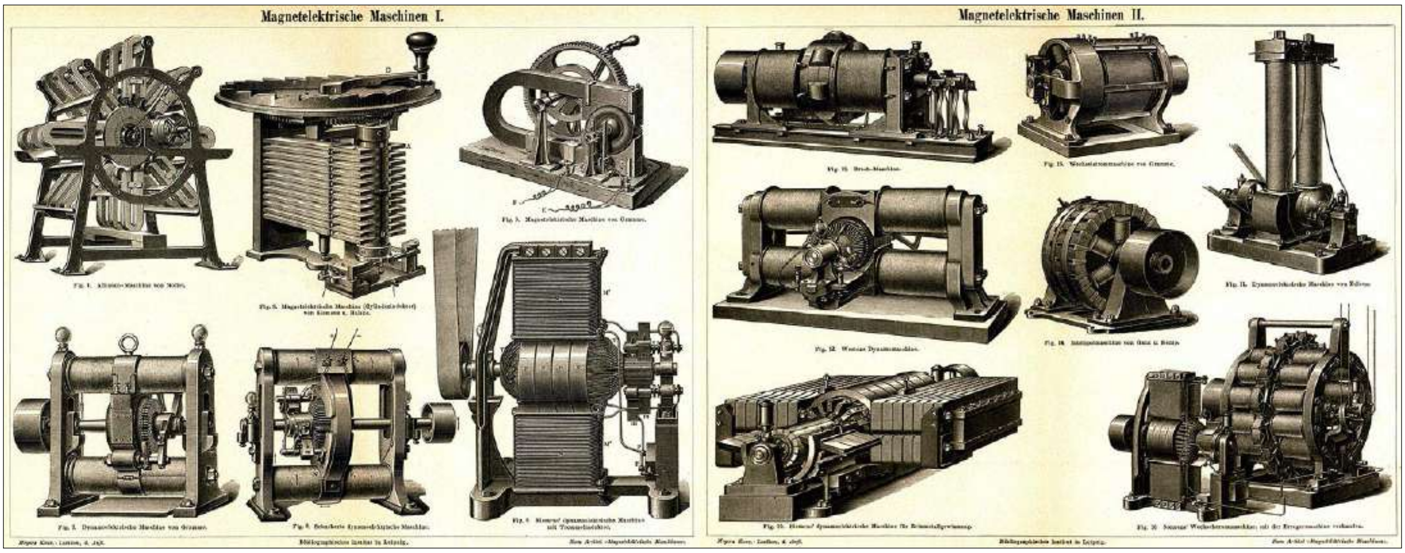


Figure 2. The legendary Lohner Porsche, taken around 1900. Ferdinand Porsche on the far right. Modern drive via four wheel hub motors with 1.5 kW each (Photo: Michael Hereward Westbrook in *The Electric Car* [3]).


Magnetelektrische Maschinen I.
Magnetelektrische Maschinen II.

Already in 1885 the *Meyersche Konversations-Lexikon* (4th edition) showed different engine designs [4][5].

downside, an electric motor is likely to have a cable on it — or a battery as a cable replacement, which makes things heavier, more expensive and a bit more cumbersome.

Variety

After the Dane Hans-Christian Ørsted noticed almost 200 years ago that a current-carrying conductor influenced a compass, he started to work on electrical engineering in general and the turbulent development of electric motors in particular (**Figure 1**). Almost 20 years later, an electrically powered boat was already sailing on the Neva in St. Petersburg, Russia. And every *homo technicus* knows that at the beginning of automotive history more cars drove electrically than on petrol (**Figure 2**). But it was only after the distribution of electrical networks by Edison or General Electric and Tesla or Westinghouse in the USA and the AEG in the then German Reich that electric motors were widely used for a wide variety of applications. Horses, water and wind power soon became secondary and there were good

Table 1. Motor types

Designation	Power	Typical/Peculiarities
3-phase asynchronous motor	W - MW	Passive rotor (no collector), rotor lags stator field; control via contactors or frequency inverter (speed); high starting current (star/delta switching)
three-phase synchronous motor	W - MW	Exciter field necessary (permanent/electrical), rotor synchronous to rotating field, frequency converter (speed); self-controlled rotating field via electronics; generator; external/internal pole machine
stepper motor	<1 W - 1 kW	Version of the synchronous motor with step-by-step rotation of the rotor through certain angles; precise positioning; 24 - 400 steps/revolution
BLDC motor	W - MW	Brushless DC motor = variant of synchronous motor with electronic commutation; for drives, very reliable, often with permanent magnets
capacitor motor	W - <1 kW	Single-phase asynchronous/synchronous motor, rotating field due to phase shift with capacitor in 2nd winding; low starting torque, low maintenance; simple
shaded-pole motor	W - <1 kW	Single-phase asynchronous/synchronous motor, rotating field due to short-circuit winding at shaded pole;low maintenance; low starting torque; smooth running, low efficiency
DC motor	W - kW	Permanently or electrically excited; mechanical commutator generates alternating field; numerous variants; series and shunt connection; high starting torque
Single-phase in-line motor	W - kW	Like direct current motor with plates due to eddy current losses with alternating current supply; high speeds; brush wear, EMC; high starting torque
linear motor	W - kW	No rotor; linear motion; high torque; many variants, positioning; hard disks, Transrapid
servo	W	Combination of electric motor with control and position encoder for angle of rotation detection, model making, PWM control, three wires

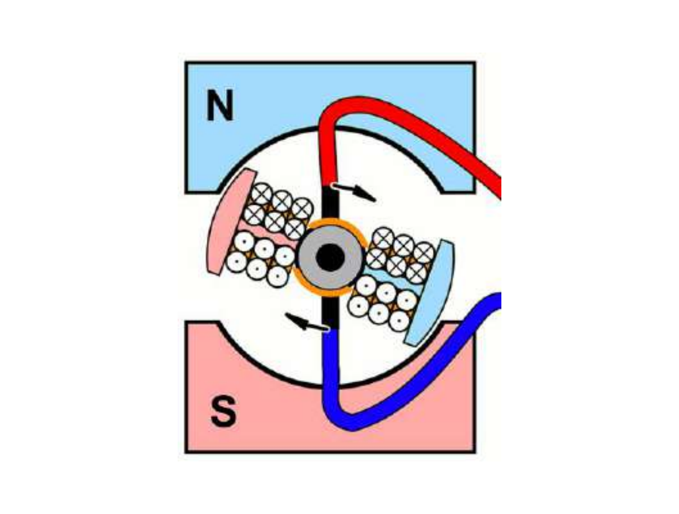


Figure 4. Functional principle and design of a permanently-excited single electric motor. In the stator, at the top the north pole is seen, and below the south pole. The rotor is an electromagnet supplied by a commutator consisting of a copper slip ring and carbon brushes (picture after Michael Frey, CC 3.0 [6]).

options for motorising objects that one could not even dream of before.

Since the end of the 19th century, tinkerers and engineers have been ecstatic about electric drives and developing the most diverse types of electric motors. Does ‚War of the Currents‘ [1] mean anything to you? Around 1890, GE and Westinghouse and the great respective inventors behind them, Edison and Tesla, fought over the fundamental question: direct or alternating current? Consequently, at that time there were not only electric motors of different power ratings and designs (**Figure 3**), but also those that specialised in direct or alternating current and in the latter were even equipped with different numbers of phases. A special motor was therefore available for every purpose. The variety of designations and the associated differences is quite confusing for electronics engineers without special knowledge in the field of electric drives. **Table 1** lists the main engine categories only.

Principle

As this article is more concerned with the operation and control of the engine, the depths and secrets of the actual operation of each individual motor type are omitted. In principle, an electric motor is a construction of one up to many electromagnets, whose magnetic field exerts a force on a permanent magnet, on another electromagnet or on a piece of suitably shaped iron, which leads to a mechanical movement. Electrical energy gets converted into mechanical energy.

As expected, this conversion is never complete, since part of the energy is not only lost through mechanical effects such as friction, but also electrical and electromagnetic effects nibble at the efficiency. The ohmic resistance of the windings, which leads to the production of heat instead of mechanical energy, should be mentioned here. This is further accelerated by the so-called ‘skin effect’ at higher-frequency alternating currents. This is not noticeable with direct current — at the usual 50 or 60 Hz the effect is very moderate and hardly problematic, but already at 30 kHz the so-called ‘penetration depth’ of the alternating current rises to only about 0.4 mm. In addition, there are phenomena such as ‘eddy current losses’ in the ferrous motor parts which are traversed by changing magnetic fields. Finally, secondary aspects such as what happens between slip rings and brushes etc. are disturbing if engines are equipped with them.

Last but not least, the losses in the control system also play a role. With simple electromechanical switches (contactors and relays) this is due to the energy lost as heat in their coils and the generally very low contact and cabling resistances. With a semiconductor-based control, as is mandatory for PWM control of motors, the energy consumption of the control electronics including the power supply unit (smaller part) as well as the voltage drops at connected semiconductors plus the switching losses during fast switching are all relevant, which is mandatory for PWM. Modern power MOSFETs are already quite well suited for small and medium motor outputs up to a few kilowatts, as their on-resistances can be in the sub-mΩ range. At higher power levels, such as in the automotive industry, and higher voltages required, the on-channel resistances of MOSFETs increase appreciably, which in recent years has led to the increased use of silicon-carbide power semiconductors. Silicon carbide allows higher field strengths and thus thinner layers. SiC MOSFETs are therefore particularly well suited for higher voltages, enabling higher switching frequencies and lower on-resistances, resulting in lower losses. In addition, SiC transistors can be operated at higher temperatures, which increases reliability. These advantages are bought, as so often in life, at higher prices.

Functionality

As already indicated, it would go too far to explain in detail how each type of electric motor works. Nevertheless, when it comes to control and operation, it is helpful to have the basic functionality of an electric motor in mind.

Figure 4 illustrates the principle structure of a simplified electric motor. Outside, respectively at the top in the picture (north) and below (south) ire the poles of a permanent magnet system. Since it does not move in this variant, it is called a stator. The part rotating inside is therefore the rotor, which is also called armature. Its iron core has a winding of isolated copper wire. The rotor functions as a rotatable electromagnet. The whole construction is called an “inner rotor” and could also be the other way round: a fixed inner part with a rotatable outer part, like in so-called hub motors.

If ‚somehow‘ direct current were allowed to flow through the winding, the blue north pole of the rotor would be attracted to the red south pole of the stator and the rotor south pole would find the stator north pole equally attractive. As a result, it would do a zigzag once and the rotor would have turned clockwise by about 60° and would stop there as if nailed down.

As this would be rather useless, the current for the rotor winding is supplied via a two-part slip ring (orange at the picture) and brushes (black at the picture, mostly made of carbon. With the smallest motors, springs made of precious metals are also used. This arrangement, called a commutator, switches the current flow through the winding depending on the rotor position, in such a way that the polarity of the electromagnet is always appropriately reversed from the vertical position. Consequently, a continuous torque is generated in one direction of rotation.

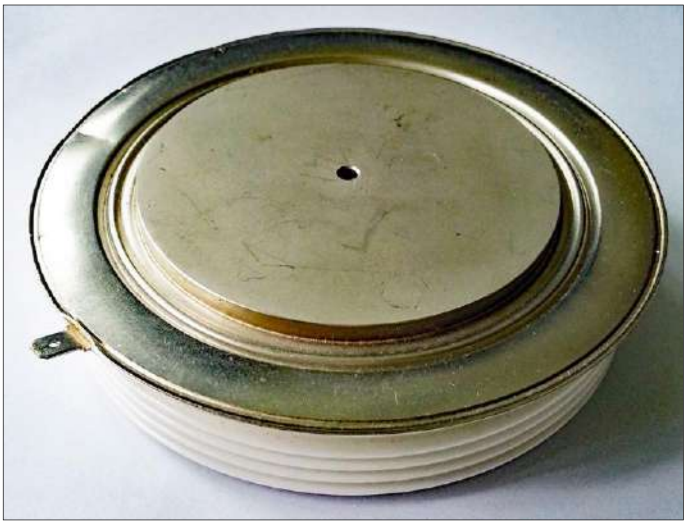


Figure 5. This high-performance thyristor for 4.2 kV at 2.4 kA has a diameter of 11 cm (image: Johnny.m76 / CC 3.0 [7]).

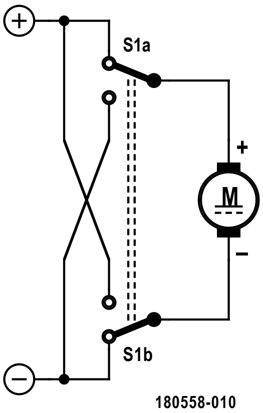


Figure 6. Switching the direction of rotation of simple DC motors is achieved with two coupled switches.

this principle is both ingenious and simple, and allows a lot of variation. Not only inside and outside can be swapped, but also permanent magnet and electromagnet. In addition, the permanent magnet can be replaced by a second electromagnet. In the example in Figure 4, the stator would then also get a winding. And already the possibilities are further diversified: the second winding can be connected in series with the first, or in parallel. In the first case, there is the so-called series-wound motor, which is also known as the 'universal motor'. It can be found almost everywhere, from drilling machines to vacuum cleaners as well as in industrial applications. It is universal because it can in principle be operated with direct and alternating current.

A disadvantage of commutators is their abrasion: the brushes wear, as well as contact surfaces of the slip ring in the long term. Consequently, multi-phase motors are built which, when operated on a multi-phase network, use the resulting rotating field and can do without slip rings. There are other tricky designs which allow a rotating field with specially attached short-circuit windings to be achieved even in single-phase mode. Today, slip rings can also be replaced by electronic switches, which are controlled either by position sensors fitted inside the motor or by the back-EMF. With inverters, the speed of synchronous motors that are otherwise rigidly coupled to the mains frequency can be made variable within limits. All these changes ultimately lead to a diversity of engines that is roughly divided into categories in Table 1.

Also interesting: almost all motor types can function as generators and thus generate electrical energy from mechanical energy. Sometimes only minor design adjustments are required. In some cases, it's really useful: for example, motors can be braked by "destroying" the kinetic energy in the wake after switching off by heating (also in the winding). In other cases this can interfere, because a motor movement may activate the connected control electronics due to the generator effect.

Controls

Linked to so many different electric motors comes a wide diversity of control concepts. A simple switch only meets the requirements and possibilities of various electric motors in a few cases and consequently does not meet the targeted applications. For this reason, a list of motor control principles has been compiled according to complexity. Model construction servos and stepper motors are omitted, being less suitable for drives.

Switches

Low and medium power motors for drives are often simply switched. In the simplest case this is done through a mechanical switch. This can be replaced by a relay or a contactor (relays for medium to high power for operation on the AC mains), which makes a motor remotely switchable via a control line. In addition, there are semiconductor relays as an electronic alternative for small and medium powers as well as small to large thyristors (**Figure 5**) and triacs with which motors of many kilowatts can not only be switched, but also controlled in terms of speed and output power. With switches, even more is possible.

Direction of rotation by switch

One of the simplest ways of switching the direction of rotation of a DC motor is to use a double-pole changeover switch (**Figure 6**). Of course, this can be achieved with any type of switch, including semiconductors. But beware: If you reverse the polarity of a running motor directly without stopping, this can lead to very high currents and voltage peaks due to the generator effect, as well as damage to the motor or its suspension due to the enormously high torques. The effects are worse than a mere short circuit of the windings. The fact that something works in principle does not mean that it is good or even practicable. Except for small motors, such a mechanical changeover switch should actually have a middle or third position in which the motor is first braked.

The direction of rotation can, of course, also be changed for several single-phase AC motors that have two separate windings. In these cases the polarity reversal of one of the two windings via a switch, is sufficient. With many 3-phase motors it is simply possible to exchange two phases. It should be noted that alternating current motors occasionally have a preferred

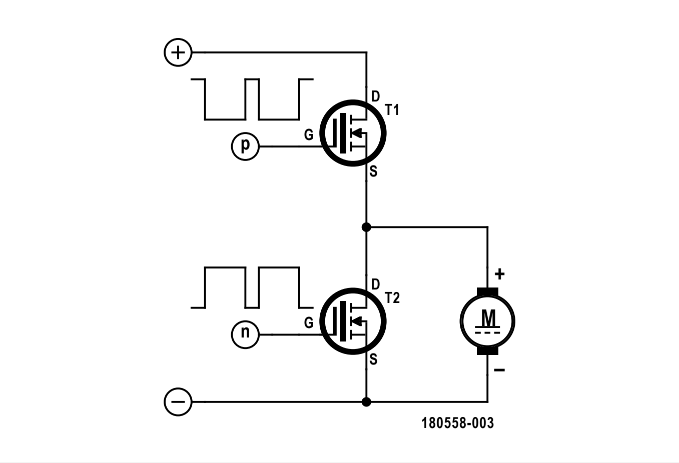


Figure 7. The speed of DC motors can be controlled with a half-bridge consisting of two electronic switches controlled by opposite-phase PWM signals.

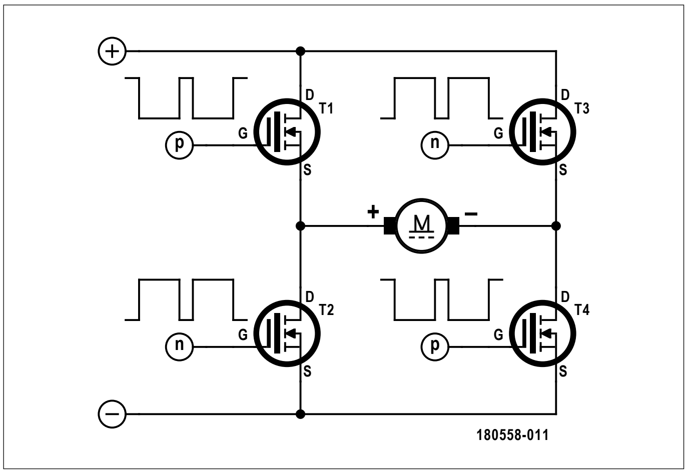


Figure 8. With a full bridge, quasi a doubling of the circuit of Figure 6, speed and direction of rotation can be controlled via PWM signals.

direction, since the commutator of these motors is rotated by a few degrees compared to the theoretical original, which can benefit some aspects such as efficiency.

Speed through PWM

In addition to the direction of rotation, the speed of DC motors can also be controlled. The simplest but energetically most unfavourable case is an adjustable series resistor, a kind of load potentiometer. However, since this heats up useless energy in partial load operation and also has other disadvantages, an electronic, 'digital' control is better. A half-bridge is sufficient using two fast semiconductor switches, pictured in **Figure 7** as typical N-channel MOSFETs. The switches are controlled by opposite-phase PWM signals, so that the switches never conduct simultaneously. The mean voltage results directly from the supply voltage times the pulse/pause ratio. If, for example, the DC motor in Figure 7 were supplied with 12 V, the symbolised PWM signal and a duty cycle of 33 % (signal 'p') would provide the motor with an effective 4 V supply. The inductance of the motor and its mechanical inertia reliably compensate the real square wave voltage at a sufficiently high control frequency.

Anyone wondering why this procedure is not just a power control but actually a speed control should consider how an (asynchronous) electric motor works. Not even idle speed results simply from the combination of input power and resulting (mechanical) losses. The back-EMF or the back voltage induced by the rotation is proportional to the speed. It therefore does not rise arbitrarily high even when idling, but becomes exactly so high that it just remains under the applied voltage. The quotient of the resulting small difference and the effective resistance of the winding(s) allows exactly enough current to flow to produce a torque that compensates for friction and other losses. With the motor loaded, speed will decrease correspondingly, hence the difference between applied voltage and counter-EMK and (i.e. current) will increase. The current is proportional to the required torque and the output power is the product of speed and torque.

These associations also mean that a running motor should not be short-circuited suddenly: its ohmic resistance is very small, the back-EMF is still large and the resulting current and torque may be quite destructive. This also applies if the set speed is suddenly and abruptly reduced with a speed controller as shown in Figure 7: the available excess kinetic energy must be removed, straining the transistors and possibly destroying them. On the other hand, fast starting ensures high starting currents that have to be kept in mind.

Full bridge

Speed plus direction of rotation can be controlled with a full or H bridge as shown in **Figure 8.** In principle, it is a doubling of the half bridge of Figure 7. The second half bridge on the right is also usually controlled with PWM signals in phase opposition. However, there is a small but fine difference here: the positive PWM signal 'p' is applied to the lower switch T4 and the N signal is applied to T3. The motor is in the H-beam of the bridge. The motor stops when all PWM signals have a pulse/pause ratio of exactly 50%, since the difference of the mean voltages of both half bridges is zero then. If the duty cycle of the P signals is above 50%, the motor rotates forward; if it is below 50%, the motor rotates backward.

In principle, full duplication with 2×2 PWM signals for controlling the four transistors is a luxury. There could also be a simpler combination in which the left half bridge is supplied with PWM signals as in Figure 7, but the right half bridge of Figure 8 is then statically controlled instead of PWM signals. In other words, the speed is controlled on the left between 0 and 100% and the direction of rotation on the right. Then you have to pay attention to the control logic: if you want to go from 100% forward to 100% reverse, you should first reduce the duty cycle for the P signal (slow) from 100% to 0%, while T4 is continuously switched through and T3 is permanently open. From then on, T4 must open and T3 close and the duty cycle of the P signal must jump abruptly to 100% so that 0 V is initially applied to the motor.

BLDC control

Multiphase motors can also be adjusted in speed and the direction of rotation can be reversed by PWM. The 'classic' is the brushless DC motor aka BLDC motor, as it is used in numerous modern drives (from e-bikes to Segway to electric cars

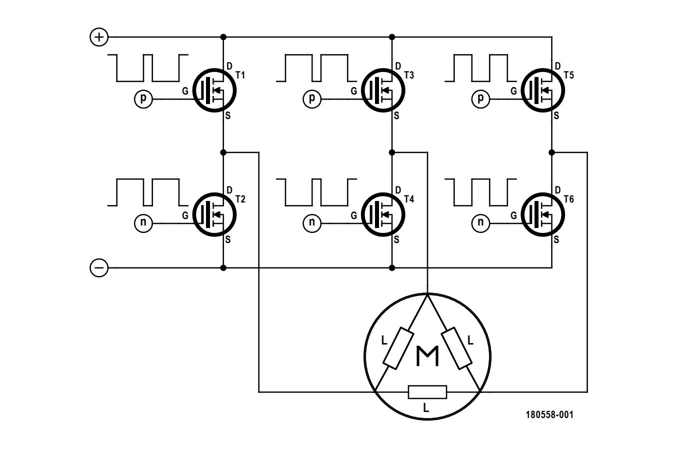


Figure 9. With (only) three half bridges, so-called BLDC motors can be supplied with the necessary three-phase current including adjustable amplitude.

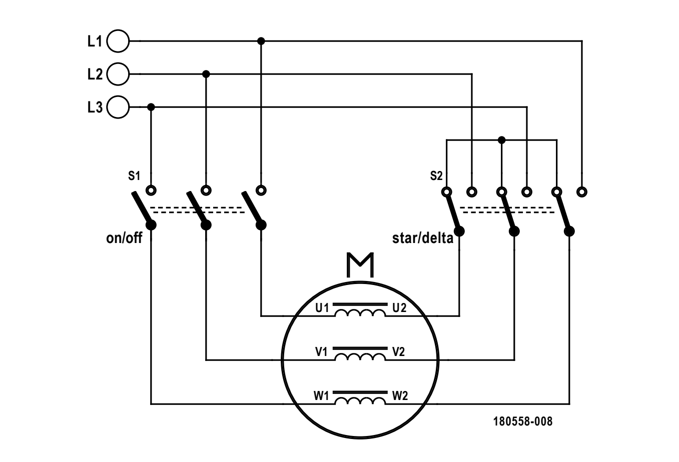


Figure 10. To avoid excessive starting currents, 3-phase motors of medium power are started in 'star' configuration and only switched to 'delta' at higher speeds.

and many other applications). The simple version has three phases with only three connections. In contrast to industrial versions, in which all connections of the three windings are usually bonded out, there is no choice between star or delta connection. However, this is not necessary due to the usual fully electronic control, since high starting currents can be intercepted by the PWM system.

The additional circuitry required compared to single-phase motors is very limited: Compared to Figure 8, **Figure 9** has only one more half bridge. The devil is in the control logic: since a rotating field has to be generated by PWM to obtain rotation, and a BLDC motor is a synchronous machine in principle, the voltages on the three lines of the motor have to be commutated according to the position of the rotor, in such a way that the three counter-phase PWM signals of the three half bridges are offset by 120° from each other. However, this electronic commutation requires information about the current position angle of the rotor. There are two ways to do this: either the motor is equipped with (at least three) Hall sensors, or the counter EMF is evaluated in the switching pauses of the half bridges. With this information, the controller can provide the appropriate PWM signals to the three half bridges so that the field and thus the motor rotate correctly. Through self-control via sensor or EMF of the motor, it ultimately shows asynchronous behaviour.

Such a BLDC motor can be controlled by 3-phase alternating voltages simulated by PWM, which in the simplest case have a rectangular shape. Sinusoidal voltage curves per PWM are also not a problem. With trapezoidal alternating voltage, high efficiencies can be achieved with low torque ripple at the same time.

Star/delta switchover

For the sake of completeness: Common three-phase motors of medium and high power often have high starting currents. In order to reduce these currents the motors can be started via limiter resistors and only be fully connected to the mains when the speed is sufficient. This is actually done in the industry -— there are even adjustable water resistors for this. For powers up to about 10 kW however it is sufficient to use the fact that a motor in star connection consumes only about 1/3 of the power in delta connection. It is therefore started in star connection and its three windings are switched as a delta after reaching more than 75 % of the nominal speed.

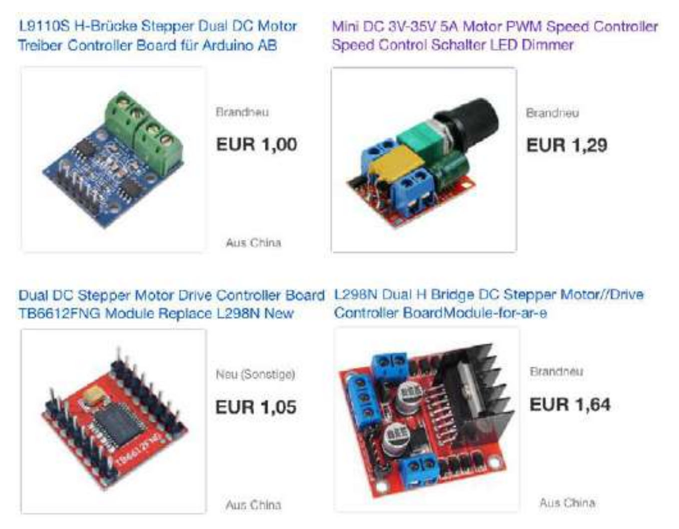
Figure 10 shows this in contactor circuit technology. The triple changeover switch S2 must not be of the 'make before break' type under any circumstances, i.e. it must interrupt before switching, otherwise there will be a bang. An additional contactor with two switches could change the direction of rotation by (correctly) exchanging two connections (e.g. U1 and U2). In principle, however, such star/delta switches are largely obsolete today, especially in industry, since the motors can be started gently by suitable frequency converters.

Electronics

It has already been mentioned that modern motor control systems mainly use MOSFETs for their power semiconductors, except for extremely high powers large plants. Interesting however is the actual control of a half bridge with the PWM signals in phase opposition. In principle, it is advisable to select the fundamental frequency of the PWM high enough so that it can no longer be perceived acoustically by humans and animals, otherwise the motor will function as a loudspeaker at least partially. On the other hand, switching losses increase with increasing PWM frequency. As a rule, the PWM frequencies are therefore at a compromise of 30 to 60 kHz.

Since the transistors of a half bridge must never conduct simultaneously, a so-called dead time should be inserted between the switching off of one transistor and the switching on of the other. The exact time depends on the characteristics of the power semiconductors and of course the PWM frequency. Typical values are between 0.1 and 2 µs. At a PWM frequency of 50 kHz, a dead time of 0.5 µs would mean that the upper 2.5% of the maximum power would not be reached, which is usually tolerable.

If you wish you can generate the matching PWM signals in phase opposition with matching delay times "by hand" with a suitable microcontroller. With enough pins, you can even generate all six PWM signals for three half bridges via software and then have everything under control. However, this flexibil-



L9110S H-Brücke Stepper Dual DC Motor Treiber Controller Board für Arduino AB

Brandneu

EUR 1,00

Aus China

Mini DC 3V-35V 5A Motor PWM Speed Controller Speed Control Schalter LED Dimmer

Brandneu

EUR 1,29

Dual DC Stepper Motor Drive Controller Board TB6612FNG Module Replace L298N New

Neu (Sonstige)

EUR 1,05

Aus China

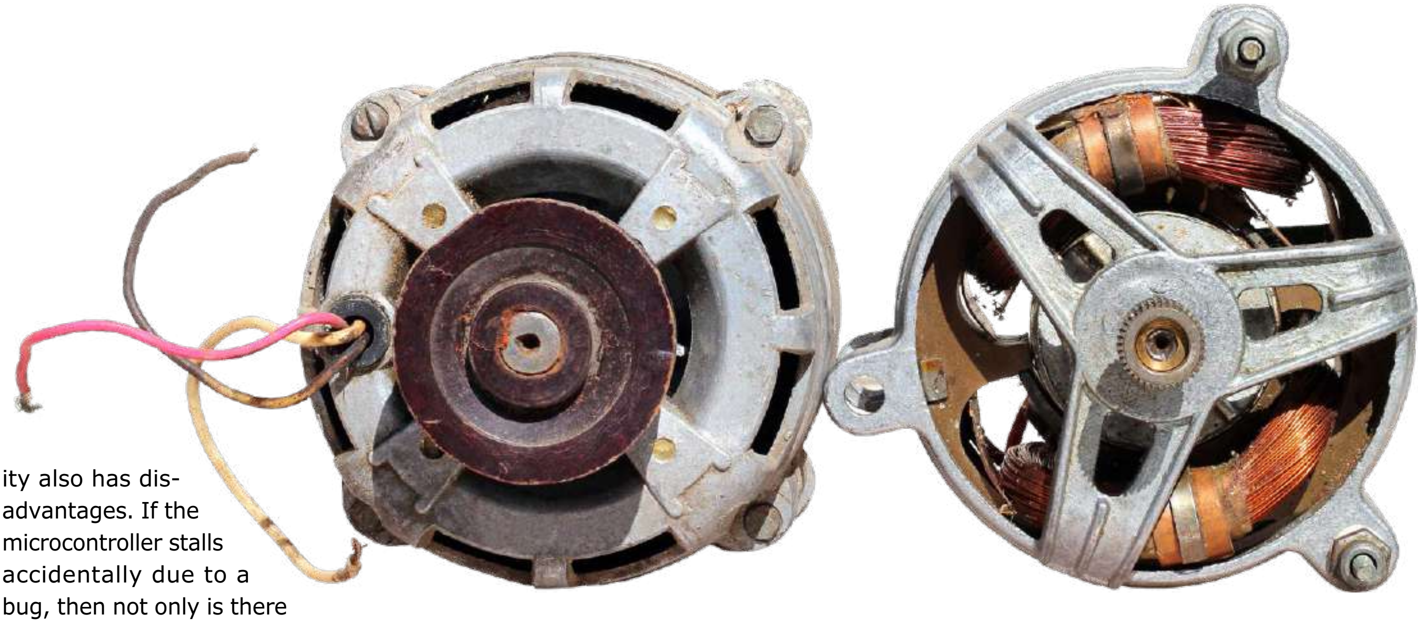
L298N Dual H Bridge DC Stepper Motor//Drive Controller BoardModule-for-ar-e

Brandneu

EUR 1,64

Aus China

Figure 11. An arbitrary selection of inexpensive small controller boards for low-power motors from eBay's offerings.



ity also has disadvantages. If the microcontroller stalls accidentally due to a bug, then not only is there a high probability that the electronics will say goodbye with a bang, but also that dangerous mechanical damage including bodily injuries will occur. Therefore, at least the use of special half-bridge drivers for MOSFETs is recommended. They contain everything you need to control a typical N-channel MOSFET pair, including the generation of reverse-phase control signals with (adjustable) delay time and subtleties such as bootstrapping to control the gates of high-side FETs (T1, T3 and T5 are shown in Figure 9). On the input side, they only need a normal single-phase PWM signal. Some driver chips even contain the electronics for three half bridges.

The PWM signals for driving such driver ICs can then be taken over by a small microcontroller that has little to do. In terms of safety, it is important that such microcontrollers, even with sufficient computing power, do not carry out any additional tasks that go beyond pure motor control including safety shutdowns, soft starts, etc. Or you can use specialised motor controller ICs and then be on the safer side at the expense of some flexibility. For small motors of up to a few tens of watts, there are even integrated complete solutions consisting of a controller, drivers and power semiconductors. You only have to stick to the datasheet ;-)

@ WWW.ELEKTOR.COM

→ SunFounder PiCar-V for Raspberry Pi:
www.elektor.com/sunfounder-picar-v-for-raspberry-pi

→ JOY-iT MotoPi – Motorsteuerung for Raspberry Pi:
www.elektor.com/joy-it-motopi-motor-control-for-raspberry-pi

→ JOY-iT Motorino – Motorsteuerung for Arduino:
www.elektor.com/joy-it-motorino-motor-control-for-arduino

If you want to experiment with ready-made solutions, you should enter the terms "motor controller" in the search field of eBay. Here there are thousands of boards and assemblies of various types at prices between 1 and k ranges — for applications ranging from single-phase low-power motors (**Figure 11**) to industrial control systems (and almost invariably from China). ⏮

180558-02

Web Links

[1] War of the Currents: https://en.wikipedia.org/wiki/War_of_the_currents

[2] Jedlik Motor: http://www.jedliktarsasag.hu

[3] Lohner-Porsche: https://tinyurl.com/y8kx2ldr

[4] Encyclopedia, left/h picture: https://tinyurl.com/y7mn24fz

[5] Encyclopedia, right/h picture: https://tinyurl.com/yc2pyp6k

[6] Motor, principle of operation: https://tinyurl.com/yb98ug89

[7] Thyristor: https://tinyurl.com/ychbckro

Frequency Divider with Adjustable Integer Division Factor

on a (CMOS) stroestring

By **Michael A. Shustov** (Russia) and **Andrey M. Shustov** (Germany)

Developing a frequency divider with an adjustable division factor is often said to be a complex assignment. In response to that “challenge”, we developed a circuit that although rudimentary, still allows you to divide the frequency of a digital signal by a factor *n*.

The circuit shown in **Figure 1** consists essentially of one CMOS XOR gate type 4070B (IC1.A) and one noninverting buffer type 4050B (IC2.A), with an adjustable *R-C* network in between them, and feedback applied. The divisor (division factor; scale fator) *n* is an integer value between 2 and 1000, and possibly upwards of that.

The desired division factor is adjustable on trimpot (preset; potentiometer) P1. With P1 effectively 0 Ω, the divisor *n* equals 2. Setting the pot to its maximum resistance (10 kΩ) results in *n* = 94. Increasing the trimpot value to 75 kΩ results in *n* = 1000. In the range 9 < *n* < 30, the relation between the resistance effectively set on P1 and the resulting division factor is practically linear and described in a formula as

n = ROUND (P1 + 0.2) / 0.133

where P1 is in kilo-ohms (kΩ), and ROUND indicates a rounding-off operation to the nearest integer number.

Figure 2 shows the input and output signals of the divider obtained with the following parameters:

- V_{DD} (IC1; IC2 supply voltage) = 10 V
- f_{in} swing at input: 10 V
- P1 set to: 2.4 kΩ

To obtain other division factors, it is possible to connect multiple, identical, frequency dividers in series (“cascading”). Each connected frequency divider should be realised using its own gate and buffer contained in the CD4070BP and CD4050BP ICs respectively. Note that buffered CMOS chips should be used (suffix: B). Also, the f_0 input must not be left open-circuit, and the same goes for any non-used inputs of the CMOS ICs used the circuit. ◀

180559-01

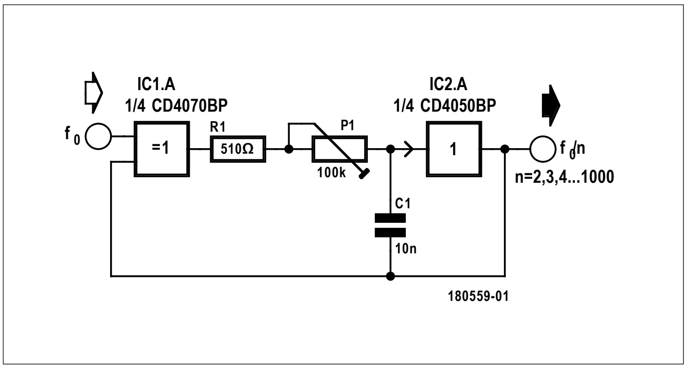


Figure 1. Schematic of the adjustable frequency divider. The IC supply voltage V_{DD} is not shown and assumed to be 10 V_{DC}.

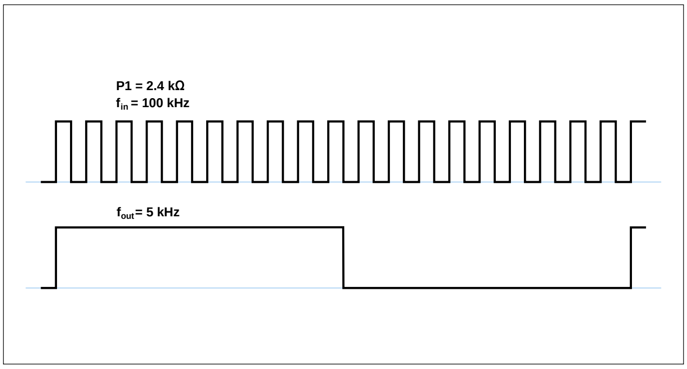


Figure 2: Example of input and output signals of the frequency divider configured for a divisor *n* = 20.

DIY Temperature Controlled Solder Station

SMD manual soldering for everyone

By **Sunil Malekar** (Elektor Labs India)

The Platino Solder Station presented back in 2015 received wide acclaim and not just from the Elektor readership. In good engineering tradition, the very fact that the original project worked off the bat was a key motivator to seek enhancements for the intrinsic design. The concerted effort and thought that went into this produced a completely new design which is described here.

The basic idea behind the DIY SMT soldering station published about three years ago [1] has not changed, and continues to use the Weller 'RT' solder tips with the handy 3.5-mm jack connectors. These tips have the heating element and temperature sensor built in, and so all you have to do is provide enough power to heat the tip, and a method to precisely control its temperature. Here's such a controller in extremely compact form.

Heating is easy

Although we do not know what exactly hides inside the type RT1 solder tip (**Figure 1**) — sadly but understandably Weller does not divulge this information — it is safe to assume that the heating element consists of some sort of resistive device that gets warm when a current flows through it: the higher the current, the hotter the tip. Heating the tip to a constant temperature can be done by regulating the current through it. Because the tip includes a temperature sensor, that's quite easy to do: pass a controlled amount of current through the tip, and subsequently measure its temperature. Then lower the current if the temperature is found too high, or up it when the temperature is too low.
Today in the digital age where zeroes and ones define the way our contraptions (and some people) operate, such a control loop is typically implemented using a microcontroller and a pulsewidth-modulated (PWM) signal. The average value of the PWM signal determines the average current through the heating element (**Figure 2**).

PWM rules

PWM (pulsewidth modulation) control is cool for several reasons. First of all, microcontrollers excel at generating PWM signals — it is one of the things they do best (besides going haywire when it is most inconvenient), and multiple channels with 16-bit precision or more are easy to get. Furthermore, PWM signals are great for controlling (power) transistors because they make them switch rapidly from On to Off and *vice versa* without spending much time in the intermediate energy-wasting region (where analogue amplifiers like to operate). As a result, power transistors can switch high currents without running hot, and even more so when they are of the MOSFET variety (metal oxide silicon field effect transistor) marked by extremely low On resistance (R_{DSON}). Together these two properties allow the design of a solder tip power driver to remain simple.

Separate measurements from interference

Complications arise when we want to measure the tip’s temperature with any sort of precision. Contrary to recommended good design practices for precision measurement systems, the temperature sensor’s leads intended for conveying the tiny varying temperature signal are very close to the heating element’s leads carrying a high-power signal. Even worse, the sensor and the heater share a wire. Although we can understand why the solder tip was designed this way, it does complicate a system designer's life a bit.

PROJECT INFORMATION

sensors
measurement
soldering

entry level
➡ intermediate level
expert level

3 hours approximately

Standard tools
(including soldering tools)

€60 / £50 / $75 approx.

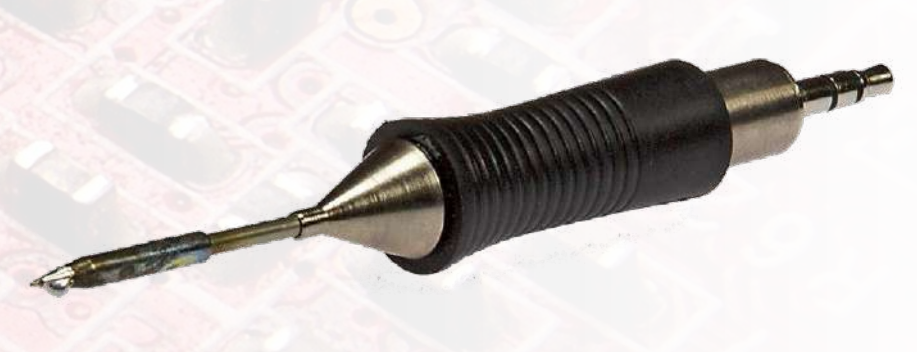


Figure 1. The Weller solder bit type RT1 has a 0.2-mm needle tip. The ‚RT‘ series has many different models to suit requirements for precision soldering.

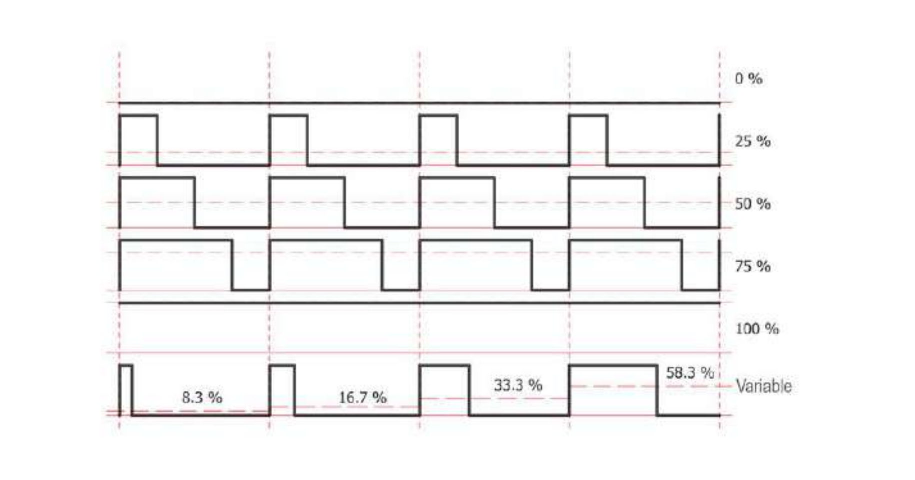


Figure 2. A digitally generated PWM signal and how it relates to analogue average values.

Features

- +50 °C to +450 °C temperature range
- Precise temperature control
- Removable RT series solder tip
- Temperature data logging
- External 12 V DC / 2 A power source

Applying some filtering to the sensor's output signal and carefully planning the moment when to sample it is therefore highly recommended if not obligatory.

Maximum thermal power control

The design principles we just discussed apply unequivocally to the design described here as well as to the previous Platino-based design. Other things shared with the previous design [1] are the display — albeit this time we use a graphical OLED display — and the rotary encoder. New is the output current measurement that enables us to control the maximum amount of thermal power the soldering tip is allowed to produce. This helps to protect the tip against dangerous currents and to protect small parts against excessive heat.

On to the circuit

After the explanations above, the circuit — shown in **Figure 3** — can be easily understood. For a change let's start our detailed description at the output, connector K3. This is where the RT1 solder tip is to be connected. Pin 1 carries the heater voltage, pin 2 the signal from the temperature sensor, and pin 3 is for Ground.

MOSFET T1, under control of the PWM signal, periodically connects the heating element in the solder tip to the heater supply voltage V_{IN} (12 V typically). The gate of a MOSFET like T1 behaves almost like a capacitor hence a push-pull stage, T2-T4, is required to drive the gate hard to make sure the FET is keyed on and off as fast as possible, simply by charging and discharging the gate capacitance and without going through hoops to optimise it all. Transistors T2 and T4 are driven by T3 because the microcontroller cannot switch as high up as V_{IN} (12 V).

The heating current produces a small voltage across R18, which is copied and

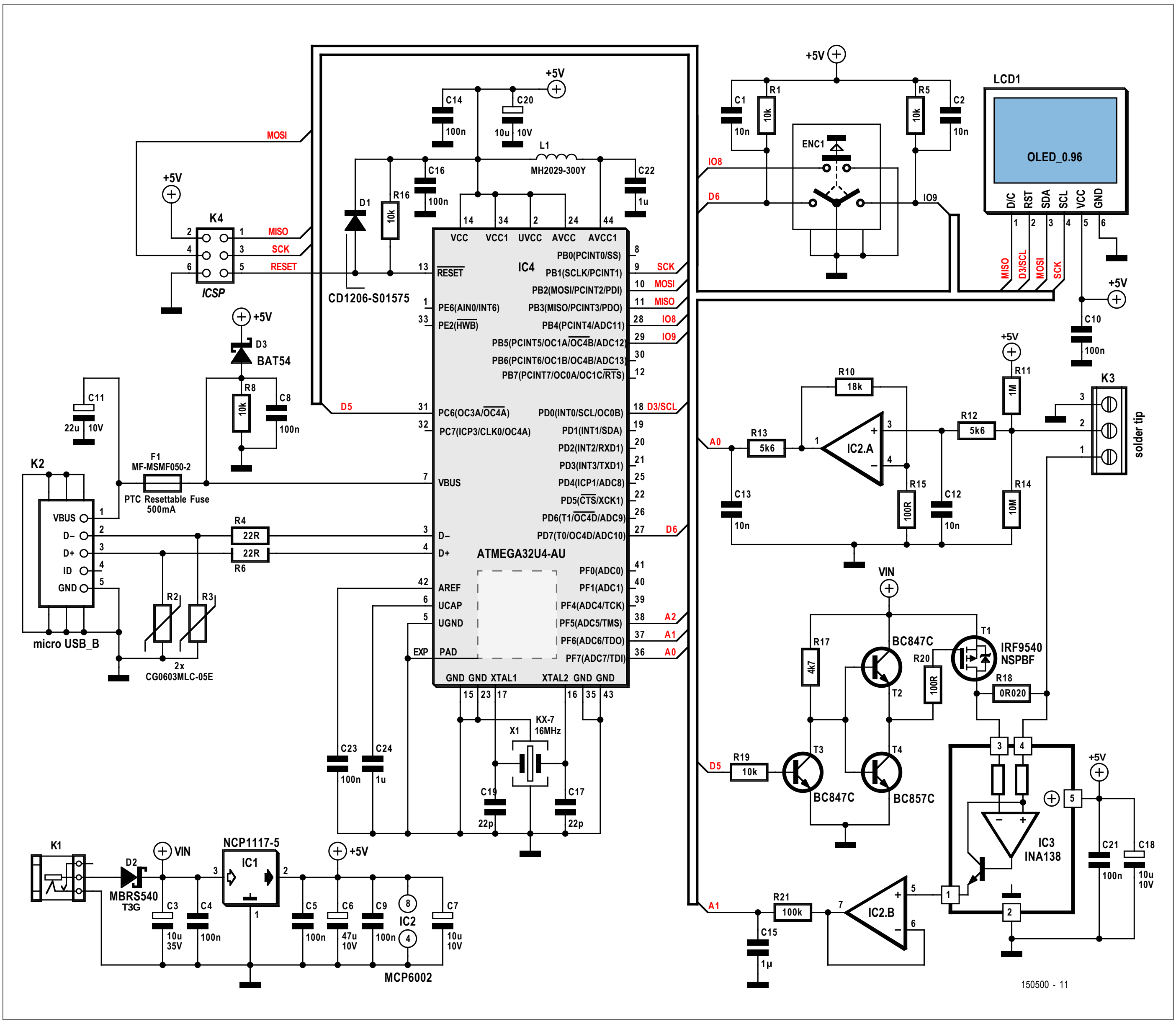


Figure 3. The schematic diagram of the soldering station cleverly mixes analogue and digital electronics to achieve its objectives.

COMPONENT LIST

Resistors

Default: SMD 0805, 100mW, 1%
R1,R5,R7,R8,R16,R19 = 10kΩ
R2,R3 = CG0603MLC-05E, ESD suppressor, USB compatible, SMD 0603
R4,R6 = 22Ω
R9 = 2.7kΩ
R10 = 18kΩ
R11 = 1MΩ
R12,R13 = 5.6kΩ
R14 = 10MΩ
R15,R20 = 100Ω
R17 = 4.7kΩ
R18 = 0.02Ω, 2 W, SMD, current sense resistor
R21 = 100kΩ
R22 = 560kΩ

Capacitors

Default: SMD 0805
C1,C2,C12,C13 = 10nF
C3 = 10 µF 35 V, Case C
C4,C5,C8,C9,C10,C14,C16,C21 ,C23 = 100nF
C6 = 47µF 10V, Case A
C7,C18,C20 = 10µF 10V, Case A
C11 = 22µF 10V, Case A
C15,C22,C24 = 1µF
C17,C19 = 22pF

Inductors

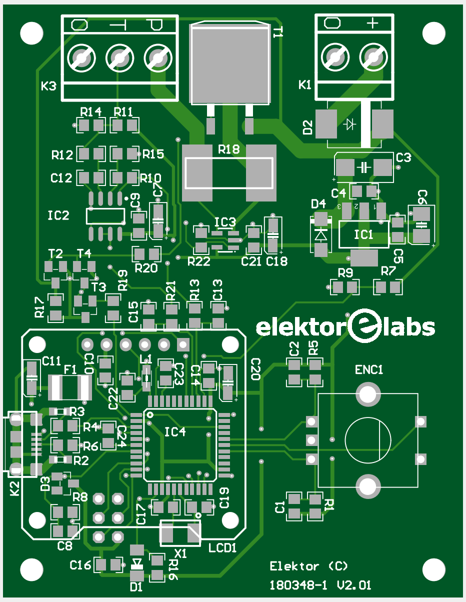
L1 = Ferrite, 3A, 30Ω @ 100MHz, 0805

Semiconductors

D1 = CD1206-S01575, SMD 1206
D2 = MBRS540T3G
D3 = BAT54
D4 = S1A
T1 = IRF9540NSPBF
T2,T3 = BC847
T4 = BC857
IC1 = NCP1117ST50T3G, SOT-223-3
IC2 = MCP6002-E/SN
IC3 = INA138NA/250G4
IC4 = ATmega32U4-AU

Miscellaneous

ENC1 = Rotary encoder with pushbutton, Bourns type PEC11R-4225F-N0024
F1 = PTC, 15V, I_{hold} = 0.5A, 4.5×3.2mm
K1 = 2-way terminal block, 0.2" pitch
K2 = micro USB type B connector
K3 = 3-way terminal block, 0.2" pitch
K4 = 6-pin (2×3) pinheader, 0.1" pitch
LCD1 = OLED display, 0.96", 128×64, SPI interface
X1 = 16MHz quartz crystal
6-pin SIL pinheader, 0.1" pitch, for LCD1
6-way pinheader socket, 0.1" pitch, for LCD1
Enclosure (suggested): Teko type 102 110 × 70 × 46mm)
PCB # 180348-1 (Elektor Store)



amplified by the circuit around IC3, IC2.B and R21 permitting the microcontroller's onboard analogue-to-digital converter (ADC) to sample it via signal line A1. Capacitor C15 provides some noise filtering. Note that the current version of the ATmega firmware limits the current to 1.5 amps.

Low-noise amplifier IC2.A boosts the solder tip's temperature sensor output voltage. Networks R12-C12 and R13-C13 together form a second-order low-pass filter to get rid of some of the noise inherently "on the line". The signal on line A0 is sampled by the microcontroller IC4. Resistors R11 and R14 allow detection of a disconnected sensor enabling the supply power to the tip to be cut. This prevents destruction of the tip through overheating.

Digital circuitry

LCD1 represents the 0.96" OLED display in the project. Its wiring may seem strange but do realise that these displays can work in either SPI or I²C mode depending on how some of its resistors and jumpers are set. In this design the display is used in SPI mode. However the labelling on these displays often refers to the I²C signals, which is why it is drawn that way. When you buy this display to use in the project, make sure to get the SPI version.

ENC1 is the rotary encoder that's used to set the temperature of the soldering station. It has a built-in pushbutton and therefore requires three digital inputs IO8, IO9, and D6.

Already briefly mentioned above, IC4 is the microcontroller that provides the calculation power and PWM circuitry for the circuit. It is a type ATmega32U4 which has a USB interface built in. To the left of IC4 we find micro USB connector K2, which connects to this interface. Components R2, R3 take care of any ESD surges that may appear on these lines for some reason. F1 is a resettable fuse. All in all the USB port is pretty resilient, and connecting a computer to it is not a great risk. The soldering station is safe too, and, when programmed with the right software, will appear to the PC as an Arduino Micro or Leonardo.

Power supply

A 12-volt 2-ampère regulated power supply is required to make the solder station work properly. Although a 12-V DC 'brick' supply is cheap and convenient, you should check its output regulation and output current ability before

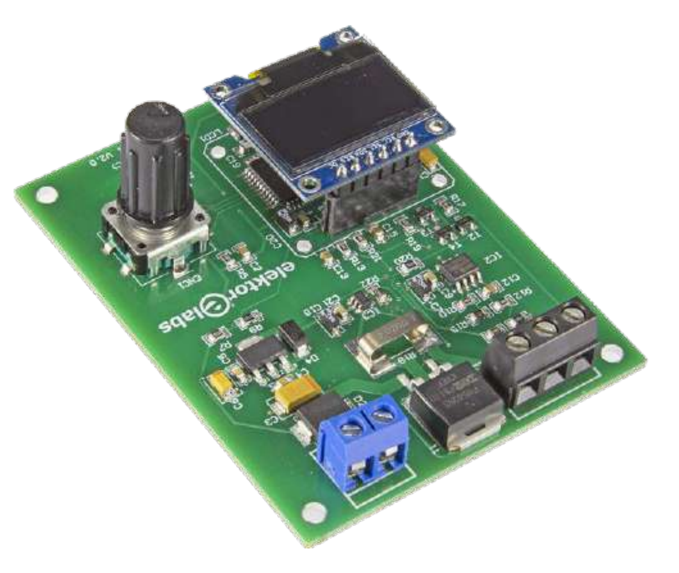


Figure 4. Top side of assembled board with LCD and rotary encoder mounted.

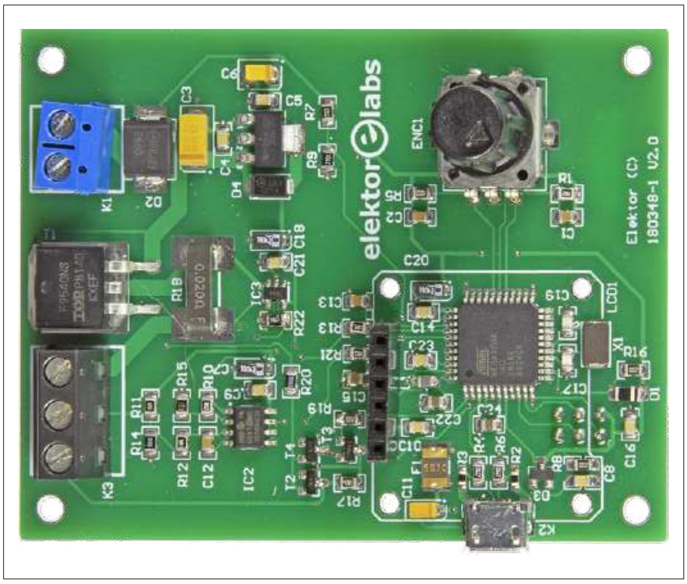


Figure 5. Component side of assembled board with all SMD parts mounted.

using it. If the supply voltage is weak or too noisy under load the temperature measurements may become inaccurate. IC1 makes 5 V out of the V_{IN} supply to power the digital part of the circuit together with the temperature sensor ADC input circuitry. Components L1 and C22 are additional measures to keep noise out of the ADC peripheral.

For development purposes it is also possible to power the circuit from the USB port. When V_{IN} is not applied, D3 connects V_{USB} to the 5-volt rail, effectively powering the circuit. The solder tip will not heat up in this case, but temperature measurements are possible, meaning that you can use this device as a thermometer too!

Details about the software

The program that makes it all work together takes care of several functions:

- PWM generation;
- temperature measurement;
- temperature sensor detection;
- heater current measurement;
- heater voltage measurement;
- USB communication;
- interaction with the user.

After powering up the program arranges for the tip to be heated full throttle to ensure a quick start. Once the target temperature is reached — this takes a few seconds — PWM regulation kicks in, and you can start soldering. Note that the target temperature is stored in the MCU’s EEPROM to avoid having to set it every time the soldering station is switched on.

The tip temperature is measured periodically. To do this properly the PWM signal is first switched off (pulled Low), and after a short delay (to allow the change to have its effect), the signal on A0 is sampled. The current temperature is calculated and compared to the target value, and if the difference is too large (about 5 °C), the PWM value is adjusted accordingly.

Contrary to temperature measurements, heater current measurements are taken when the PWM signal is High (i.e. on) because otherwise the value would always be zero or thereabouts.

The user interface (UI) consists of the rotary encoder and the display. For the first pin-change (PC) interrupts are used; the process is controlled using the popular Adafruit graphics library.

Target as well as measured temperatures and the complete solder station status are transmitted via the USB serial port.

Programming the firmware

The software for IC4 (see [2]) was written as an Arduino sketch. As such you need the Arduino IDE to compile it. Because the solder station is Arduino Micro (Leonardo) compatible, that is also the “Board” to select in the IDE.

K4 is intended for programming the microcontroller. Once you have flashed the Arduino Micro or Leonardo bootloader you can also program the MCU over USB from the Arduino IDE.

Assembly

You can choose to buy a semi kit from Elektor Store, or hamster the parts from your favourite supplier(s) and the board from Elektor, and spend an afternoon soldering SMDs.

Assembling the board requires SMT tools and the photographs in **Figures 4 and 5** of the finished board should provide useful in replicating the project at home. One special component is the 0.02-ohm current sense resistor pictured in **Figure 6**.

The resulting 'module' is quite small and should easily fit in all sorts of enclosures and leave plenty of space for a 'brick' style 12 V/2 A power adapter or a power transformer with rectifier, stabiliser, etc. to give it some weight! The prototype of the solder station pictured here was built into a Teko type 102 case with dimensions 110 × 70 × 46 mm.

Along that same thought and for reasons of comfort you may want to build the solder station board into a heavy metal case (no pun intended). If the case is too light it will move around too easily, whereas you want it to stay put on your bench. A good solution therefore would be to build the module with its power supply — using a heavy transformer — into a single case.

Users opting to build a discrete 12-V, 2-A power supply for the solder station, must observe all electrical safety precautions applicable in their region or country. That's why Elektor Labs recommends the use of a sealed 12-V DC supply of the type used for printers and lighting as that largely eliminates the safety hazard. So off to the local charity shop to find a suitable 'brick'!

Another light challenge is to find a cable that's:

- flexible;
- heat- and spillage-resistant;
- specified for the rated current;
- solderable to 3.5-mm jack plugs.

We found USB cables quite suitable — simply cut off the connectors and use two wires for ground and power.

One idea is to mount a socket on the enclosure and run wires from it to the module. The solder tip connects to the cable and the cable in turn connects to the socket.

In short, make your DIY solder station look like a commercially available product; there are good reasons why they all look much the same.

Using the solder station

The solder station is pretty easy to use as all you have to do is power it and set a target temperature by turning the rotary encoder. Any value from 50 °C to 450 °C can be set. The target temperature is displayed together with the measured temperature of the solder tip.

If the rotary encoder is not operated for about 15 minutes, the solder station will enter its power-saving mode. Push the encoder button to wake it from slumber. Pushing the encoder button for 10 seconds will also put the solder station to sleep. ⏮

180348-01

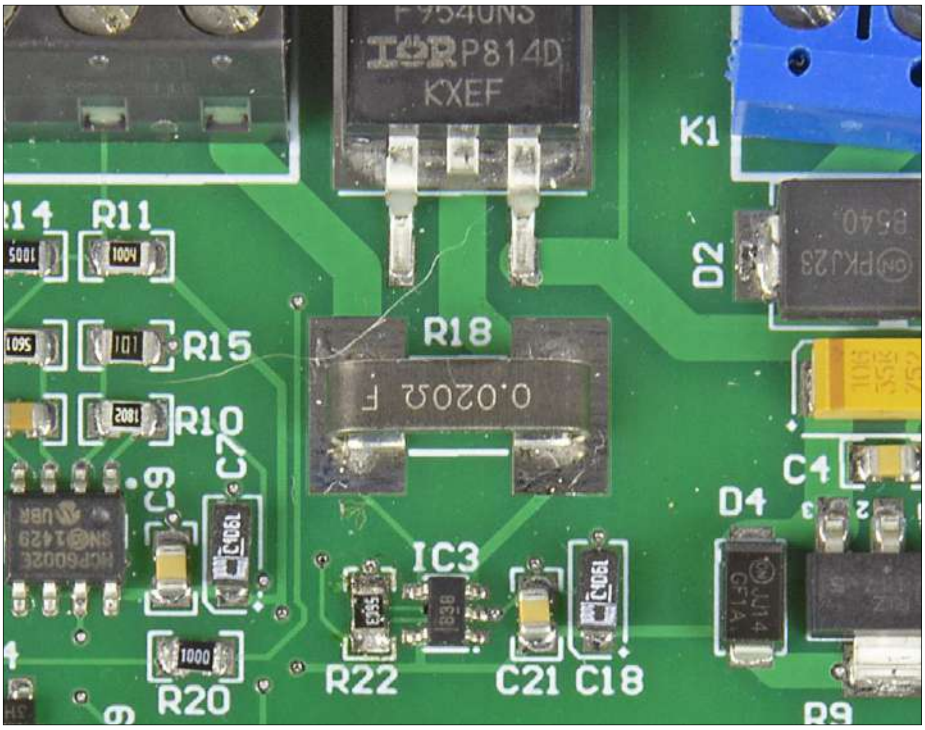


Figure 6. Close up of the 0.02-ohm, 2-watt current sensing resistor on the prototype.

@ WWW.ELEKTOR.COM

→ DIY Temperature Controlled Solder Station, bare PCB:
www.elektor.com/diy-solder-station-bare-pcb

→ DIY Temperature Controlled Solder Station, kit of parts
www.elektor.com/diy-solder-station-kit-of-parts

→ DIY Temperature Controlled Solder Station, programmed microcontroller
www.elektor.com/diy-solder-station-controller

Weblinks

[1] Platino-based Solder Station, original publication
www.elektormagazine.com/140107

[2] Downloads for this article
www.elektormagazine.com/180348-01

electronica
a glimpse of the

By **Clemens Valens** (ElektorLabs)

The second edition of electronica Fast Forward, the startup platform powered by Elektor — organised in collaboration with and staged at the world's most important electronica trade show in Munich — will remain in the yearbooks thanks to the high technical and professional level of the participating companies and their projects.

Each of the 32 contestants from no fewer than 10 countries was given five minutes to pitch their project to a Jury consisting of experts from (in alphabetical order) Arrow, Avnet-Silica, EBV Elektronik, and of course, yours truly, Elektor.

Three intense days of presentations resulted in six finalists: Teiimo, Querom, V-Juice, ValCUN, Volabo, and Wizama. The next day the 'wildcard' was awarded to Wisebatt, raising the number of finalists to seven in total. After the final round of pitches, Teiimo — a start-up company specialising in wearable electronics and smart textiles — was unanimously chosen as the overall winner.

During the three days of the event, with sponsorship from Arrow, Avnet-Silica & EBV Elektronik (Diamond class) and Trinamic (Bronze class), both the audience and the Jury were given an interesting and impressive view of what the future may hold. Subjects as varied as 3D metal printing and interactive board games, from laser-printed strain gauges to online

Markus Strecker from **Teiimo GmbH** — Technologies in Motion, overall winner of electronica Fast Forward, the startup platform powered by Elektor 2018 — received the Grand Prize from electronica CEO Dr Martin Lechner.

With their innovative interactive board game Second Prize winner **Wizama** hopes to recreate the cosy family atmosphere our parents or even grand parents will remember from the old days, meaning pre-smartphone and -tablet.

Fast-Forward 2018

future as imagined by smart start-ups

The Winners

1. €75,000 media budget and a booth at electronica 2020: **Teiimo GmbH (Germany) — Wearable Electronics and Smart Textiles**
2. €50,000 media budget: **Wizama (France) — Interactive Board Game Console**
3. €25,000 media budget: **Querom Elektronik GmbH (Germany) — High-voltage DC/DC Converter Modules**

Honourable Mention ***(we want one too!)***
€750: Max & Michael Buschmann —
Cocktail Joe Automatic Cocktail Machine

fast-prototyping services, cocktail robots and virtual reality systems, were presented by contestants from as far as Australia and the USA, and, Europe for sure. It sure was an honour and pleasure to meet such inspired people and discover what they are up to in terms of electronic product development. Will you be there too next time?

180702-01

Web Link

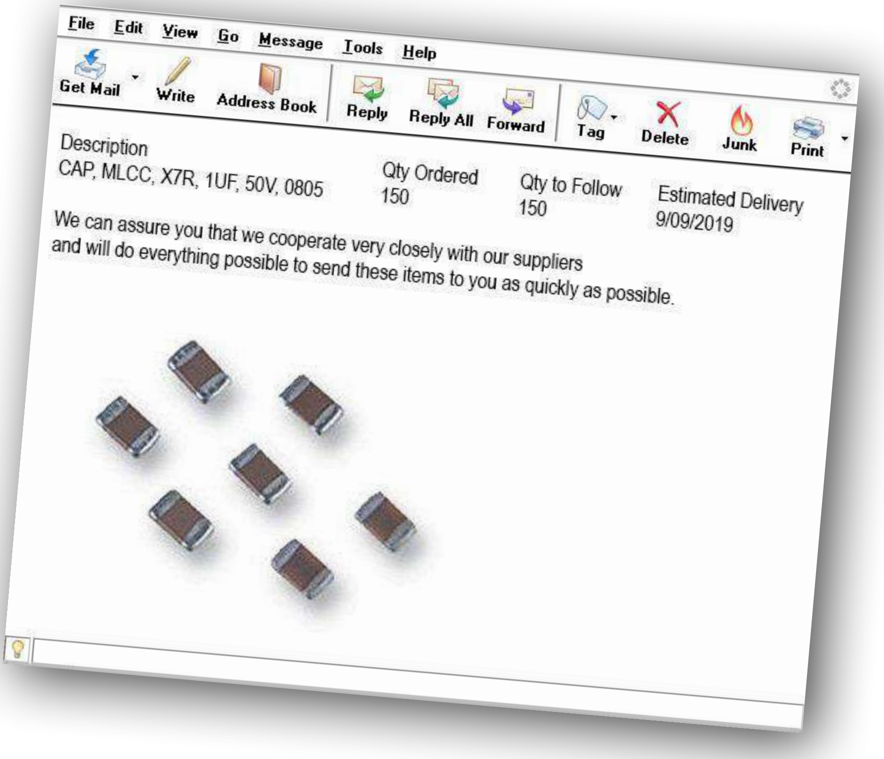
[1] electronica Fast Forward 2018 main contest page: www.elektormagazine.com/labs/contest/e-ffwd-contest

Third Prize winner **Querom Elektronik GmbH** specialises in custom power electronics and high-voltage DC/DC converters with output powers scalable from 1 kW up to 10 kW, input voltages from 260 VDC to 900 VDC and output voltages from 12 VDC to 48 VDC.

Cocktail Joe, an automated cocktail mixer that's sure to draw attention at parties.

Obsolete or “Allocated”?

By **Jan Visser** (Elektor Labs)



Description	Qty Ordered	Qty to Follow	Estimated Delivery
CAP, MLCC, X7R, 1UF, 50V, 0805	150	150	9/09/2019

We can assure you that we cooperate very closely with our suppliers and will do everything possible to send these items to you as quickly as possible.

Wednesday, around 3 pm: time to quickly order some components for a few current projects. But first I look through our stock of SMD components. Fortunately, this check reveals that we are only missing a few standard components in our SMD trays — some 0805 resistors with values of 1 kΩ, 10 kΩ, 100 kΩ, etc., as well as some 0805 capacitors with values of 1 nF, 10 nF and 100 nF. On my wish list for the projects there are several ‘heavy duty’ microcontrollers and other exotic components, for which I might encounter some delivery problems, but that can wait. The first thing is to order the standard components from the usual major suppliers, then that’s out of the way.

Two hours and several cups of coffee later, it turns out that the exotic components and microcontrollers are not the problem, but instead the ‘normal’ components are difficult to obtain! The exotic components and ‘difficult’ microcontrollers are available from various suppliers in all sorts of packages and in any desired quantities, but the availability of the standard components is really dismal.

Ordinary 0805 SMD resistors and capacitors suddenly pop up as having delivery times of several weeks to a few months, or the delivery time is stated as ‘expected’ — without a date, so you don’t know when it will be. Some suppliers kindly send you a message with the expected delivery time after you have ordered a component. For example, I recently received an email about a 1 µF capacitor that would soon be available again: expected delivery date late 2019! From enquiries at various suppliers and a bit of phoning around with contacts in the industry, I learned that stocks of ordinary components had been bought up on a massive scale by manufacturers of mobile phones, televisions and other electronic devices, in order to meet the growing consumer demand.

And then there are the brokers, who buy up scarce stocks and then start acting as though they were trading securities on the stock market. A downside of economic growth, I suppose. But these brokers, who buy up component stocks and keep them in a warehouse until the asking price has doubled or even risen by a factor of 10, have no interest in electronics. All they want is to earn as much as possible from their investment. Unfortunately, a consequence of their purchasing strategy is that even the most common components *appear* to be obsolete. In reality, these components are still there, but not in the right place: they have been “allocated”. For the reasonably normal mortals among us, or for small labs or the maker community, all this makes it difficult to obtain components unless you start negotiating for quantities of ten thousand or you are prepared to pay exorbitant prices for a few dozen resistors.

I have been working at Elektor for a very long time, and could not help being reminded of the article ‘Part Mining’ in January 2006 that warned of scarcity in the components market and recommended unsoldering ICs, microcontrollers and exotic components from used circuit boards so they could be reused. The article said that removing low-value components such as resistors or capacitors was not worthwhile because they were always readily available.

Today we should take a completely different view of the situation. In some large cities in China and India, there are already entire back alleys where countless people are busy all day desoldering all sorts of SMD components, so they can then be sold as ‘refurbished’ in shops on the main street. Even simple standard resistors and capacitors are not overlooked in that process. Perhaps we will soon see this phenomenon making its appearance in the streets of Amsterdam, London or Munich. In any case, nowadays it pays to not discard your old projects, but instead occasionally have a look at them to see if you can unsolder some simple components here and there. Of course, that raises the question of whether you previously used a particular component in some other project. And if so, where is the circuit board now? That brings us back to where we started, because the question is still the same: obsolete or “allocated”?

Kind regards from the Elektor Labs,
Jan Visser

180308-02

HOMELAB PROJECT

Lighthouse 2.0

With adjustable light characteristic

By **Friedrich Lischeck** (Germany)

In the November 2011 issue Elektor we presented a circuit by Leo Szumylowycz that simulated the rotating beacon of a lighthouse [1]. That circuit used only analogue components, such as the legendary 555 timer chip and a common LM358 opamp. It worked perfectly. The only disadvantage was the number of components that were required — a total of 29. Using an ATtiny45 microcontroller the expenditure on components can be drastically reduced. The circuit presented here requires only eight parts yet offers an extensive range of functions.

The rotating light of a lighthouse is perceived as a slow increase in light intensity when the light beam is moving towards you. Then follows a short, bright flash as the beam is pointed directly towards you, and subsequently the light intensity reduces again. And all this in a *light characteristic* (i.e. with shorter and longer intervals) that is specific to a lighthouse. This 'signature' can be approximated electronically with a triangular signal with a strong impulse superimposed at the peak. In this project we realise this in a BASCOM programmed microcontroller.

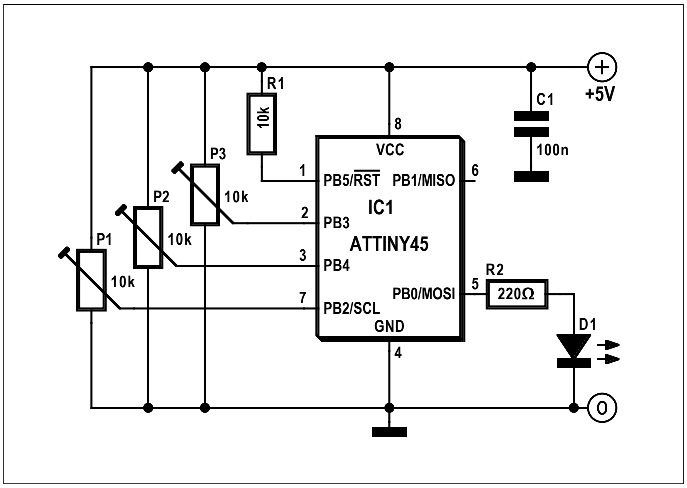


Figure 1. The schematic containing only eight parts is aptly described as extremely simple.

Hardware

The circuit is very straightforward, as is so often the case with microcontrollers: two resistors, one capacitor, three potentiometers, one LED and, sure, one microcontroller. That's all you need (see **Figure 1**). Most of the work is done in software,so we can be very brief here. The three potentiometers are connected to analogue inputs ADC1, ADC2 and ADC3 of the ATtiny45. Resistor R1 ensures a clean reset. R2 limits the current through the LED (D1) while C1 suppresses any noise. The following parameters can be set on the three potentiometers:

- P1: the virtual lighthouse can blink (light on and off) or flash (increase to a certain intensity and then suddenly flash and then reduce in intensity again, such as the circuit by Leo Szumylowycz used to do). P1 determines the ratio between the time of brightening and extinguishing and the bright flash, and consequently also sets the duration of the flash. At one end position you will have only increase and decrease, and at the other end, blinking only. The total duration of the cycle is always one second.

- P2: the complete cycle of brightening and extinguishing is called the *light characteristic*. If the light, for example, is on for 1 second, off for 1 second, on for 1 second and off for 10 seconds and then repeats again that's considered a light characteristic of 13 seconds. In contrast with the original circuit from 2011, using P2 it is possible the set the duration of the light characteristic.
- P3: this sets the number of flashes in the light characteristic. This function didn't exist in the original circuit either. If the light characteristic is shorter than the total time necessary for the number of flashes, then the duration of the light characteristic is automatically adjusted to the total duration of the selected flashes plus 2 seconds.

Using these options and controls it is possible to simulate many existing lighthouses.
Software
As already mentioned, most of the work is done in the software. The program is written in BASCOM-AVR. The source code is available from [2]. Here we highlight a few of the more interesting parts of the program.
Lighthouse 2.0 requires both Timer0 and Timer1. Timer0 controls the pulsewidth modulation (PWM) that sets the brightness of the LED. Timer1 is the clock generator that is used to set the desired On times. The program contains a peculiarity in the timer programming. Because BASCOM cannot control Timer1 of the ATtiny45, the register TCCR1 of the ATtiny45 is accessed directly:

```
Preload1 = 99 'corresponds to 0.01 seconds
TCCR1 = &H0A 'instead of Config Timer1=TIMER, Prescale
= 512
on Timer1 Zeitbasis1
enable Timer1
enable Interrupts
```

Remarkably, a prescale value of 512 is valid — a value that's not available in other current microcontrollers from Atmel, such as the ATmega8. Zeitbasis1 ('time base') is an interrupt-driven time routine. Timer0, as already stated, is used for the pulsewidth modulation and is configured in the normal way though BASCOM:

```
Config Timer0 = Pwm , Compare_A_Pwm = Clear_Up , Com-
pare_B_Pwm = Clear_Up , Prescale = 1
Enable Timer0
```

`Zeitbasis1` is the core of the program. Let's take a look at that (see **Listing 1**). The interrupt routine `Zeitbasis1` uses the variables `LEDAn` ('LED on'), `Blitzstart` ('flash start') and `Blitzende` ('flash end'), which are all set by the main program. The routine also uses the array variable `Form(N)`, which is initialised in the main program and then doesn't change. It passes the variable `Zehntel` ('tenth') to the main program.
The interrupt routine is called every one-hundredth of a second and is divided into two parts. In the first part the variables `Hunderstel`, N and `Zehntel` are incremented. `Hundertstel` represents one-hundredth of a second and `Zehntel` represents one-tenth of a second. N also represents hundredths of a second, but differs in the sense that N can have a value from 1 to 100, while `Hundertstel` goes from 0 to 9 and only serves to increment the tenths of seconds count in the variable `Zehntel`.

Listing 1

```
Zeitbasis1:
 Timer1 = Preload1
  incr Hundertstel
  incr N
  if Hundertstel>9 then
   Hundertstel=0
   Incr Zehntel
 end if

 if N >100 then N=1
 if LEDAn=1 then
  Ausgabe=Form(n)
  if n>= Blitzstart and n<= Blitzende then
   Ausgabe=255
  OCR0A= Ausgabe 'Set new brightness
 else
  OCR0A=0
 end if
Return
```

The second part of the routine handles the brightness control of the LED. Every 100th of a second, represented by variable N, a new brightness value is set, but only when the LED is on at that moment, which is indicated by the variable `LEDAn` in the main program. The brightness values are stored in `Form(N)`, where N increments by one every 100th of a second. Together, all 100 values of N therefore add up to 1 second — the duration of the on-state of the LED. The brightness value from `Form(N)` is copied into the variable `Ausgabe` ('output') each time.
With the above, the brightness of the LED only increases and decreases. Now we add the variables `Blitzstart` and `Blitzende`. These also contain values between 1 and 100, corresponding to the 100th of seconds during the on-state of the LED. When N is greater or equal to the value `Blitzstart` and smaller or equal to the value of `Blitzende`, the LED is turned on at the maximum value ('255'). This value will then overwrite the current value of the variable `Ausgabe`. The new brightness value is set with `OCR0A= Ausgabe` (expressed in duty cycle) for the pulsewidth modulation.
As already mentioned, this only occurs when `LEDAn` indicates the on-state. When `LEDAn` indicates the off-state, then none of this happens and the value of the pulsewidth modulation is set to zero with `OCR0A=0`.

Main program

The main program runs in an endless DO-LOOP and contains two main function groups. In the first part the values of the three potentiometers P1 through P3 are read, and the three corresponding variables `Blitzstart/Blitzende`, light characteristic and identification duration are assigned and normalised with the required values. This, however, only occurs when the variable `LEDAn` has the value zero.
The second part of the main program (see **Listing 2**) controls the switching on and off of the variable `LEDAn` with the desired timing. Not every cycle through the do-loop should result in an operation. The intention is that an operation is performed

Listing 2

```
If Zehntel <> Lzehntel Then
 Incr X
  If X = Wiederkehr Then
   X = 1
   disable Interrupts
   N = 0
   enable Interrupts
  end if
  if X < Kennungsdauer then
   If X = Naechste Then
    Naechste = Naechste + 10
    if LEDAn = 1 then
     LEDAn = 0
    else
     LEDAn = 1
    end If
   end If
  End if
  If X = Kennungsdauer then
   LEDAn= 0
   Naechste=1
  end if
 Lzehntel = Zehntel
End If
```

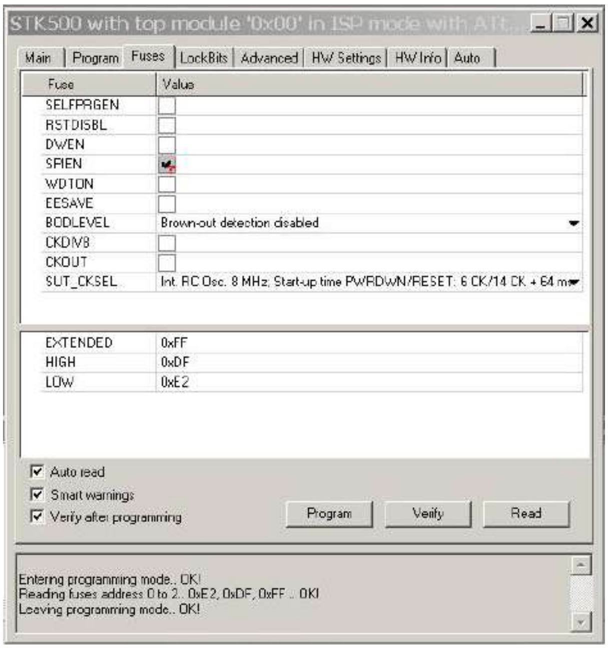
only once per tenth of a second. This is achieved by the two variables `Zehntel` and `LZehntel`. The name `LZehntel` represents the most-recent tenth in which any operation took place.

In each interval of one second, `LEDAn` is set to one or zero. The program uses the variables `Naechste` (‘next’) and X for this. X is increased by 1 every tenth of a second and only when it reaches the value of the light characteristic (‘`Wiederkehr`’) is it reset back to 1. With that, the variable N for the interrupt routine `Zeitbasis1` is synchronised and set to 0. To ensure that this does not happen while the interrupt routine is active, interrupts are disabled beforehand and re-enabled afterwards. When X reaches the value of `Naechste`, i.e. when one second has elapsed, the variable `LEDAn` toggles. In addition, the value of `Naechste` is increased by 10, what corresponds to activating the next operation after another second. However, this only happens as long as the value of X is smaller than the value of the identification duration (‘`Kennungsdauer`’). Once this is exceeded, `LEDAn` is set to 0 and `Naechste` to value 1, so that `LEDAn` will only change at the beginning of a new cycle.

For the duration of the on-state the following code is used:

```
for I=1 to 10: Form(I)=1:Next I
for I=11 to 20: Form(I)=2:Next I
for I=21 to 30: Form(I)=3:Next I
for I=31 to 40: Form(I)=4:Next I
for I=41 to 50: Form(I)=5:Next I

For I = 51 To 100
 J = 101 - I
 Form(i) = Form(j)
Next I
```

The values of elements 1 to 50 are initialised in blocks of 10. The values for elements 51 to 100 are these same values, but copied in reverse (that is, from 50 to 1). So, in the end, the



Fuse	Value
SELFPRGEN	□
RSTDISBL	□
DWEN	□
SPIEN	☑
WDTON	□
EESAVE	□
BODLEVEL	Brown-out detection disabled
CKDIV8	□
CKOUT	□
SUT_CKSEL	Int. RC Osc. 8 MHz; Start-up time PWRDWN/RESET: 6 CK/14 CK + 64 ms

EXTENDED	0xFF
HIGH	0xDF
LOW	0xE2

☑ Auto read
☑ Smart warnings
☑ Verify after programming

```
Entering programming mode.. OK!
Reading fuses address 0 to 2.. 0xE2, 0xDF, 0xFF .. OK!
Leaving programming mode.. OK!
```

Figure 2. Here the fuse settings for the ATtiny45 are shown as they are displayed in AVR-Studio 4.18.

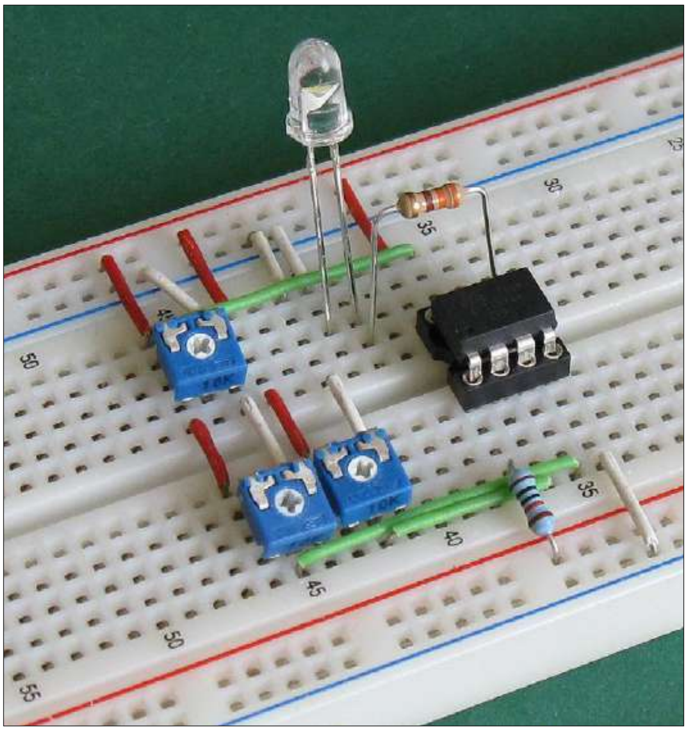


Figure 3. The circuit is easily built on a breadboard.

array variable `Form(I)` therefore contains an approximate triangle wave with a peak value of 5.
For the circuit to work properly, the fuses of the ATtiny45 have to be set to the following values:

```
EXTENDED: 0xFF
HIGH : 0xDF
LOW : 0xE2
```

See also **Figure 2.**

Construction

The circuit can be built on a piece of prototyping board or, as can be seen in **Figure 3**, on a breadboard. It is advised to use an IC socket. Very practical are those sockets with a built-in capacitor. If used, C1 does not need to be fitted — just as in Figure 2.

Adjustment

First set all three pots to their lowest value. Now, on P1 set the desired flash duration. Next it is recommended to set the number of flashes per cycle with P3. Because the light characteristic is still set to the minimum duration, the program automatically extends this by two seconds more than what the flashes and accompanying interruptions require. Once you have set the number of flashes and the flash duration, you use pot P2 to set the duration of the light characteristic to the desired value. Now the parameters for the lighthouse are all set.

180334-02

@ WWW.ELEKTOR.COM

→ E-Book: Microcontroller Basics
www.elektor.com/microcontroller-basics-ebook-en

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

Web Links

[1] Lifelike Lighthouse (Elektor November 2011): www.elektormagazine.com/magazine/elektor-201111/19764

[2] Project files (free downloads): www.elektormagazine.com/180334-02

Post your ideas and
electronics projects
all sizes/all levels/allsorts,
at www.elektor-labs.com
and become famous!
Elektor Labs: www.elektor-labs.com
Design, Build, Share... Electronics!

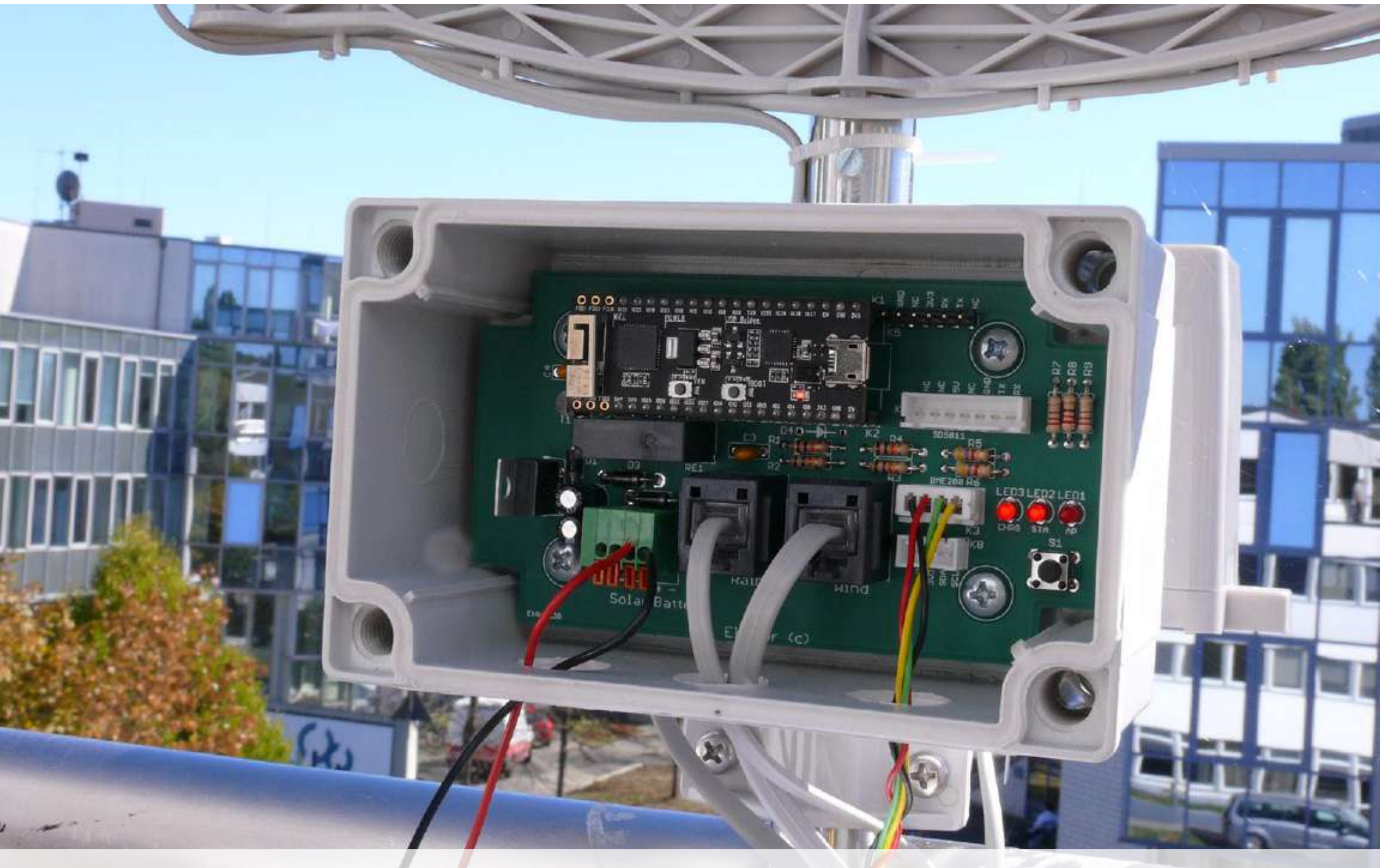
eLABS PROJECT

ESP32 Weather Station

Reading sensor data online

By **Roy Aarts** (Elektor Labs) and **Thijs Beckers** (Editorial)

The ESP32 board is a versatile and very affordable development platform that is quite suitable for all sorts of home automation projects. Here we present a specific application with a weather station built around the ESP32, which makes all data directly available online.



[Our prototype fits perfectly into the chosen housing. Note: pictured is a standard 7805 on the PCB instead of the DC/DC converter].

Features

- Measures temperature, wind direction and wind speed, humidity, air pressure and precipitation
- Optional sensor for fine particles: Nova Fitness SDS-011
- Additional ports for Grove sensors or other devices
- Supports Thingspeak and senseBox
- Can be configured on the internal web page of the ESP32
- Operates from a solar panel, 12 V battery, and/or 8-28 V DC adapter

The ESP32 Weather Station measures the usual weather parameters: temperature, wind direction and wind speed, humidity, air pressure and precipitation. With a suitable sensor, such as the Nova Fitness SDS-011 [1] (PM2.5, resolution 0.3 μg/m^3), it can also measure the concentration of fine particles. A Bosch FME280 sensor [2] is used to measure the temperature, humidity and air pressure. For wind speed, wind direction and precipitation we use a weather station kit that is available in the Elektor Store (see the '@ www.elektor.com' inset).

The ESP32 makes the measurements and can upload the resulting data to Thingspeak [3] or senseBox [4]. Thingspeak is an online database operated by Mathworks, where you can upload data and view it nicely plotted in charts. The data can also be processed with Matlab.

SenseBox acts as a sort of open-source synoptic weather map. Users who have a senseBox kit or other senseBox compatible device can upload their measurements, which are then displayed on a world map and visible to everyone.

Hardware

The weather station is built around the ESP32 Pico Kit V4 (see the schematic in **Figure 1**, which handles all the necessary tasks. To make sensor connection and fitting in a waterproof enclosure (a Fibox PC 100/60 HT) a bit easier, we designed a carrier board. The wind and rain sensors are connected to the carrier board through RJ45 connectors, while the BME280 and SDS011 sensors are connected through JST XH connectors. The BME280 sensor uses the I2C bus, and the SDS011 sensor uses a UART port. We also implemented two additional ports on the board to allow supplementary sensors (or other sensors) or peripheral devices to be read. The first additional port consists of a Grove I²C connector for Grove modules from Seeed studio [5, 6]. The second is an FTDI port that feeds out a UART connection.

The circuit can be powered from a 12 V lead-acid battery. If you would like to use a 12 V solar panel to charge the battery, we have a handy built-in feature for you: the ESP32 can close a relay when the battery is low and open it when the battery is again fully charged. The ESP32 uses a voltage divider with resistor values of 470 kΩ and 100 kΩ to measure the battery voltage, so you can adapt the start and stop points for battery charging in the software if necessary. LED3 indicates the battery charging status. We use a simple FET (T1) as a buffer to drive the relay. Diode D3 acts as a snubber to suppress voltage spikes from the relay coil and protect T1 against premature failure. Resistor R10 limits the base current, while R7 limits the current through

PROJECT INFORMATION

sensors
measurement

entry level
➡ intermediate level
expert level

4 hours approx.

Standard tools (including soldering tools)

€90 / £80 / $100 approx.

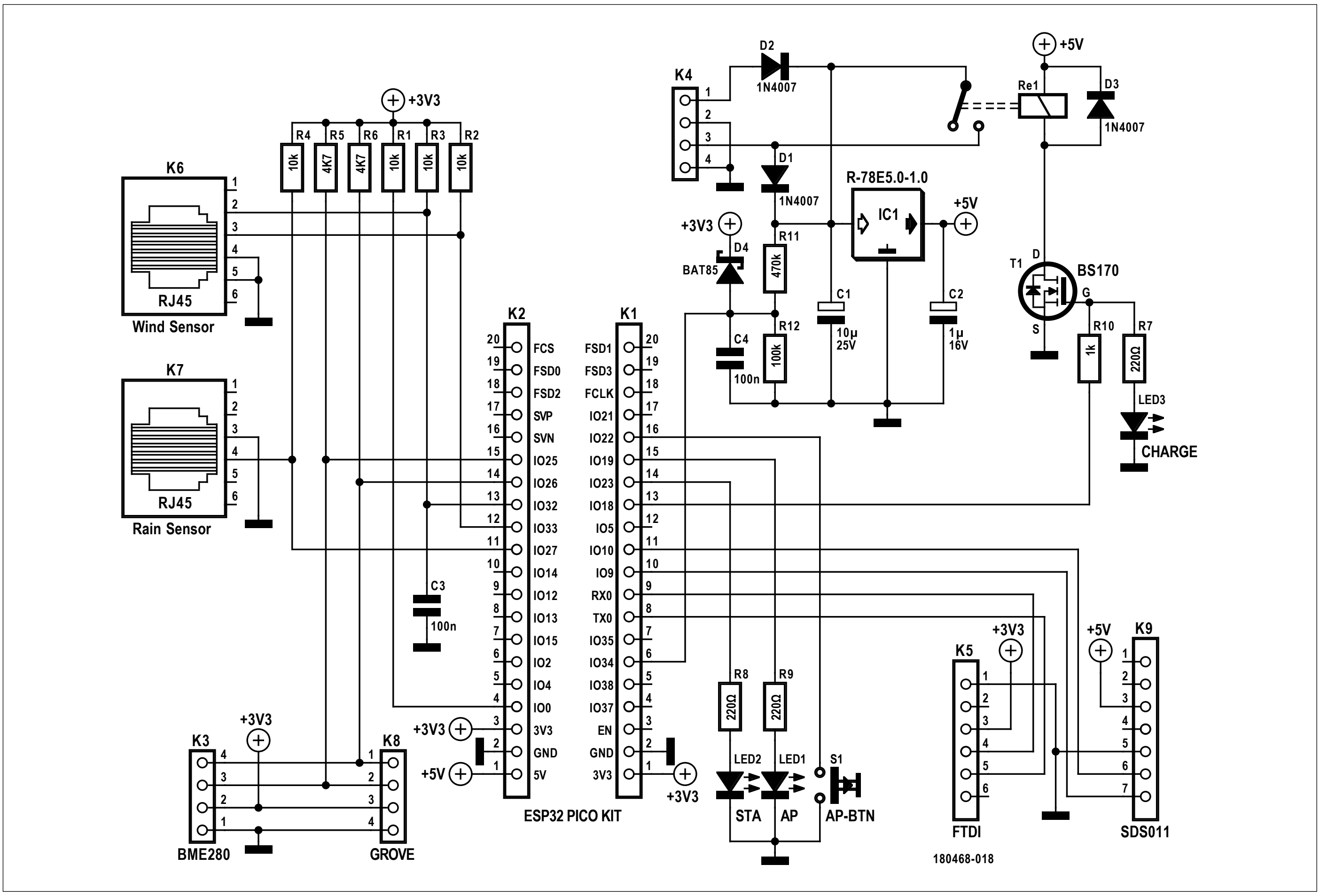


Figure 1. As you can see from the schematic, the ESP32 is the heart of the weather station.

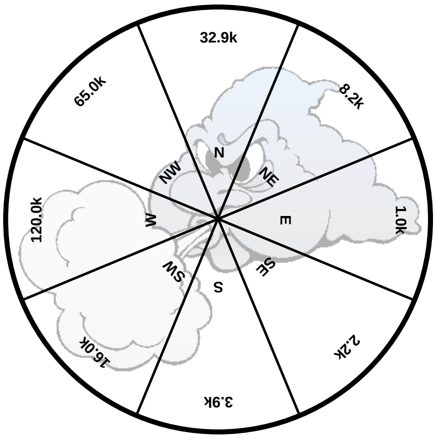


Figure 2. The internal resistance of the wind vane depends on the wind direction.

Sensor data accessible worldwide

LED3, as do resistors R8 and R9 for LED2 and LED1, respectively.

A DC/DC converter (IC1) reduces the power supply voltage to 5 V. This converter is a good deal more efficient than a standard 7805 voltage regulator, so it is a better choice for a battery-powered circuit. On the ESP32 board, the 5 V supply voltage is converted to 3.3 V by a type 1117 voltage regulator IC, and this voltage is available on the board's connector for use elsewhere (that's where the 3V3 in the schematic comes from). Diode D4 protects the ESP32 against any overvoltage from the battery (via voltage divider R11/R12), while capacitor C4 suppresses noise spikes to foster stable readout of the battery voltage.

Resistors R4 through R6 are pull-up resistors that are necessary for proper operation of the corresponding signal lines.

Rain and wind sensors

The rain sensor consists of a small tray that fills with rainwater. When it is full, it tips and briefly closes a reed switch contact that is monitored by the ESP32. According to the datasheet, each tip of the tray corresponds to 0.33 mm of precipitation.

The wind sensor consists of a wind vane for measuring the wind direction and an anemometer for measuring the wind speed. The internal resistance of the wind vane changes depending on the wind direction. **Figure 2** shows the layout for the various compass directions. We combined a 10 kΩ resistor (R2) with this wind direction dependent resistance to form a voltage divider. The resulting voltage changes when the wind vane sensor resistance changes, and it can be measured by the ESP32.

The anemometer has a reed switch that generates two pulses for each rotation. According to the data sheet, the wind speed in m/s is equal to the pulse frequency (in Hz) multiplied by 0.33. For determination of the rotation frequency, the data sheet says that the time interval between two pulses (in seconds) should be multiplied by 2 (since the sensor generates two pules per rotation), and then the rotation frequency can be calculated as the inverse of this time interval. If you multiply that value by 0.66, you get the wind speed in metres per second.

The wind speed in km/h can be calculated by multiplying the speed in m/s by 3.6. In other words, 1 rps corresponds to a wind speed of 2.4 km/h.

Uploading the software

In order to upload the software to the ESP32, you first have to install the ESP32 Arduino core [7]. Then you have to install the SPIFFS download tool in order to

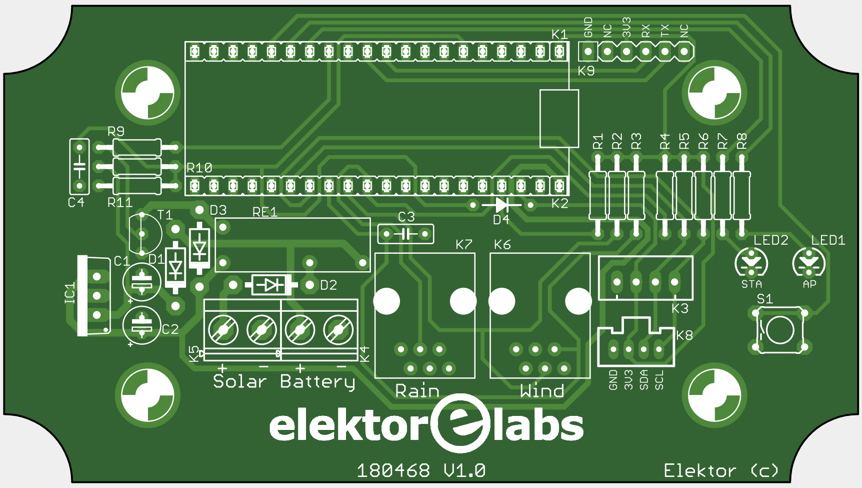
COMPONENT LIST

Resistors
R1-R4 = 10kΩ
R5,R6 = 4.7kΩ
R7-R9 = 220Ω
R10 = 1kΩ
R11 = 470kΩ
R12 = 100kΩ

Capacitors
C1 = 10µF, 50V
C2 = 1µF, 10V
C3,C4 = 100nF

Semiconductors
D1-D3 = 1N4007
D4 = BAT85
IC1 = 5V DC/DC converter, 5W, R-78E5.0-1.0 Recom Power
LED1, LED2 = yellow, 3mm
LED3 = red, 3mm
T1 = BS170

Miscellaneous
K1, K2 = 17-pin (1×17) stacking header, 0.1" pitch
K3 = 4-pin pinheader, 0.1" pitch
K4 = 4-way barrier strip terminal block, 0.1" pitch (Würth 691403900004B)
K5 = 6-pin pinheader, 0.1" pitch
K6, K7 = RJ11 jack, Molex RJ11 6P V
K8 = Grove connector, vertical, TWIG-4P-2.0-2.0
K9 = 7-pin pinheader, 0.1" pitch, JST-XH-07-PIN-VER
RE1 = relay ALDP105
S1 = pushbutton, 6x6mm
BME280 breakout board, I²C version
Wind and precipitation sensor
Enclosure PC 100/60 HT, Fibox
PCB 180468-1
Optional: fine particle sensor, Nova Fitness SDS-011



download the web page to the file system of the ESP32 [8]. Once everything is correctly installed, you can select the ESP32 Pico Kit in the boards menu and then download the web page and the sketch.

Software

As previously mentioned, the weather station is controlled by the ESP32. You can configure it on a web page hosted directly by the ESP32. When the ESP32 starts up, it tries to connect to the configured network. If that fails, the ESP32 launches the web server that hosts the configuration page. The web server can also be launched manually by pressing the button on the board while switching on power, or by pressing the EN button on the ESP32 module. If the user has not done anything on the configuration page for ten minutes, the ESP32 restarts and connects to the configured network. You can suppress this by keeping the button pressed or bypassing the button.

On the configuration page you can view the current measurements as well as the measured battery voltage. You can also configure the network settings and the upload settings. The network settings include the SSID (name) of the network, which can be selected from a list, and the password. The upload settings include the required API keys for Thingspeak and senseBox, as well as the upload interval.

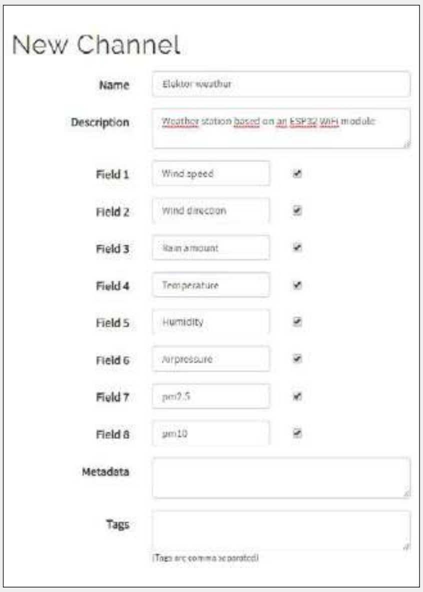
To obtain access to the web page, you have to connect to the ESP32 network provided by the ESP32. Then you can open the web page at the IP address 192.168.4.1. Note that with smartphones (e.g. Android) you must temporarily disable mobile data, because smartphones often switch automatically to mobile data if they do not detect an valid Internet connection via WiFi. In that case the configuration page will not be available.

The web page consists of a single HTML file, which in turn consists of three sections. The <style> element at the top defines the appearance of the page, the <body> element contains the structure and content of the page, and the <script> element at the bottom defines the functionality of the page. For the web page to work properly, javascript must be enabled in the web browser.

The communication between the web page and the ESP32 is done via an ‘asynchronous XMLHttpRequest’ on the web page. When the page is loaded, an http

Thingspeak

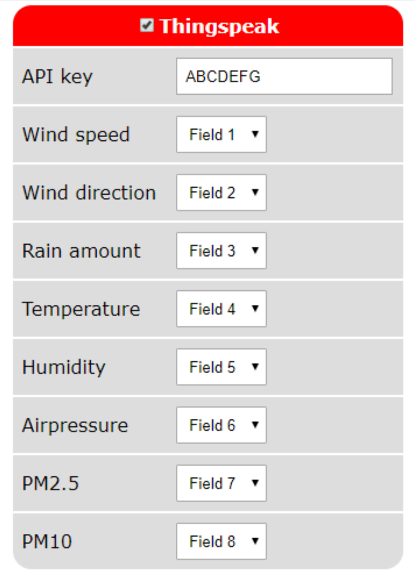
Thingspeak is a free online database from Mathworks to which data can be uploaded. For this, a channel and associated field must be created for each measured parameter (see **Figure 3**). After the channel is created, you receive a write API key (**Figure 4**). This key must be entered on the configuration page of the weather station.



New Channel

Name	Elektor weather	
Description	Weather station based on an ESP32 WiFi module	
Field 1	Wind speed	☑
Field 2	Wind direction	☑
Field 3	Rain amount	☑
Field 4	Temperature	☑
Field 5	Humidity	☑
Field 6	Airpressure	☑
Field 7	pm2.5	☑
Field 8	pm10	☑
Metadata		
Tags	(Tags are comma separated)	

Figure 3. At Thingspeak, you create a new channel for your application and enter the data types in the corresponding boxes.



☑ Thingspeak

API key	ABCDEFG
Wind speed	Field 1 ▾
Wind direction	Field 2 ▾
Rain amount	Field 3 ▾
Temperature	Field 4 ▾
Humidity	Field 5 ▾
Airpressure	Field 6 ▾
PM2.5	Field 7 ▾
PM10	Field 8 ▾

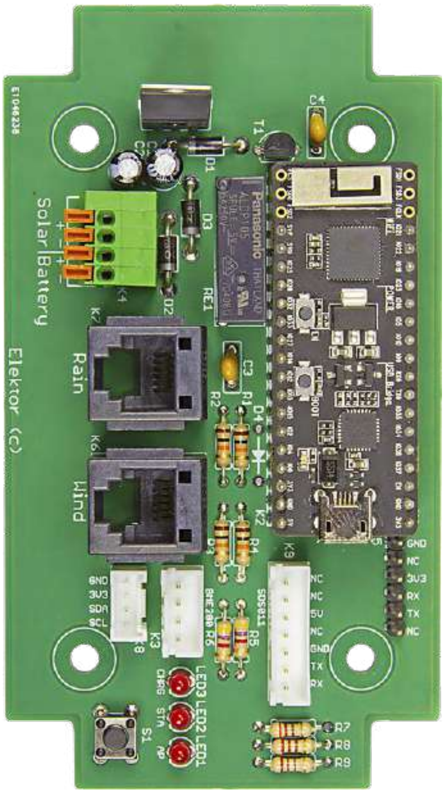
Figure 4. Then you receive an API key to be used in the software of your application.

request is generated and sent to the ESP32. The ESP32 receives this request and sends an answer back to the web page with the requested settings or values. For example, when you press a Submit button on the web page, a similar request is sent that now contains the values you entered.

Practical matters

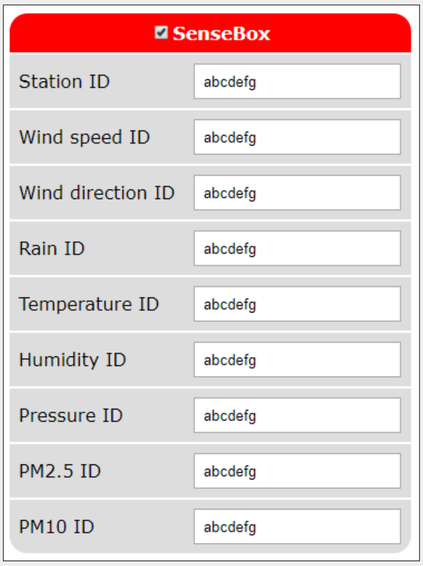
The only thing left to describe is how to set up the device and connect the sensors. In order to upload the data, it is of course necessary that the ESP32 can access your home network (or some other network). For setting up the sensors, you should also take into account the specific requirements of the sensors concerned. For example, putting the precipitation sensor close to a wall is not a good choice. For this you should be guided by your common sense. ◄

180468-02



senseBox

A senseBox kit makes it easy to set up your own sensor station. The measured parameters can be uploaded to opensensemap.org, where the measurements from all senseBoxes or devices compatible with senseBox can be viewed. To register your own weather station, you first create an account at opensensemap.org and then you can add a senseBox. You can select your specific senseBox type on the account creation screen, or you can configure your measurement station yourself by entering the measured parameters, the measurement units and the sensor types (see **Figure 5**). You have to create an account for the EPS32 weather station manually. After you register the weather station, you receive an ID for the station and a sensor ID for each sensor. Then you enter these IDs on the configuration page of the EPS32 weather station. After that it can upload the measured data.



☑ SenseBox

Station ID	abcdefg
Wind speed ID	abcdefg
Wind direction ID	abcdefg
Rain ID	abcdefg
Temperature ID	abcdefg
Humidity ID	abcdefg
Pressure ID	abcdefg
PM2.5 ID	abcdefg
PM10 ID	abcdefg

Figure 5. At opensensemap.org you can create your own sensors for the ESP32 weather station.

Web Links

[1] Inovafitness: www.inovafitness.com/en/a/chanpinzhongxin/95.html

[2] BME280 (Mouser): www.elektor.com/bme280-mouser-intel-i2c-version-160109-91

[3] Thingspeak: https://thingspeak.com

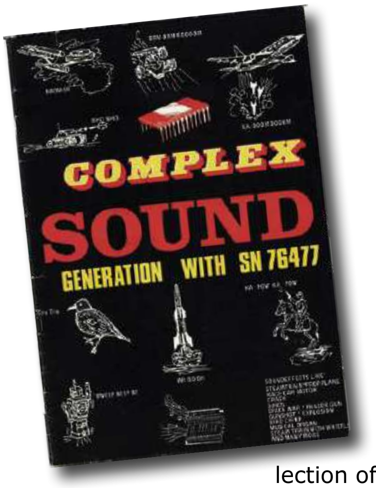
[4] Sensebox: https://sensebox.de/en

[5] Seeedstudio products at Elektor: www.elektor.com/seeedstudio

[6] Seeedstudio Grove: www.seeedstudio.com/s/grove.html

[7] Github for Arduino-Esp32: https://github.com/espressif/arduino-esp32

[8] Github for Arduino Esp32fs plugin: https://github.com/me-no-dev/arduino-esp32fs-plugin



The SN76477

Peculiar Parts, the series

Contributed by **Tore Bergvill** (Norway) – Series Editor: **Neil Gruending** (Canada)

I was cleaning up the basement and found a nice collection of old papers, one of which was an extensive printed manual for the SN76477 sound chip [1]. It was written using a typewriter and hand drawn illustrations which is something you just don't see today. I still have the SN76477 chip that I used to make a fantastic rumbling vibrato sound effect in a Ghostbusters costume Proton Pack.

Texas Instruments launched the SN76477 Complex Sound Generator 1978 and it was quickly adopted by several toy/arcade game vendors and hobbyists. Elektor even ran a couple of articles in 1978 and 1981 describing it as "a single IC full of surprises" with several circuit designs, PCB layout and development suggestions. The chip, shown in **Figure 1**, contained a super low frequency generator, a voltage controlled oscillator and a noise generator that you could control with a handful of passive components. Then the internal mixer and output envelope shaper would combine these signals to generate complex sound effects. Some of the reference designs in the TI manual [2] named the sounds effects Siren, Phasor gun, Race Car, Gun Shot, Explosion, barking Dog, Steam Train with Whistle and even the Bird Silencer with Explosion. The SN76477 proved to be excellent for generating sound effects but it was not very suitable for use in music instruments or synthesizers because it was difficult to control the overall pitch of the resulting sound output.

The SN76477 was used in many commercial products of the time. Several Taito arcade games used the chip although Space Invaders was the first and possibly most famous where the SN76477 generated the flying saucer sound effect. The Swedish company Luxor also used the SN76477 in their Z80-based ABC 80 home computer shown in **Figure 2**. Luxor was a major vendor of TVs and other home electronics in the Nordics and their ABC 80 mostly gained success in Sweden due to the large amount of Basic software available in Swedish language.

Texas Instruments soon made a more versatile sound chip called the SN76489 Digital Complex Sound Generator in 1980. It quickly gained popularity in computers and game systems such as the Acorn BBC Micro, ColecoVision, Sega Game Gear and many others. The SN76477 was discontinued in the 1990's but replacements are still available on eBay or elsewhere and there's a modern alternative called ICS76477 also on the market.

The SN76477's approach to generating sound seems outdated by today's methods but it paved the way for later analogue sound chip development such as the General Instrument AY-3 (Elektor TV Games Computer [3], TV Games, ZX Spectrum) and MOS Technology 6581 SID (Commodore 64) families of integrated circuits. Then came the digital era, but that is another story! ⏮

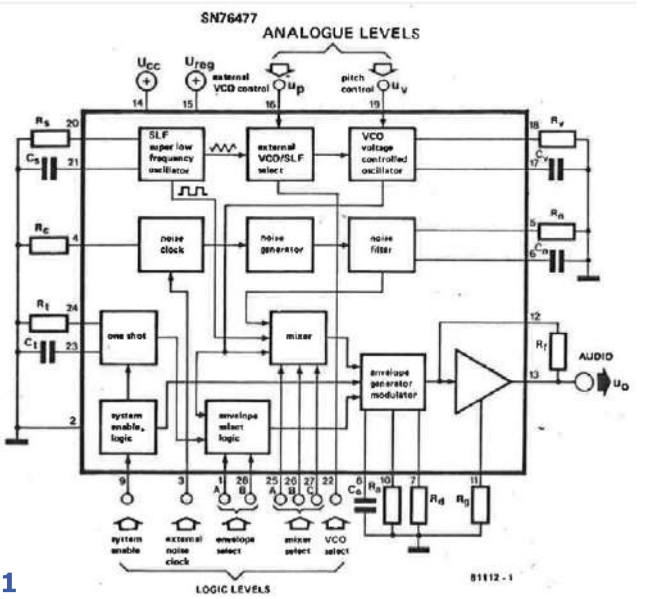
(160556)

Web Links

[1] Wikipedia entry on SN76477: https://en.wikipedia.org/wiki/Texas_Instruments_SN76477

[2] TI SN76477 manual: www.ece.usu.edu/ece_store/spec/SN76477.pdf

[3] Elektor TV Games Computer (Retronics series): www.elektormagazine.com/magazine/elektor-200810/18930



1

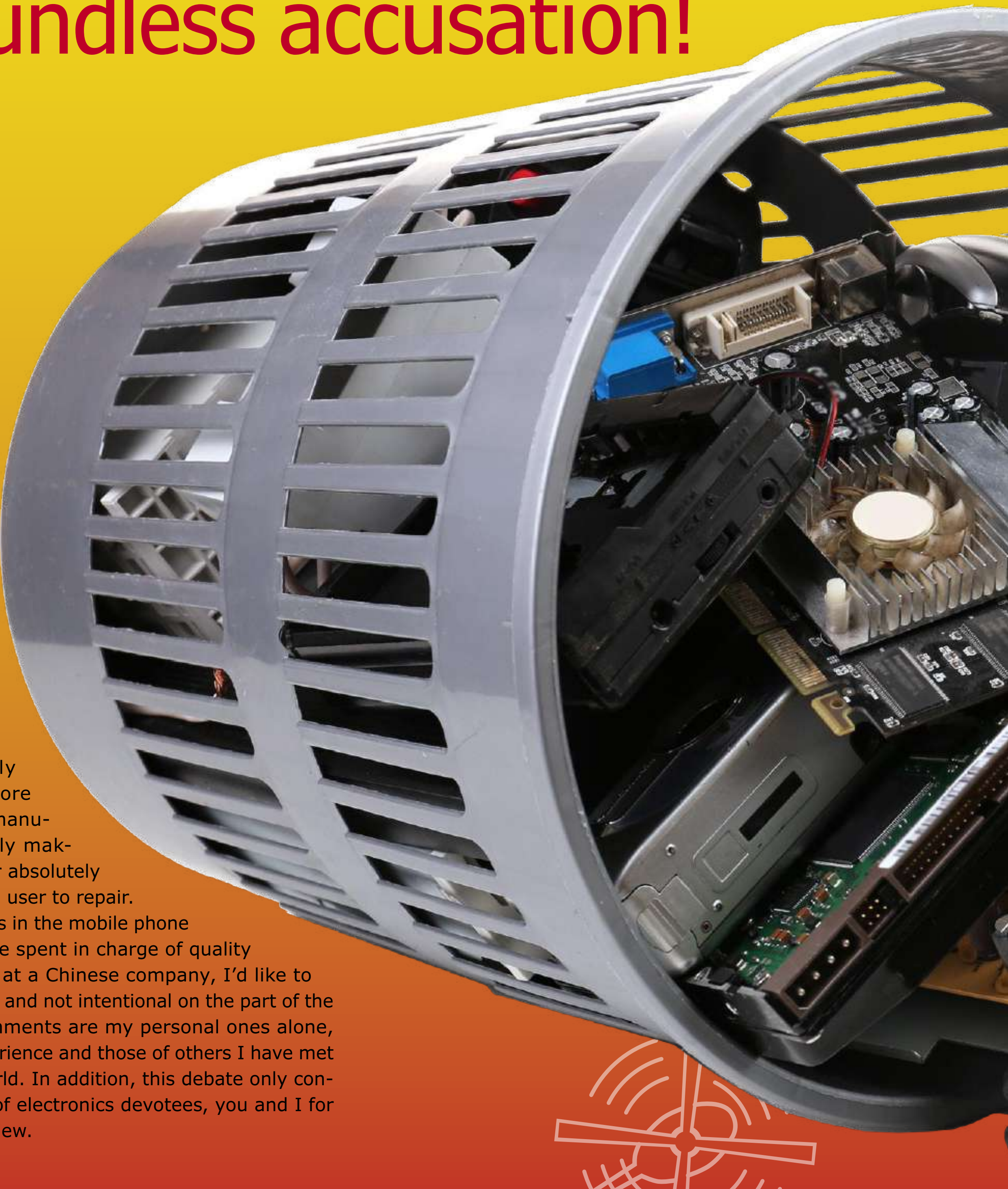
2

HomeLab Helicopter

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

Planned Obsolescence ?
A groundless accusation!

By our special representative
Laurent Labbé (France)



Bringing in the customers

You would have surely already read one or more articles which accuse manufacturers of deliberately making which are difficult or absolutely impossible for their final user to repair. After more than 30 years in the mobile phone industry, fifteen of those spent in charge of quality and after-sales service at a Chinese company, I'd like to explain why this is false, and not intentional on the part of the manufacturers. My comments are my personal ones alone, and come from my experience and those of others I have met in the mobile phone world. In addition, this debate only concerns the microsphere of electronics devotees, you and I for the purposes of this review.

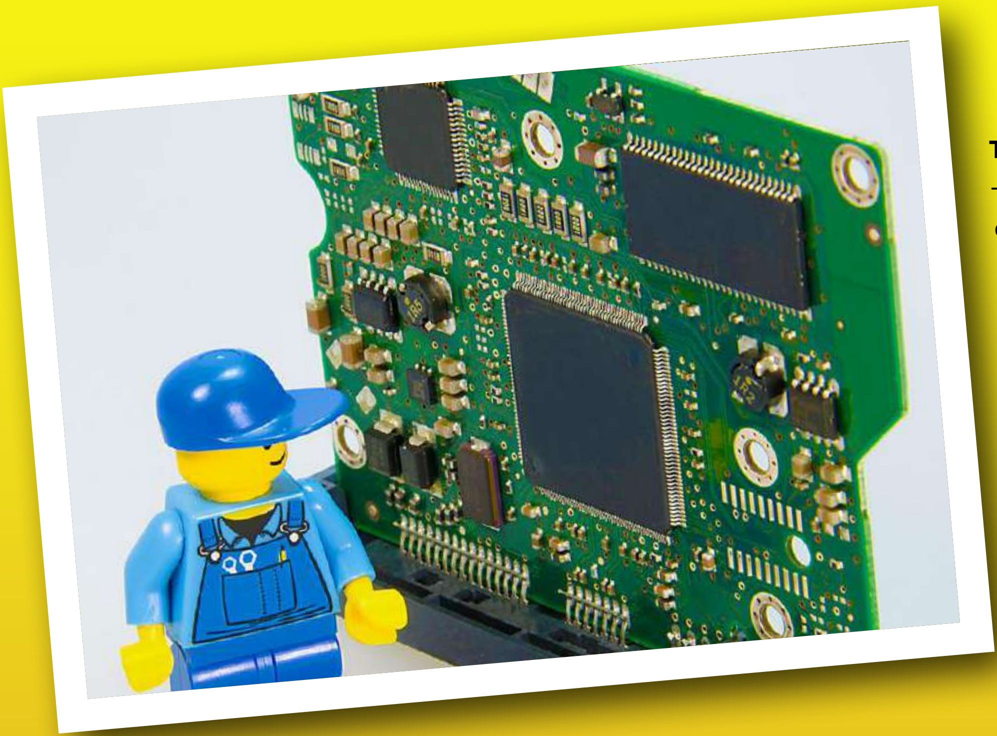
During the design of a mobile phone (or smartphone) a project team is responsible for the design, the production and the after-sales service. They are subject to constraints imposed by the success of the launch on the market. For a product to be successful, there are two primary factors: the look and the price. Ten years ago everyone was happy with a phone 20 mm thick and costing around 200 euros. Today's buyers are looking for something half as thick for half the price. In the mobile world, price and appearance are the dictating factors.

The project team thus has as their objective to create a sexier, thinner product with no screws, no battery cover (too thick) and overall as cheap as possible. As well as this, two team members will do all they can to make sure the product is repairable: the heads of production and of after-sales service. The factory will have to produce some tens of thousands of units per day and also repair production faults. Usually in a production line, around 90% of the products work first-time, but 10% will need to be repaired on specialised lines. Finally, after-sales service will have to take care of between 5% and 10% of the sold units. This budget item can 'explode' if the product is not repairable.

The design has a huge effect on the reparability. So how do you repair a touch screen in a fancy thin device? The touch screen is fixed with glue or double sided tape. It needs a heated table or similar to remove it. A phone with a glass back looks very nice, but it's irreparable by the end-user, because the glass is fixed the same way. The same part in ABS is fixed much more simply, but the phone won't look as nice, it will be thicker and the client won't buy it.

Playing with security

There is another factor to take into consideration: any 'dangerous' parts of the phone (charger, battery) must not be easy to remove by an ordinary user. You would be astonished at the number of people who believe they can fix anything (far more than the number who actually can). Now, is some countries like the U.S.A., if the user can do a simple operation with a common tool (like a screwdriver) and there is an incident, it is the manufacturer who is responsible. Take the case of a phone with a LiPo battery protected with a cover and a shield which can be removed. If the screening is fixed with simple visible screws and the user can remove them (in



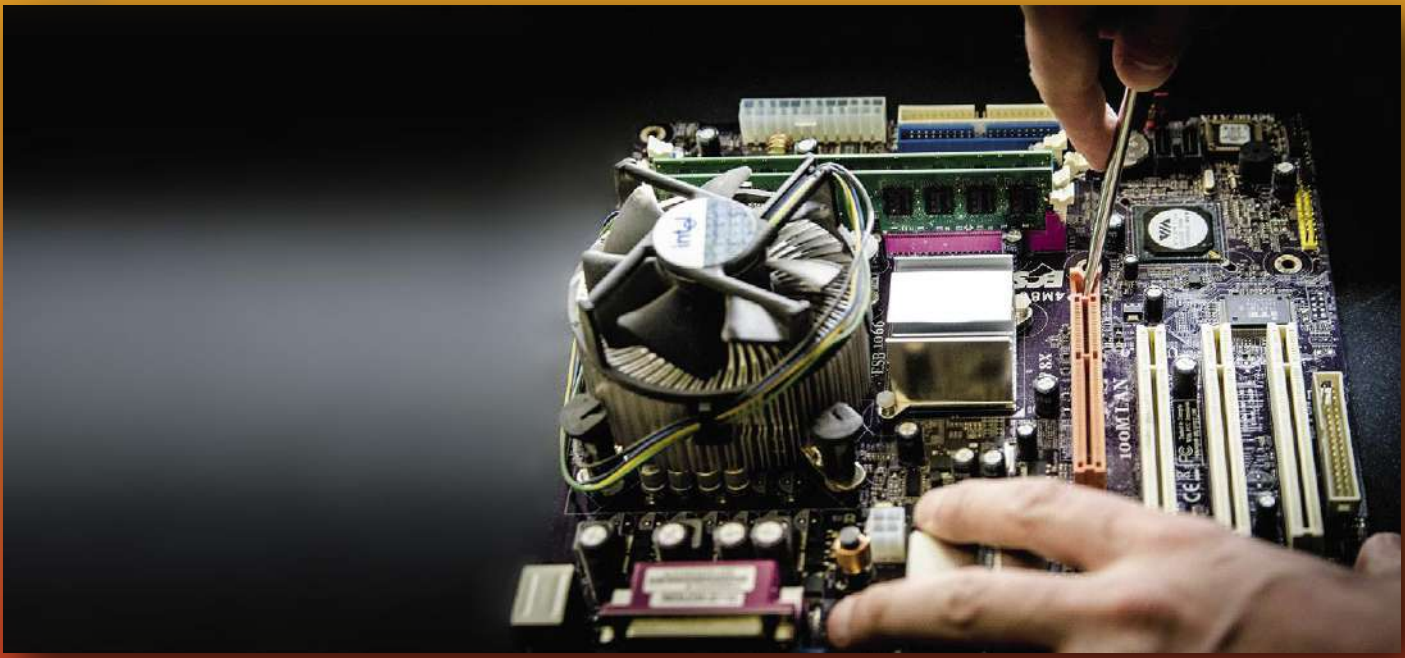
spite of warning messages on the shield) and tries to remove the battery with a screwdriver), the battery may be punctured and catch fire. In the eyes of the law, the manufacturer is responsible, not the @#$% user. This sort of thing has already happened in the U.S.A. This explains why most mains power supply units are ultrasonically welded, not screwed together… The cost price and as a consequence the selling price (which the buyer always wants as low as possible) have a direct impact on the design of the device. For example, it costs less to solder the connections of a display rather than use a connector, or to solder the wires of a headset rather than use spring connectors. This war of 'always cheaper' has the result of making the devices more difficult to take apart and thus to repair. And end users don't want to pay a bit more for a 'fixable' device, because in any case the product is under guarantee.

To sum up

This idea that devices are deliberately designed to fail just after the expiry of the guarantee is an old chestnut that we see regularly. All manufacturers have a standard test plan for each new product. This is the result of years of experience designed to result in a device which goes faulty as little as possible, while keeping the price attractive to the customer (there we are again). For this purpose, the manufacturer creates profiles of average daily use and, for his tests, multiplies the reference values by 1 000 (1 000 days, or roughly 3 years). For example, if he reckons the average user lets his phone fall from a maximum height of 10 cm sixteen times a day, the device must therefore withstand 16 000 such falls. And he does this for every imaginable occurrence. At the end of the day, every new device has to pass all these tests before being launched on the market. Of course, you could take 3 650 days (10 years) as the basis for your tests. But in this case, the materials and components would have to be stronger, so more expensive, so the sale price would be increased to a level that the customer is not ready to pay.

Conclusion

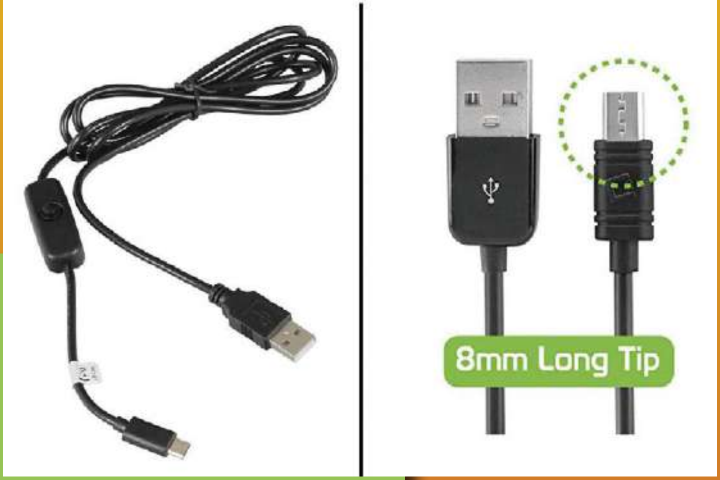
It is certain that the difficulty in repairing some devices is a direct result of the appetite of consumers for products with a neat design, and that this is not a deliberate ploy of the manufacturers. I have been in this industry for thirty years and I've been involved in the design of hundreds of phones, and never has it been a question of making a device that can't be repaired so that the customer has to buy another one in a shorter time.



Costly spare parts

Yes, they are often costly and the repair of a product after the guarantee period is expensive. Why? For the answer, we must look at the law, industrial constraints and the global market. The law insists on repair under guarantee for a period of one or two years (depending on the country). It's also obligatory to supply spare parts for three years after the last sale. The manufacturer must thus maintain a stock of spare parts for after-sales service often during production, which are often quickly used up. It's widely known that most parts in this industry have a very short life cycle due to the evolution of the technology, and it's often impossible to re-order them after the end of production. If the manufacturer cannot repair a product under guarantee, he must exchange it for a new product, and that's a complete loss. He thus has a strong incentive to keep his spare parts for all the repairs under guarantee, repairs outside the guarantee are of far less priority, and thus expensive. Finally, spares which are too cheap generate a grey market and trafficking of parts between countries and continents.

Indispensable lab tools



Here are two very practical USB cables. Firstly there is the USB cable with built in switch. No more pulling out the cable to switch off or restart you boards of devices, just press the button. Much less wear on the USB connectors and cables due to frequent connections. Then there is a USB cable with an extra-long plug (8 mm).This cable is indispensable in situations where it is not possible to get the micro-USB connector close enough to the case, or when the opening is not big enough for the plug body.

www.elektor.com/usb-a-to-micro-usb-b-cable-with-switch

www.elektor.com/8mm-micro-usb-connector-white-2m

180567-01

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

Tips and Tricks

From readers for readers

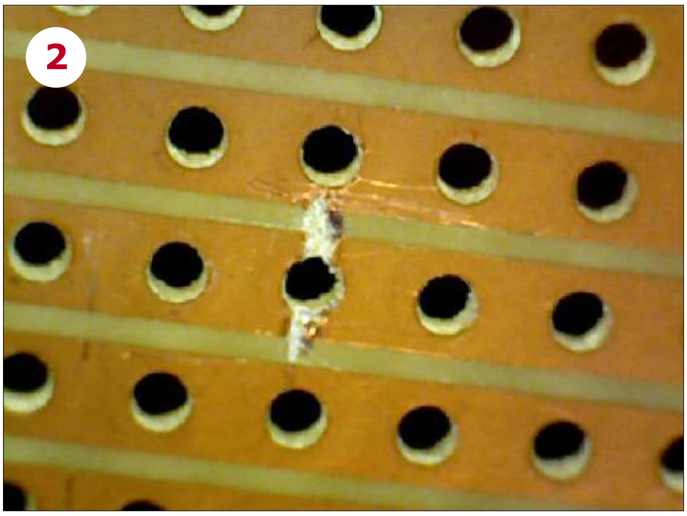
Another neat solution to a tricky problem

1

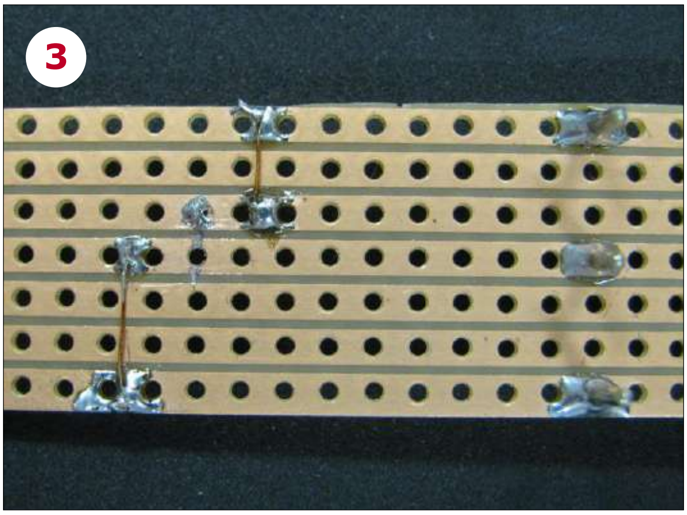
Low-cost SMD-Testadapter

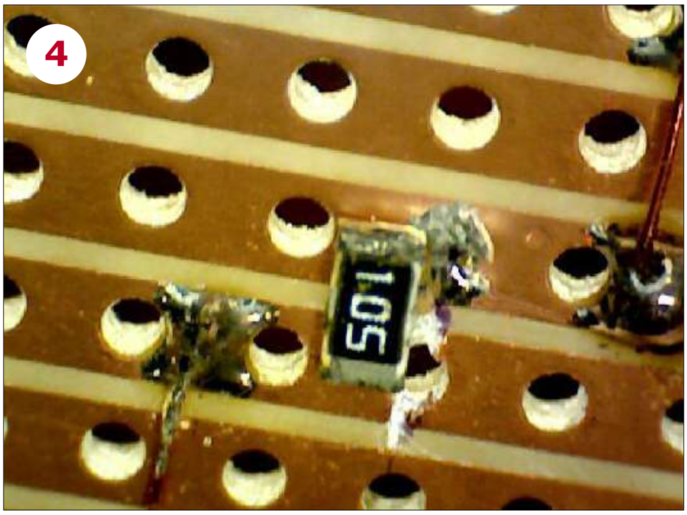
By **Dipl.-Inf. Karl-Ludwig Butte**

Have you ever tried to measure an SMD resistor or SMD transistor using an ohmmeter? It’s not as easy as it looks and your luck was probably like mine. Faster than a flea, the resistor sprang straight off the lab bench into infinity and was never seen again.


2

Take a hairgrip (**Figure 1**), a small piece of stripboard, three solder pins and some enamelled copper wire. In a few simple steps you can create a small but extremely useful SMD test adapter.
First slide the hairgrip onto the stripboard in line with the copper tracks. At the end of the hairgrip, cut the track at the nearest hole using a watchmaker’s screwdriver, a countersink tool, a Veroboard track cutter or something similar (**Figure 2**). Next, solder two short wire strap (bridges of enamelled copper wire) and three solder pins as in **Figure 3**. Fill the hole above the gap in the conductor with a blob of solder. **Figure 4** and **Figure 5** show how an SMD resistor and an SMD transistor are connected afterwards. But hold off for now; first of all, we need first to attach the hairgrip. Slide it onto the stripboard as shown in **Figure 6** and make sure that the end of the clip


3


4

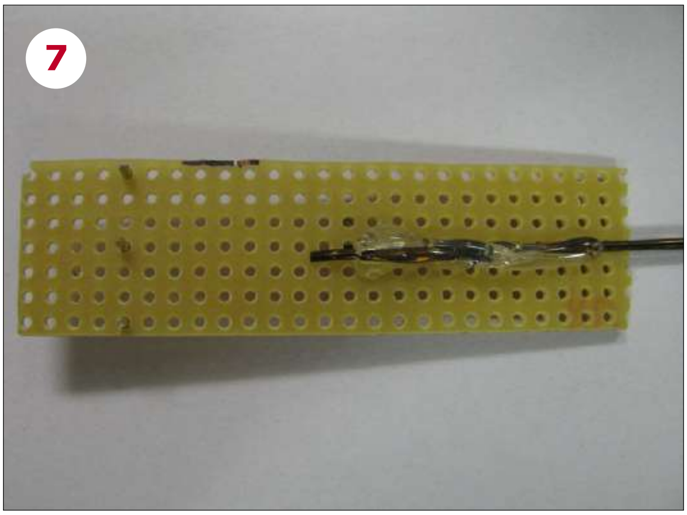
Have you come up with an inspired way of solving a really tricky problem? Perhaps you've discovered an ingenious but unconventional way of using a component or tool? Maybe you've invented a better or simpler way of tackling a task? You deserve a reward, write in — for every tip we publish you win £40 (or local equivalent)!

presses on the hole where the copper track was broken. Now turn the PCB over and fix the hairgrip with hot-melt glue (**Figure 7**). Since mechanical tension tends to make the clip bulge upwards, it needs to be pressed down with a suitable pair of pliers until the adhesive has solidified.

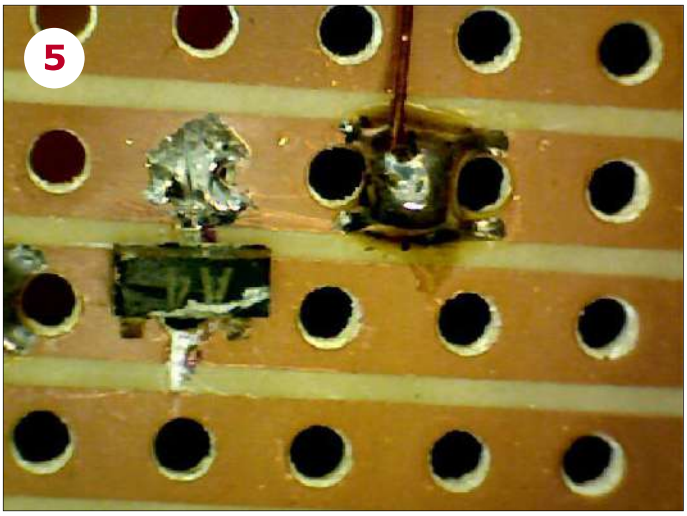
Now we're ready to try out the test adapter. Push an SMD resistor under the hairgrip and connect an ohmmeter to the corresponding solder pins (**Figure 8**). On the uppermost scale in the top-left corner we read a value of solid 1 kΩ, although this actually means 1 MΩ in the measurement range selected. The markings printed on the resistor must therefore be read as '105' and not, as you might assume, as '501' (which would mean 500 Ω). See how easy it is to measure SMDs now!

All that remains now is to test a transistor. Slide an SMD transistor under the hairgrip, as in Figure 5, and connect your transistor tester (**Figure 9**). The test instrument shows that (a) the transistor is OK, (b) it is a PNP type and (c) its base is connected to the black crocodile clip. ⏮

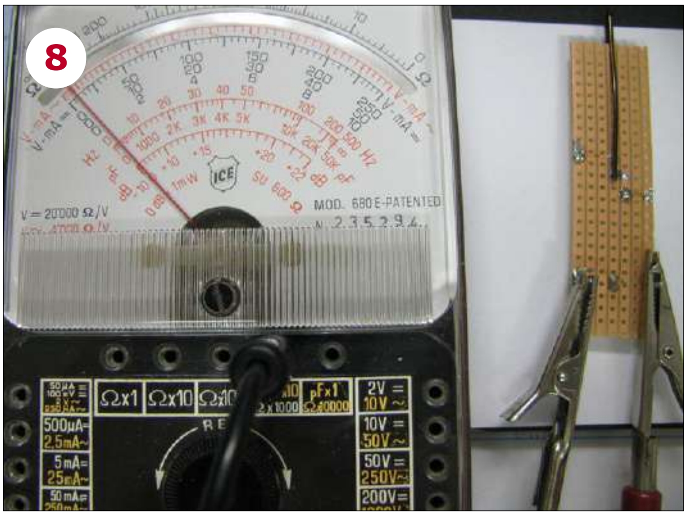
170569-02



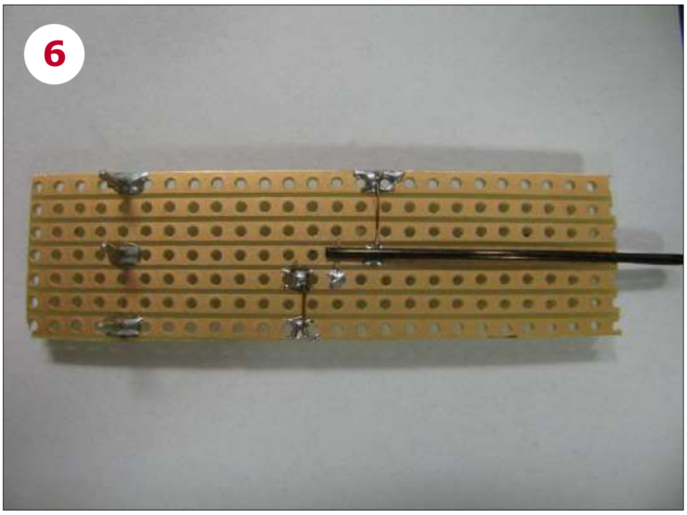
7



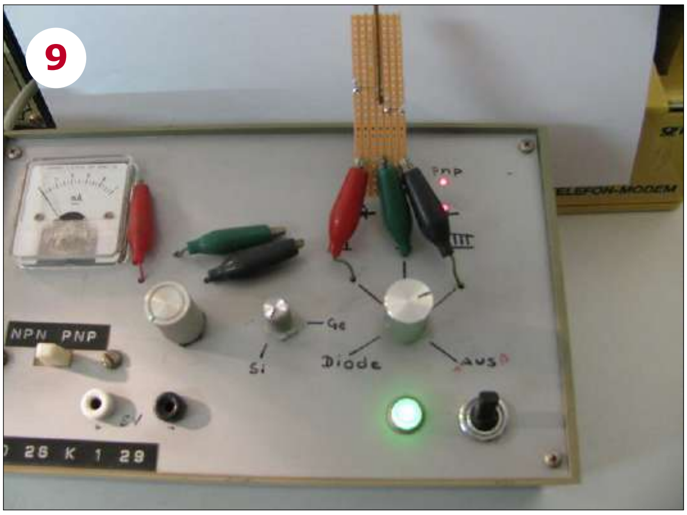
5



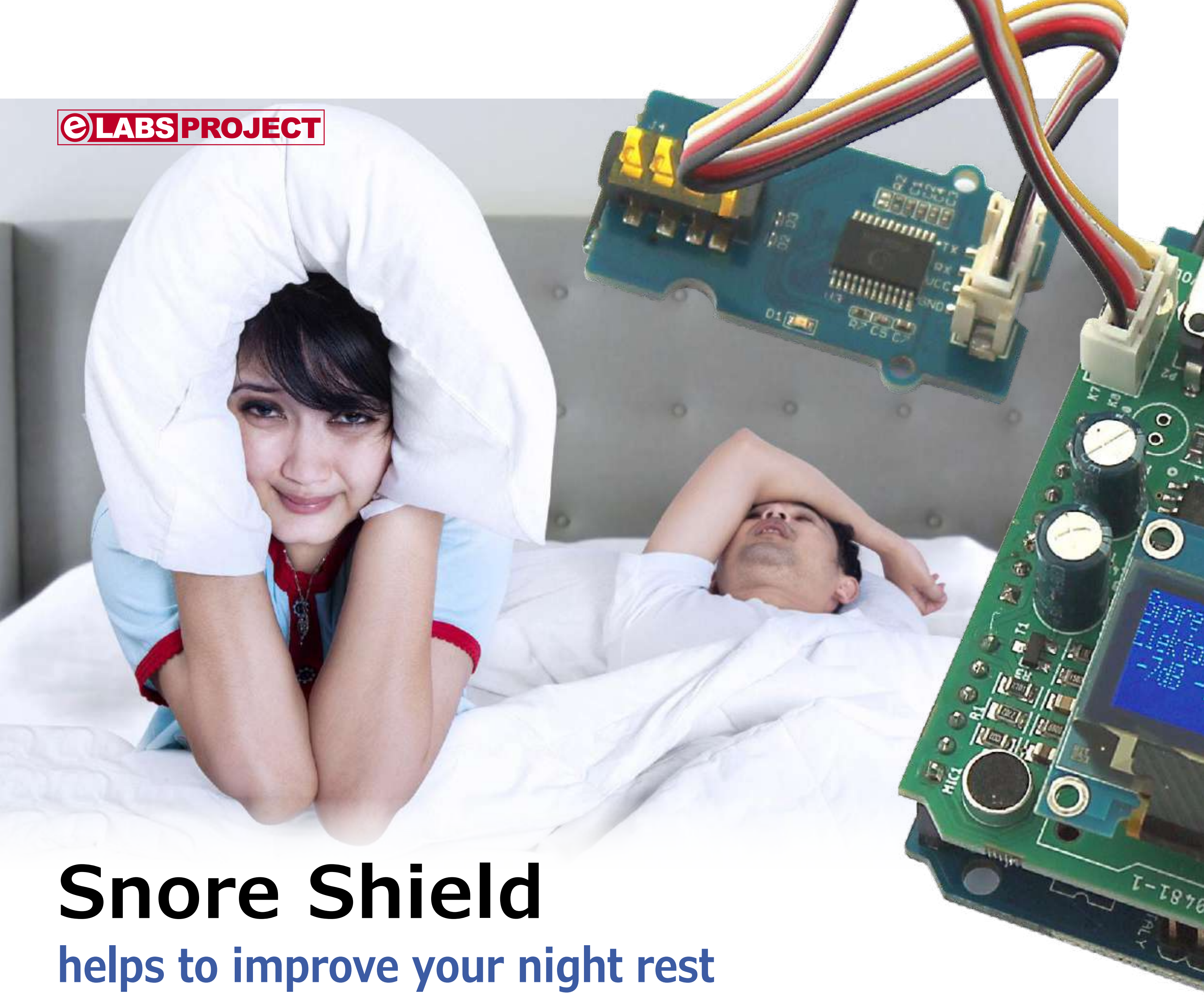
8



6



9



Snore Shield

helps to improve your night rest

By **Clemens Valens** (Elektor Labs)

Many people snore and many don't mind or seem to notice the noise. Sadly, another good part of humanity looses sleep as a result of snoring bed partners and/or roommates. This project is for them.

Snore stress

The problem when trying to fall asleep in the same room as a snorer is that both focussing and anticipating on the snoring sounds results in stress, which keeps you awake. Kicking, slapping, pushing or prodding the snorer doesn't always help and typically increases the amount of stress, making it even more difficult to fall asleep.

Enter the Snore Shield

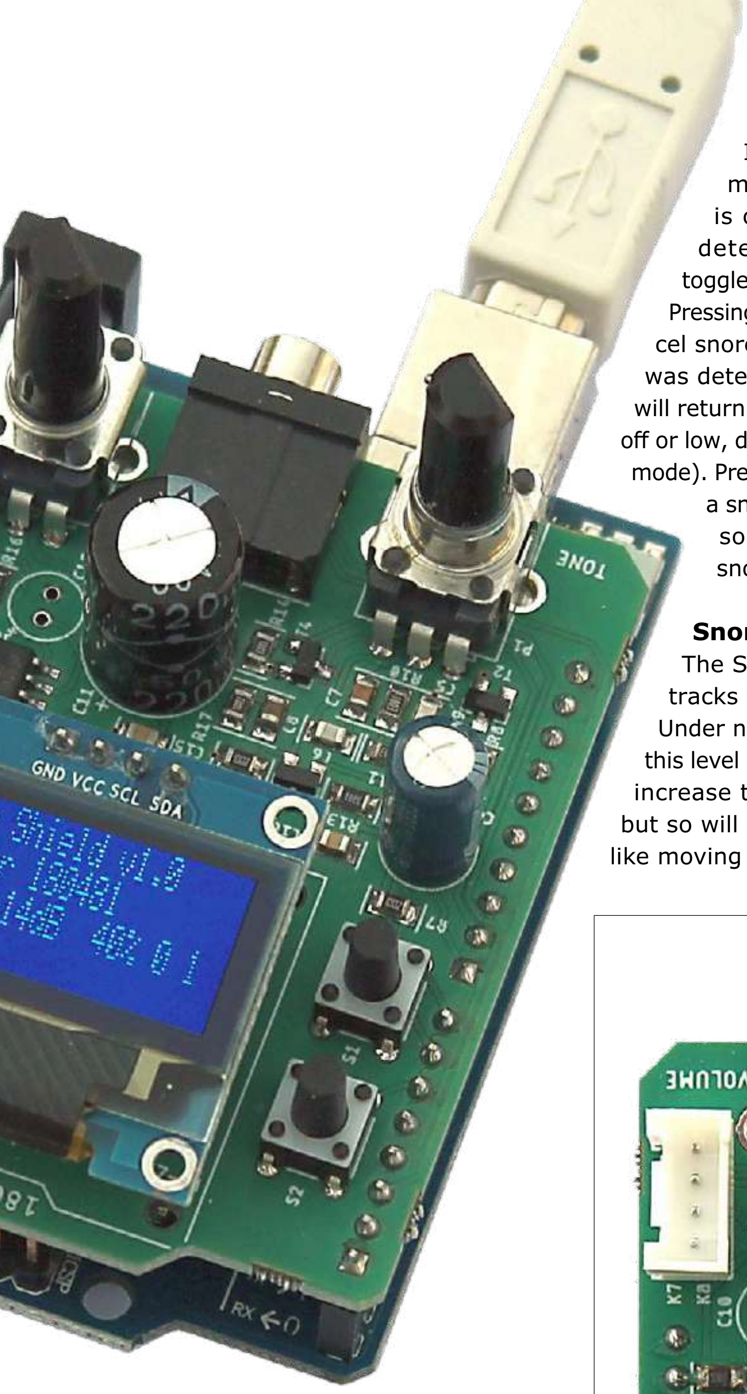
The device presented here produces an agreeable noise for the non-snorer to harken to and hopefully relax with. The device also "listens" for snores. When it detects snoring it increases the volume of the relaxing sound to cover the snoring. After five minutes — supposed to be enough for either the snoring to cease or for the non-snorer to fall asleep — the sound level decreases, completing a cycle, and snore detection starts over.

Relaxing sound

The sounds of the sea, a light surf, and wind through the trees are all considered relaxing sounds by many people. The Snore Shield (**Figure 1**) therefore produces white noise (with adjustable colouring), which aims to approach these relaxing sounds. Those who prefer other sounds like running water of relaxing music can connect an MP3 player module.

It started out as an Android app

The Snore Shield is based on the Android application 'SleepSation' (*sic*; no typos; available in the Google Play Store). Because this app runs on a mobile phone (or tablet) it is necessary to keep a phone next to your bed. Unfortunately, a lot of people don't want a mobile phone next to their pillow, which is why the developers of the app asked us to design a smartphone-free device.



The Snore Shield turns any bedroom into a high-tech Sleeping Lab (very unlike Elektor Labs).

The Snore Shield is Arduino shield-compatible and plugs onto an Arduino, preferably a Uno (to keep it small), but other types should work too.

Two modes

The device has two operating modes:

- continuous sound;
- snore-controlled sound.

In continuous-sound mode the relaxing sound is on continuously. When snoring is detected the sound level is increased during five minutes.

In snore-controlled sound mode the relaxing sound is on only when snoring is detected. Pushbutton SW2 toggles between the two modes. Pressing pushbutton SW1 will cancel snore detection when snoring was detected and the sound level will return to its default level (either off or low, depending on the operating mode). Pressing it when idle will force a snore detection, causing the sound level to increase to snore-covering level.

Snore detection

The Snore Shield continuously tracks the ambient sound level. Under normal sleeping conditions this level should be low. Snoring will increase the average sound level, but so will any other sound sources like moving in bed or passing traffic.

PROJECT INFORMATION

Health

Arduino

entry level

➡ intermediate level

expert level

2 hours approx.

SMD soldering tools,
Arduino IDE

€25 / £30 / $40 approx.

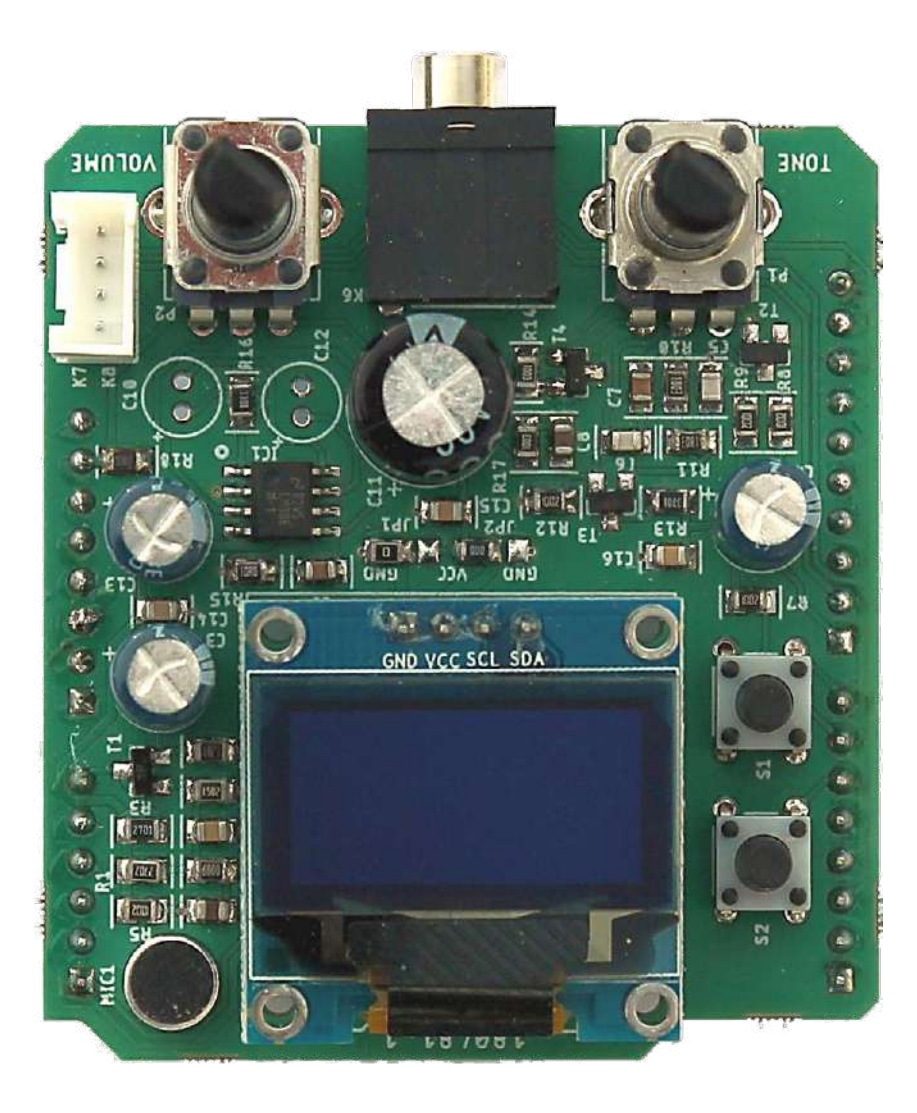


Figure 1. The Snore Shield is an Arduino-Uno-compatible add-on board.

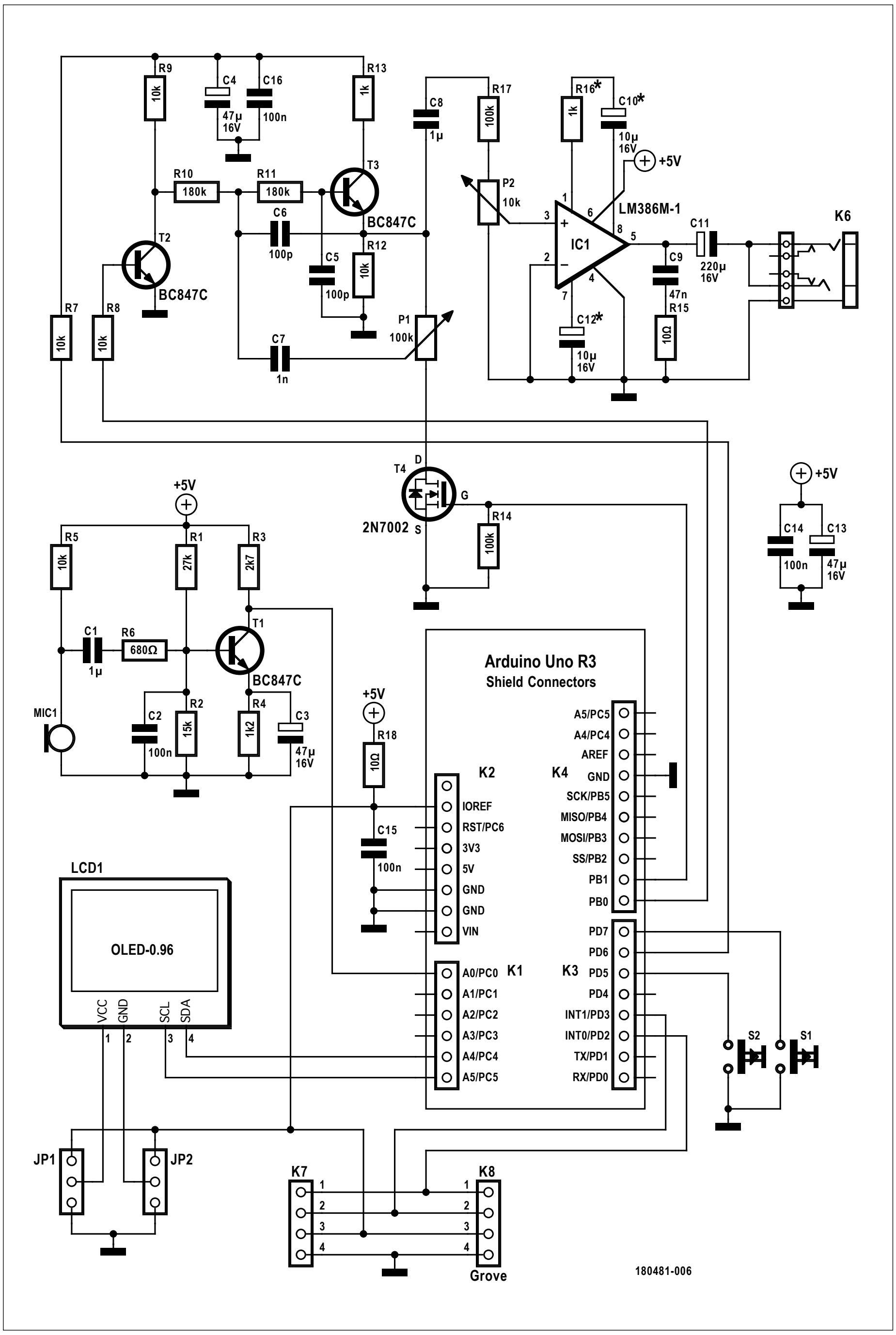


Figure 2. The schematic of the Snore Shield consists mainly of microphone amplifier T1, voltage-controlled noise processor T2 & T3, and power amplifier IC1.

Specifications

- Two snore detection modes
- Electret microphone
- Headphone output
- OLED I²C graphical display
- Two pushbuttons
- Volume potentiometer
- Noise colour user control
- Compatible with Grove MP3 player module
- Powered by 9-12◦VDC adapter or 5-V USB powerbank

Detecting snoring therefore is a bit more complicated than simply comparing the current sound level to a running average. The Snore Shield requires three snores in a row, where a snore is defined as a short (up to about three seconds) loud sound followed by a period of silence lasting five to ten seconds.

Of course, false detections cannot be excluded and sounds with similar characteristics may trigger the device too, like three coughs in a (slow) row.

Details about the inner workings

The schematic of the Snore Shield is depicted in **Figure 2**. Snore detection starts with a microphone, MIC1, connected to an amplifier with a passband covering about 100 Hz to 3 kHz. The amplifier output is connected to analogue input A0 of the Arduino's ATmega328 which takes care of digitising and digital processing. Inside the MCU the audio signal is sampled at a rate of 7 kHz. Although this sounds more than twice the input bandwidth, in reality the amplifier's slope is not very steep and frequency analysis of the audio signal will be difficult. It's not much of a problem here as we are just looking at the sound volume. To cover the snore sounds the system produces white noise. This is done in software using a linear feedback shift register (LFSR) approach using a 32-bit shift register and a sample rate of 20 kHz. The digital noise is output on Arduino pin D8. Transistor T2 functions as a voltage controlled amplifier to allow amplitude modulation of the noise signal. The control voltage is produced by a lowpass-filtered PWM signal available on Arduino pin D6. The circuit around transistor T3 is a filter to adjust the noise "colour". For simplicity's sake the filter's feedback is controlled with a classic potentiometer. The software can switch transistor T4 on or off to select between two feedback modes, resulting in two types of noise.
The output of the noise amplifier is connected to a classic LM386-based power amplifier with manual volume control. This amplifier can drive headphones or small speakers.

MP3 player for custom sounds

The MP3 player module used is a Grove module from Seeedstudio (**Figure 3**)

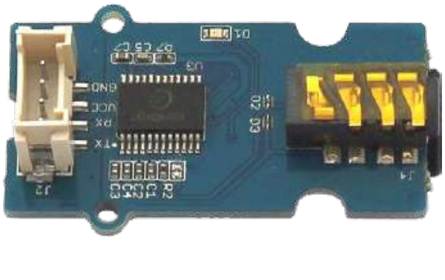


Figure 3. An MP3 player module may be connected to improve overall sound quality and to make the Snore Shield play custom sounds.

Use a high-quality loudspeaker

It is important that the relaxing sound is relaxing, so be sure to connect a proper sound system. A crappy loudspeaker will produce crappy sound, defeating the purpose of the system. Please carefully read the section about the power supply elsewhere in this article. A volume control and noise colour control are available to adjust the sound.

and it connects to K7. This module is controlled over a serial port emulated in software on Arduino pins D2 and D3. Other types of MP3 player modules using similar control schemes can be used too if you are ready to modify the software. Note that Grove connector K8 is a special type with a 2-mm pitch. K7 offers a 0.1” pitch alternative here.

The software serial port unfortunately has a negative impact on the digital noise signal as it switches off interrupts when sending, effectively introducing audible glitches in the interrupt-controlled noise signal. If the MP3 module is not used, you should comment-out a line in the program to suppress these glitches. A comment in the code indicates which line is concerned.

The display

The OLED display is one of those popular 0.96” 64 × 128-pixel graphic types. These displays exist in several versions, with either six or four pins, and with different pinouts; we opted for the 4-pin type. This type comes in two flavours: pin 1 = GND, pin 2 = VCC and the other way around. Solder jumpers JP1 and JP2 let you use either one of them.

A display is practical for user interaction, but at the moment it isn't very helpful. It was added because the SoundSation app has a number of timer and recording options that were not (yet?) ported to the Snore Shield.

On the first line (**Figure 4**) it shows the device's name (in case you forgot it) and firmware version; on the second is the

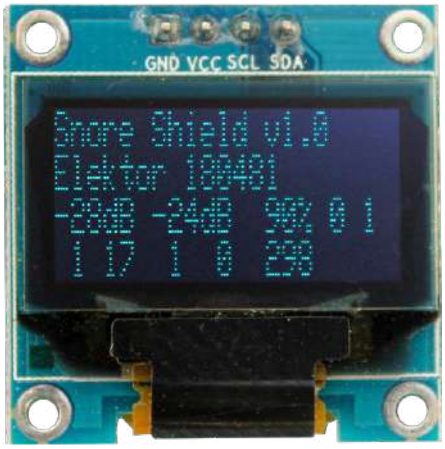


Figure 4: The Snore Shield’s OLED display currently specialises in useless debug information. The number “298” on the fourth line is the number of seconds to wait before the system returns to snore-detection mode (in this case snore-shield mode was triggered manually).

Elektor project number.

The third line shows sound statistics (detection threshold and ambient sound level in dB) and the sound level (as a percentage). At the end is the mode: '0' for continuous sound or '1' for snore-controlled sound. At the end is the noise type '0' or '1'.

Finally, the fourth line shows snore detection status with a bunch of numbers counting up, down and around. When the last number counts up to three the system will enter snore-covering sound mode. Snore detection stops and these numbers freeze for five minutes. The volume level on line three will slowly increase to a higher percentage.

Software

The software needed to get the Snore Shield up and running can be downloaded for free from [1], and you are invited, nay encouraged, to experiment with it. Maybe you can improve the snore detection algorithm, or maybe you want to customise the sound levels? It is a single-file Arduino sketch and a bunch of libraries, and as such it can be programmed into the board without special tools. All you need is a recent Arduino IDE (we used 1.8.0).

Please share your findings with the world by posting them at [1]. ◀

180481-01

A word about power supplies

A typical stand-alone Arduino application would be powered from a phone charger. Unfortunately these chargers are very noisy electrically, badly affecting the quality of the relaxing noise produced by the Snore Shield. It is therefore highly recommended to power the device from an old-school 9 VDC (12 VDC max) wall wart (power adapter), although a USB powerbank should be fine too. The upshot: do not use a phone charger or a PC USB port.

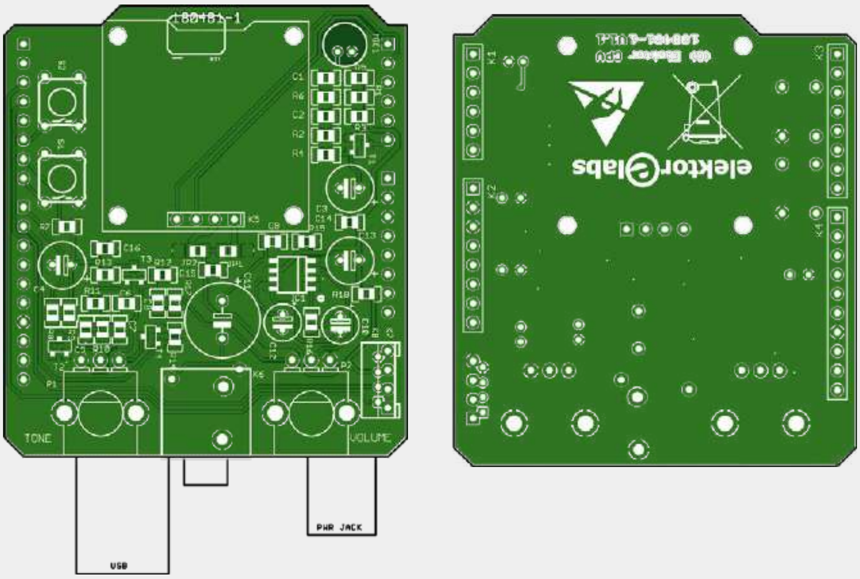
COMPONENT LIST

Resistors
Default: SMD 0805
JP1,JP2 = 0Ω
R15,R18 = 10Ω
R6 = 680Ω
R13,R16 = 1kΩ
R4 = 1.2kΩ
R3 = 2.7kΩ
R5,R7,R8,R9,R12 = 10kΩ
R2 = 15kΩ
R1 = 27kΩ
R14,R17 = 100kΩ
R10,R11 = 180kΩ
P2 = 10kΩ, potentiometer, vertical
P1 = 100kΩ, potentiometer, vertical

Capacitors
Default: SMD 0805
C5,C6 = 100pF
C7 = 1nF
C9 = 47nF
C2,C14,C15,C16 = 100nF
C1,C8 = 1µF
C10,C12 = 10µF 16V, 2mm pitch
C3,C4,C13 = 47µF 16V, 2.5mm pitch
C11 = 220µF 16V, 5mm pitch

Semiconductors
T4 = 2N7002
T1, T2, T3 = BC847C
IC1 = LM386M-1/NOPB

Miscellaneous
MIC1 = electret condenser microphone
K6 = jack connector, 3.5mm
S1,S2 = tactile switch, pushbutton, 6x6mm
K1 = 6-pin pinheader, 0.1" pitch
K2,K3 = 8-pin pinheader, 0.1" pitch
K4 = 10-pin pinheader, 0.1" pitch
K5 = 4-way pinheader socket, 0.1" pitch
K7 = 4-pin pinheader, 0.1" pitch
K8 = 4-pin pinheader, 2mm pitch
MOD1 = OLED display, 0.96", I²C controlled
PCB #180481-1, from Elektor Store



@ WWW.ELEKTOR.COM

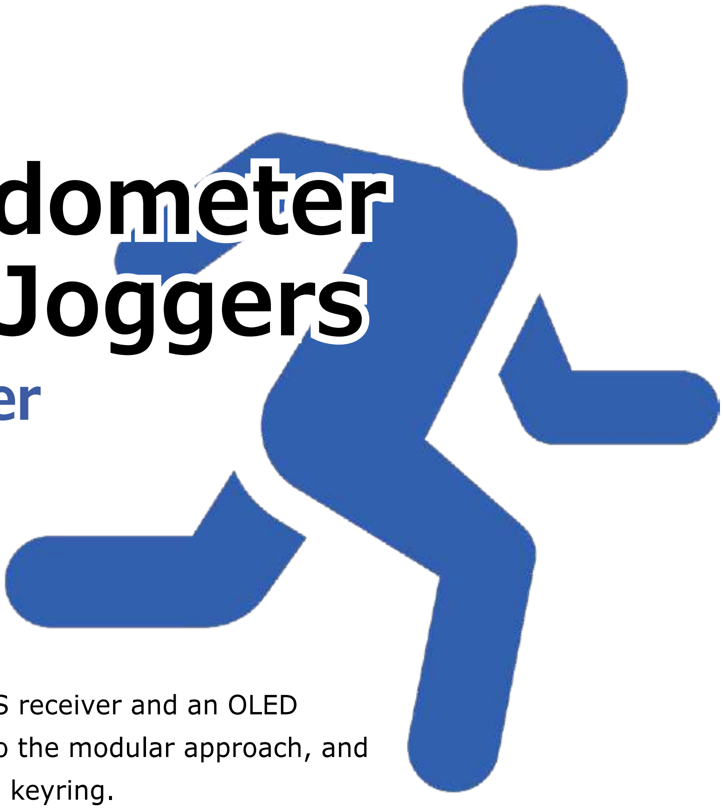
→ 180481-1 Snore Shield, bare PCB.
www.elektor.com/snore-shield-180481-1

→ Grove MP3 v2 – Programmable compact audio player
www.elektor.com/grove-mp3-v2

→ Grove universal 4-pin connector (2-mm pitch)
www.elektor.com/10-x-4-pin-2-mm-pitch

Web Link

[1] Project at Elektor Labs: www.elektormagazine.com/labs/snore-shield



Personal Speedometer for Runners & Joggers

With trip meter and logger

By **Bera Somnath** (India)

For keeping track of the route travelled and calculating the average speed while jogging, the author has coupled an Arduino Nano to a simple GPS receiver and an OLED display. The circuit fits easily in your hand, thanks to the modular approach, and with a little extra effort it could even be made into a keyring.

I often go jogging in the morning, and for some time I have attempted to measure my speed in a reasonably reliable manner. I first tried to use a high-frequency Doppler sensor, but that was not a success for measuring my own speed. What did work, however, was measuring the relative speed at the instant that a large object passed by. After that I experimented with a BME280 sensor. This sensor measures atmospheric pressure, temperature and humidity. My thought was that a change in speed would cause the air pressure in the horizontal direction to change and a sensor mounted perpendicular to this direction should be able to measure that. But unfortunately that was not the case. Or in any case, the difference in air pressure was too small to be measured with the BME280. For my next (and final) attempt I deployed a GPS receiver [1]. This, fortunately, was successful at measuring the speed with sufficient accuracy, when joined with an Arduino Nano and provided at least three satellites are within view. An additional advantage of a GPS receiver is that, besides the speed, I can also measure the current time and the route travelled during my jogging exercise. And from that also the total distance covered and the average speed during a *run*. Adding a nice OLED display completes the whole thing.

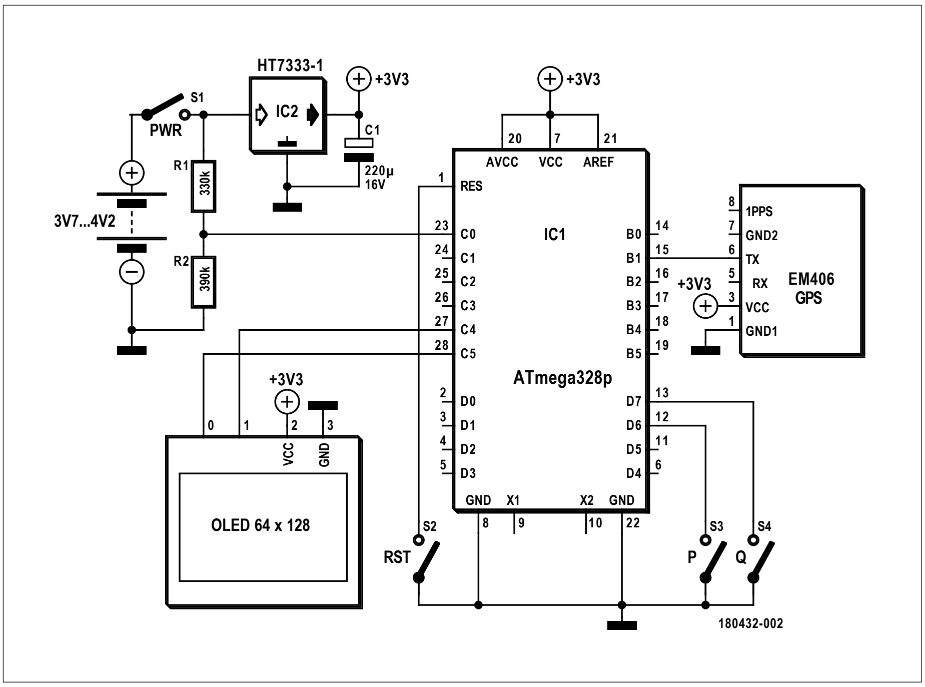


Figure 1. We find only seven components in the schematic, because modules have been used, all tied together by the ATmega328P.

Measurement philosophy — V1

Suppose the situation that, when during your jog you turn around and then return the same way you came. The distance covered, which is a function of latitude and longitude of the path traversed, would be zero in this case, but in actuality you have covered the route twice. To solve this problem I have connected a push button (‘P’, S3) to the digital pin 6 of the microcontroller (see **Figure 1**). You need to press this the moment you start walking and then every time there is a change of direction.

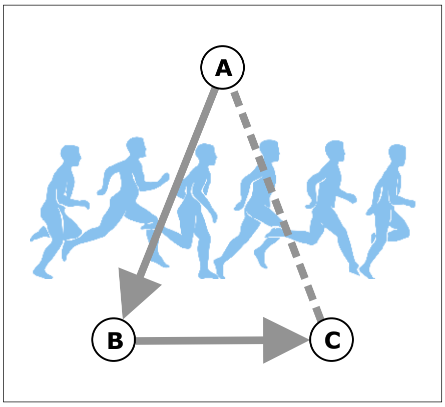


Figure 2. A jogger who runs from A to B to C and back to A, has not moved anywhere in the end, but nevertheless covered some distance.

Why not a smartphone?

My daughter asked me why I just didn't use a smartphone for this — a legitimate question from the new generation. I have my reasons for that...

- I don't like to take my phone with me while jogging. That is a distraction.
- Searching for 'jogging' you will find more than 30 apps ranging from 7 to 181 MB. I can't see the wood for the trees.
- In some apps you have to enter some data about yourself: step size, body weight, height, ... I run daily just 6 to 7 km at 9-10 km/h and relax afterwards. I have no need for all those other features.
- Nearly all apps require an active Internet connection.
- This is a small circuit that easily fits in your hand. It could even be built as a keyring when you replace the Nano with a plain SMD ATmega328.
- A smartphone will cost at least €150; this circuit costs much less!

Suppose you jog from A to B to C and then finally return to A (see **Figure 2**). The net distance is zero, but you certainly have covered some distance, namely AB+BC+CA. To keep track of the total distance covered, you start with pushing button P at point A (this will start the calculation), then again at point B, then point C, and one last time after returning to point A. The total distance covered, that is AB+BC+CA, together with the total time in seconds that you needed to cover that distance, is stored in the EEPROM memory of the ATmega328. This microcontroller has 1024 bytes available for this.

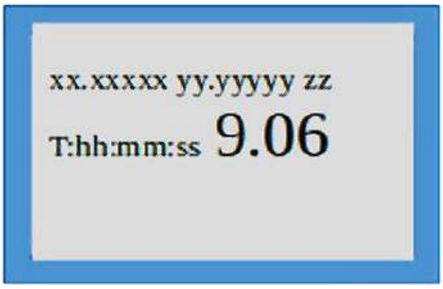
Display

When P or Q are not pressed, the display shows from top left to bottom right the latitude, the longitude and the number of satellites that the GPS module is receiving signals from, the current time and the instantaneous speed (in km/h), see **Figure 3**. When P is pressed for the first time, the display shows the latitude, the longitude, the number of 'visible' satellites, the current time, the current speed and "I/L:AB/AB 00 00"; where 'I' means 'Instant distance' (current distance of the present track segment), 'L' means the total distance (cumulative), the first 'AB' stands for the number of meters covered in the present track segment, the second 'AB' stands for the number of meters of the total distance travelled up to now. The first '00' is the number of track segments and the second '00' is the number of seconds that have elapsed since the start.

When P is pressed for the second time, the display shows latitude, longitude, number of visible satellites, current time, current speed and for example I/L: 'CA/(AB+BC+CA)' 03 1950, where CA and (AB+BC+CA) are presented in meters, '03' shows the number of track segments and '1950' the number of seconds that have elapsed since the start of the measurement. With this information we can, for the example shown in **Figure 4**, read that a distance of 12 m has been covered in the present track segment, that a total distance of 25 m has been covered in 5 different track segments and that the measurement started 62 s ago. When you press the second button ('Q', S4), the total time since the start of the measurement is shown (in s), the distance covered (in m), the current time and the average speed (in km/h) over the entire distance (see **Figure 5**). You can push this button as often as you like to display the information on the screen. The information also remains in memory after the power supply (via S1) is switched off. However, when button P is pressed, this data is replaced by new information.

Adaptations — V.2

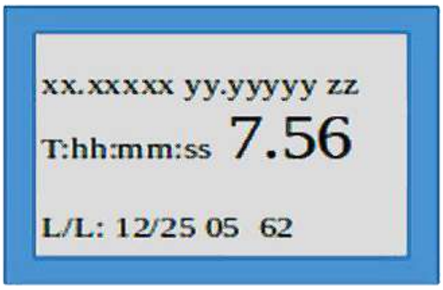
A few of my friends also used the circuit in their daily training routine and because they asked me to change a few things, I was happy to do that for them. It is this modified version that is shown here in the schematic of **Figure 1**.



xx.xxxxx yy.yyyyy zz

T:hh:mm:ss 9.06

Figure 3. Default state.

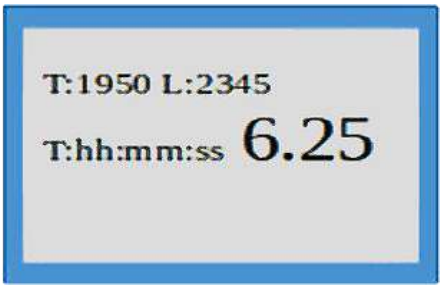


xx.xxxxx yy.yyyyy zz

T:hh:mm:ss 7.56

L/L: 12/25 05 62

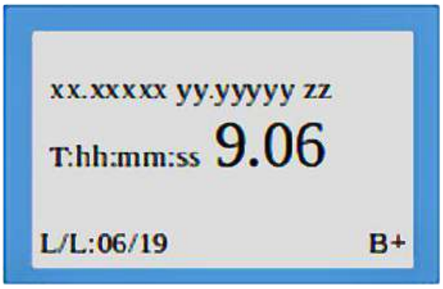
Figure 4. Button P pressed.



T:1950 L:2345

T:hh:mm:ss 6.25

Figure 5. Button Q pressed.



xx.xxxxx yy.yyyyy zz

T:hh:mm:ss 9.06

L/L:06/19 B+

Figure 6. Button P pressed, adapted version.

When switch P is pressed once while the default screen is shown on the display (**Figure 4**), the meter is reset to zero and the meter starts to count as soon as it moves. On the last line of the display, on the right, I added a simple battery indicator. As long as the voltage of the LiPo battery is more than 3.3 V, a ‘B+’ appears, as shown in **Figure 6**. Once this falls below 3.3 V, it changes to ‘B–’, which indicates that it is time for the LiPo to be charged.

I added a voltage divider (R1, R2) for monitoring the battery voltage. I also included an HT7333-1 voltage regulator, which is eminently suitable here because of its low voltage drop and low quiescent current consumption.

Construction and testing

The actual construction is one of the simplest parts of this project. Find a small piece of double-sided circuit board and fit the Arduino, the OLED display and the GPS receiver on the front side. The antenna for the GPS module has to be placed at the front (so it is not screened by the PCB) to ensure the best possible reception of the GPS satellite signals. The LiPo cell can be mounted in the back of the PCB. **Figure 7** shows my built-up circuit. Don’t forget to program the Arduino with the software [2].

I always get the greatest pleasure from testing my own creations. Provide the circuit with the correct voltage using a LiPo or Li-ion battery and close S1 (at a higher power supply voltage, such as 9 or 12 V, S2 can probably be left open). Take your circuit with you when you go training and optionally your partner or friend to impress him/her with your new gadget! Once the GPS receiver sees enough GPS satellites, the PPS LED (Pulse Per Signal LED) will begin to flash and the OLED display will show the information as described above. ⏮

180432-02

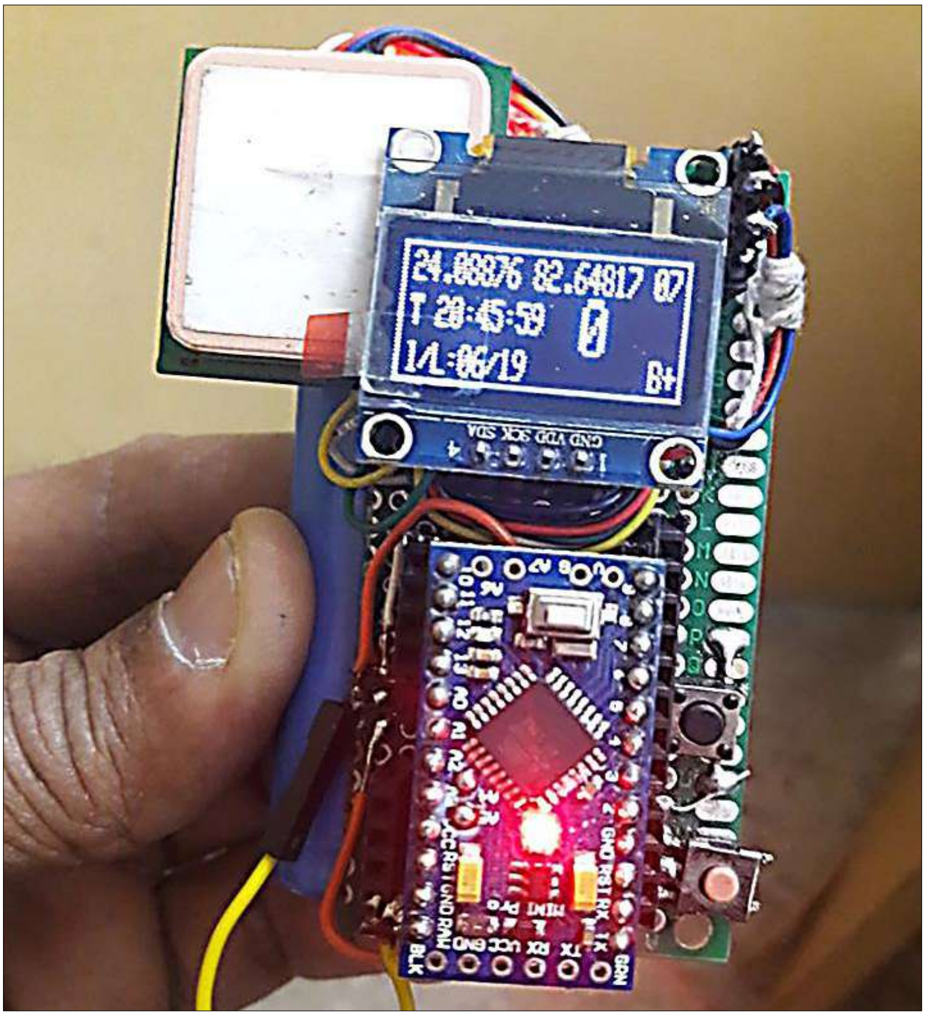


Figure 7. The circuit is easily built on a small piece of circuit board and is small enough to be held in your hand.

Web Links

[1] GPS module: www.elektor.com/gy-neo6mv2

[2] Project downloads: www.elektormagazine.com/180432-02

In memoriam

The author dedicates this article to his mother, Sefali Bera, whose death deeply affected him and the community where she lived in Howrah, India. She was a great source of inspiration for his articles.

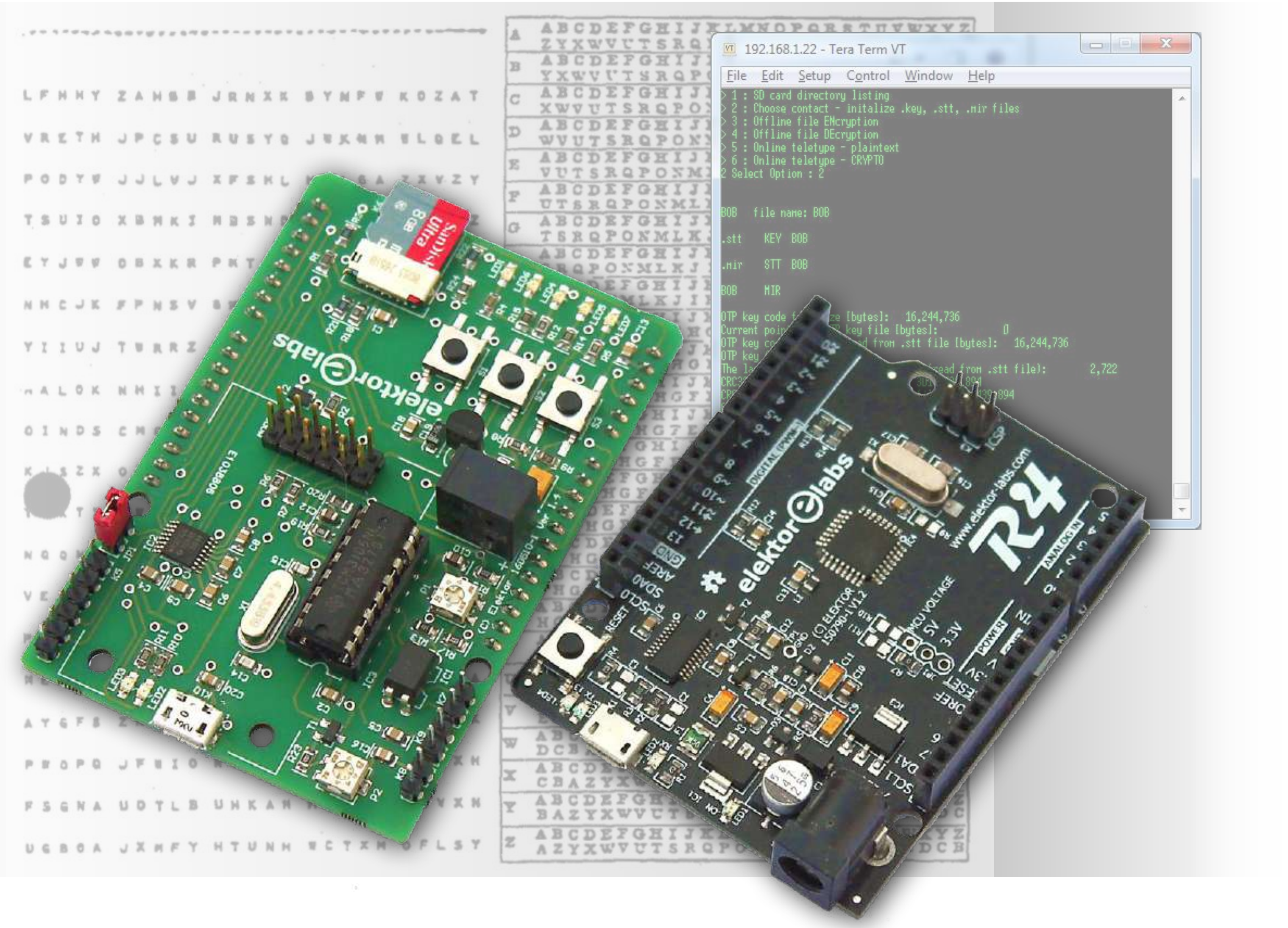
eLABS PROJECT

One-time Pad Crypto Shield

secure communication through openness

By **Luka Matić**

With a wide variety of cryptography hardware and software commercially available, it may not be obvious why you would want to build your own system. Yet the reason is simple: you just can't trust most of these devices and programs because in 99% of the cases you have no clue of what is going on inside!


192.168.1.22 - Tera Term VT
File Edit Setup Control Window Help

```
> 1 : SD card directory listing
> 2 : Choose contact - initalize .key, .stt, .mir files
> 3 : Offline file ENcryption
> 4 : Offline file DEcryption
> 5 : Online teletype - plaintext
> 6 : Online teletype - CRYPTO
2 Select Option : 2


BOB   file name: BOB

.stt    KEY  BOB

.mir    STT  BOB

BOB     MIR

OTP key code f     ze [bytes]:   16,244,736
Current poin      P key file [bytes]:           0
OTP key co         d from .stt file [bytes]:   16,244,736
OTP key
The la                  read from .stt file):        2,722
CRC3               894
CR                 139,894
```

Let's start this article with a little light history of cryptography (you may want to watch video [1] first). The well-known German *Enigma* cipher machine from the previous century (**Figure 1**) was a relatively simple electromechanical device consisting of a keyboard, a battery, some lamps and a constellation of rotary switches. Calculating the total number of possible combinations (and thus estimate its 'crypto' strength) was easy, as was monitoring its operation and checking for (hardware-only) errors. It had no software, and consequently it was certified free from software bugs. It was resilient enough to ward off attempts at crypto-analysis and other methods of attack available back in its time. *Enigma* had no built-in communications hardware; the keyboard input and lamp output were completely controlled by a human operator (the Cipher Officer) while a second person, the Radio Telegraph Operator, was in charge of the radio transmissions. Furthermore!

- It was impossible to plant a Trojan in the *Enigma* and create a plaintext leak to some IP address.
- It was immune to any sort of memory buffer overflow, and its keystrokes could not be recorded on a key logger (hardware or software).
- Operating at very low frequencies, it had almost no residual TEMPEST radiation [2], which was even less of a problem when operated inside a near-perfect Faraday cage like an average U-boat.

Today, most encryption is done by software running on a general-purpose PC, with some fairly expensive devices like cryptophones taking care of the rest. Even the best encryption algorithm is useless when running on an insecure device like a PC. Most cryptography hardware comes as a black box, and is thus difficult to analyse for a non-informed user. Even without hardware or software protection, the systems are often too complex for thorough analysis anyway. It is easy to understand that the old *Enigma* beats them all hands down in almost every security aspect! Do you see where I am getting at now? A system is actually more secure if it is easy to analyse. If you feel that this is all too paranoid, please read [3], and pay attention to the chapter dealing with “analysis of attack trees”.

One-time pad or OTP encryption

According to Wikibooks [4], one-time pad (OTP) encryption is the only potentially unbreakable encryption method. Potentially, because it only works when the people using it scrupulously respect its rules.

In OTP every message is encrypted with a unique random key (**Figure 2**). The key must satisfy the following conditions:

- a key must be generated by a non-deterministic, non-repeatable process — no algorithm of any kind may be used for this;
- a key is used only once;
- the key must not fall in the hands of the enemy.

OTP encryption itself is easy enough: generate a suitable key and XOR it with the plaintext message. Decryption is equally simple: just XOR the encrypted message with the same key and recover the plaintext. The encrypted message therefore has the same length as the plaintext message. Due to the random nature of the key it is impossible to determine the original message from the encrypted message alone, as all possible plaintexts of the same lengths are equally likely. OTP is the only cryptosystem for which such a proof is known.

A practical system

As mentioned above, OTP is based on a simple, easy-to-program mathematical algorithm suitable for execution by a microcontroller unit (MCU) with a known (Harvard) architecture. When the source code of the algorithm is published, it can be analysed and debugged in detail by

PROJECT INFORMATION

Cryptography OTP TRNG
Secure Communication
Arduino

entry level
➡ intermediate level
expert level

2 hours approx.

SMD soldering tools,
AVR programmer,
Atmel Studio

€25 /£30 / $40 approx.

Figure 1. A close look of the famous *Enigma* electromechanical rotor cipher machine.

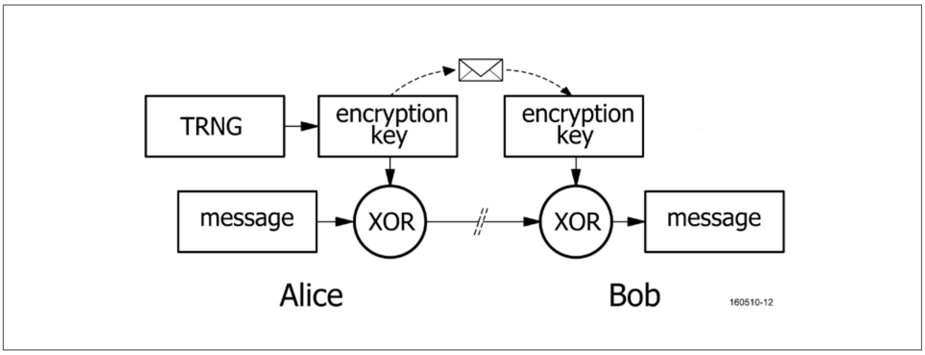


Figure 2. How Alice sends a message to Bob using one-time pad (OTP) encryption.

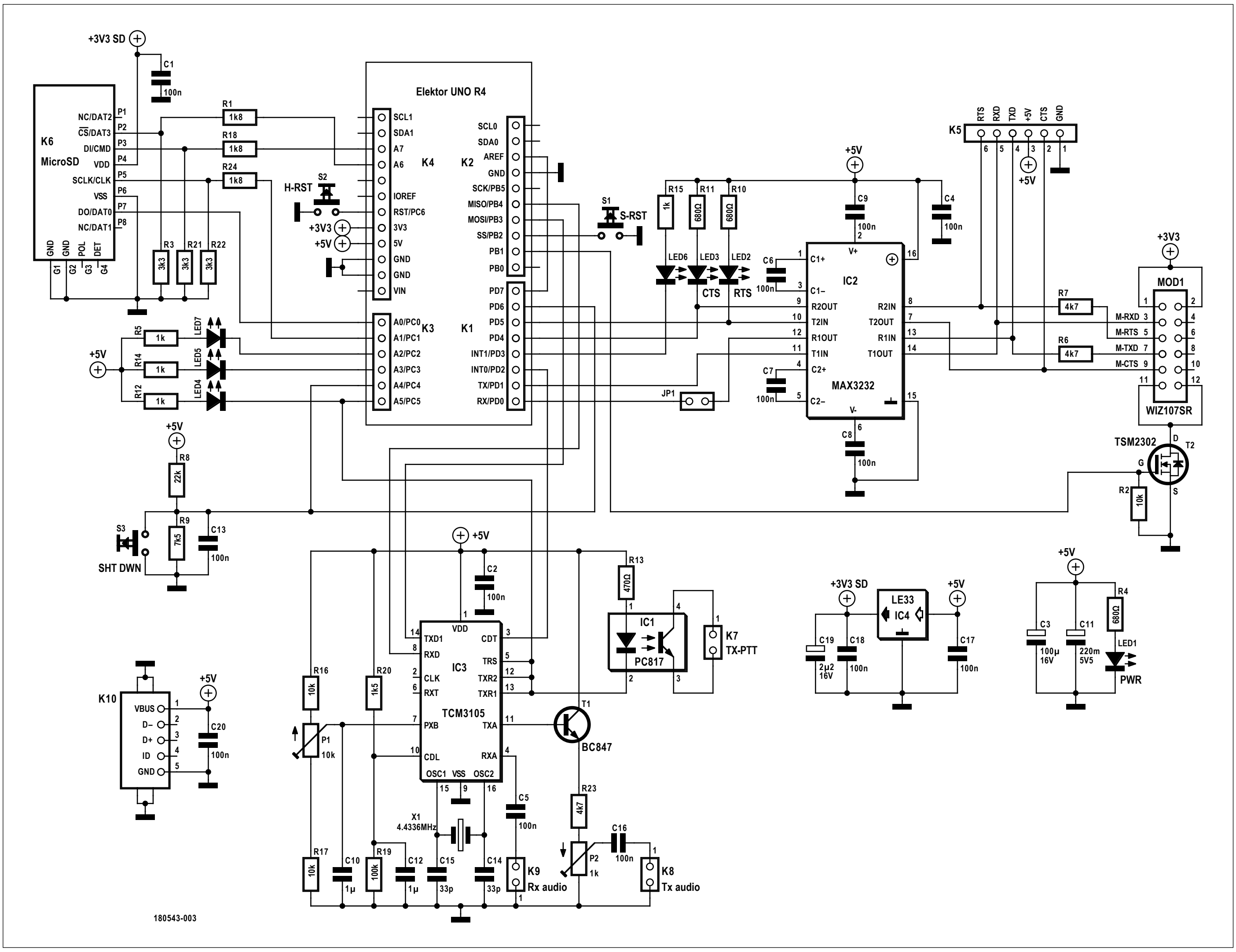


Figure 3. The schematic of the OTP Crypto Shield can be cut into three main parts: the RS-232 and Ethernet port around IC2, K5 and MOD1; the audio interface with FSK MODEM IC3 at its heart, and the microSD card slot for storing key data and messages.

anyone to check what is really going on inside the MCU. The algorithm uses only two data RAM buffers that need to be kept from overflowing.

Generating strong encryption keys requires a good true random number generator (TRNG, see [5]) also built out of open-source software and open hardware so that it is possible to analyse every aspect of its operation. OTP keys are stored on an SD card, allowing gigabytes of random sequences to be used (no need for bulky OTP paper notebooks as used in the old days).

A terminal (for instance a cheap 7-inch laptop with its WiFi module and camera removed, never-ever to be connected to the Internet) with a wired Ethernet or serial port to communicate with the MCU, and an SD card slot to manage files on the SD card is used. Encryption is effectively done inside the microcontroller, which — like the *Enigma* — is physically separated from other computer operations and wireless transmissions. This makes the system difficult to attack by a virus or a Trojan malware. Its TEMPEST radiation is still much stronger than Enigma's though.

Built without any special hardware and with ubiquitous components, such a system is easy to assemble and can be analysed in as much detail as you want.

Two operating modes

The OTP system presented here has two operating modes: Online and Offline. In Online mode encrypted messages are sent and received over an audio connection. In Offline mode messages and files are encrypted and stored on some medium like an SD card and then transported to the destination by means of email, land mail or a courier service.

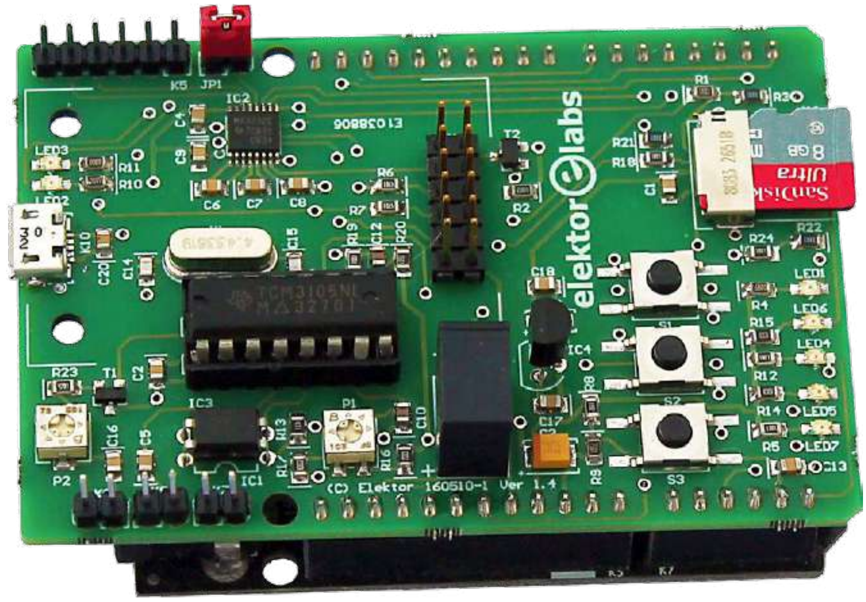
Description of the hardware

Our OTP cryptography system consists of an Arduino-compatible board with an extension card — the OTP Crypto Shield — plugged on top of it. The Elektor Uno R4 was chosen as a base board because of the ATmega328PB microcontroller with its two SPI ports and two UARTs. The schematic of the shield is shown in **Figure 3**. The main components of the OTP Crypto Shield are:

1. A microSD card slot used for OTP key files (online- and offline mode), and files to encrypt/decrypt (offline mode).
2. An RS-232 port (MAX3232) used to connect the shield to a terminal. Serial data can be sent over a real RS-232 connection (K5) or over Ethernet with the help of a WIZ107SR Ethernet-to-RS-232 adapter (e.g. when the terminal does not have a serial port like a cheap 7-inch laptop). R6 and R7

can be used to disconnect the Ethernet module if the latter is soldered to the shield.

3. An FSK modem (TCM3105) to send and receive the ciphertexts over an analogue audio channel (online mode).



Soft shutdown

To avoid using the same OTP sequence twice it is important to keep track of the position of the current OTP key pointer and other important variables. In case of an accidental power loss, this data must be stored quickly on the SD card. In case a power failure occurs, supercapacitor C11 will continue powering the circuit for a few seconds, thus allowing a graceful ('soft') shutdown. A voltage drop on PD6 is detected by the MCU's internal voltage comparator, allowing the soft shutdown to be performed in an interrupt service routine. A soft shutdown can also be requested manually by pressing pushbutton S3.

Transistor T2 disconnects the WIZ107SR module (if present) in the event of a power failure because it draws more than 200 mA which would cause the voltage to drop too much due to the internal resistance of supercap C11, and writing to the SD card becomes impossible.

S2 is a hard Reset button which must be used with care to prevent loss of crucial data (like the OTP key pointer value). S1, on the other hand, provides a safe soft reset without data loss, and is usually used when the MCU's UARTs get flooded in online mode or to stop a key pointer search if it takes too long.

Power supply subtleties

K10 is used to connect a 5 V supply voltage to the device. The micro USB connector on the Elektor Uno R4 is fine for flashing the firmware with the aid of the bootloader (note that JP1 on the shield must be open when doing this), but it must not be used for powering the OTP Crypto Shield during normal operation because it can affect the MCU reset. To avoid this from happening the shield is actually powering the Uno R4.

Why so many LEDs?

The OTP Crypto Shield has a number of LEDs. LED1 is a power indicator, while LED2 and LED3 indicate activity on the RTS and CTS signals on UART0 of the MCU. LED4 indicates data transmission on the modem's audio port (coming from UART1), LED5 indicates data reception; LED7 indicates a reception error.

LED6 indicates a soft reset (when button S1 got pressed and the request is being processed by the MCU), or flashes during a long search for a key pointer when the MCU attempts to decrypt a message (or a file in Offline mode) after having lost the key pointer position. When a low-voltage condition is detected or after pressing S3, LED5 and LED6 blink alternately to indicate a successful update of the status ('.STT', see below) file; when they blink simultaneously, the status file was not updated (because its contents did not change of the write attempt was unsuccessful).

Audio channel

An FSK modem (IC3) provides an audio channel for transmitting and receiving encrypted messages using, for instance, a phone or handheld PMR. IC3 operates at 1200 baud using FSK frequencies of 1300 Hz and 2100 Hz. According to Carson's Rule, the bandwidth of the analogue signal will be between 700 Hz and 2700 Hz which is fine even for a low-quality audio channel. P1 sets the threshold to distinguish between the two frequencies ('0' or '1'). To adjust it, send a message from another shield with a weak signal in Online plaintext mode (option 5, see below). P2 is set in the middle position for connection to the microphone or line input of most devices. Set it to a low value when adjusting of P1 on the other shield.

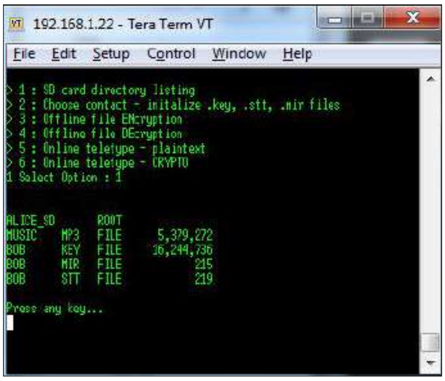
Optocoupler IC1 is used to 'push' the push-to-talk (PTT) button if a simplex device like a VHF/UHF PMR is used to transmit/receive the encrypted audio FSK signal.

Besides a portable PMR, any other analogue channel (direct wire connection, analogue telephone) or digital channel with lossless compression (like G.711, A-law or μ-law) can be used for Online mode. That's including any smartphone connected to the Internet, running some lossless codec VoIP software. Connect it to the OTP Crypto Shield through its EAR-MIC analogue audio port. Lossy codecs like GSM or those used by Skype or Viber can't be used. They are great for human speech signals, but not for an FSK modem signal because its pitch changes much faster than a human voice.

So how does it work?

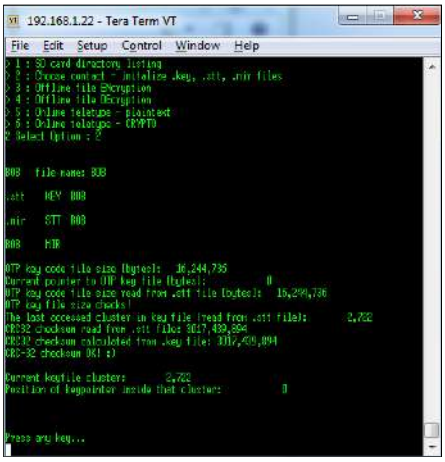
Suppose two persons want to communicate using OTP Crypto Shields. To make this possible, both parties (i.e. peers) must have the same OTP key file (KEY, generated by the true random number generator TRNG). As said before, the same subsequence of random bytes taken from the key file and used for data encryption may never be used more than once to ensure security. This means that the peers must have some kind of pointer into the key file, and these pointers must be synchronized. This also means that when synchronisation is lost (for any reason), it must be possible to restore the key pointer. To avoid reusing the same subsequence (which would compromise OTP system security immediately), both parties must have a way to keep track of both key pointers.

This is why besides the KEY file each peer must also have a status file (STT) containing his/her key pointer (and other control data), and a 'mirror' file (MIR) file. This one, sent to or received from the remote peer, contains the encrypted version of the STT file, and is used for authentication and status checking of the remote peer only in offline mode.



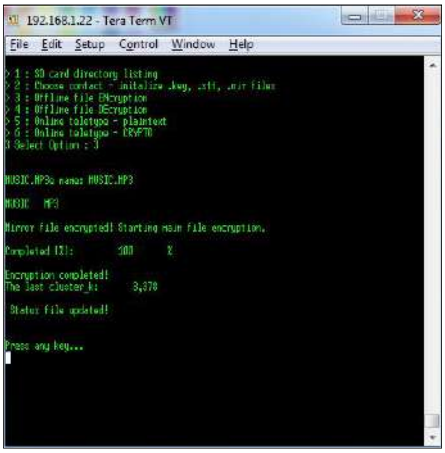
```
> 1 : SD card directory listing
> 2 : Choose contact - initalize .key, .stt, .mir files
> 3 : Offline file ENcryption
> 4 : Offline file DEcryption
> 5 : Online teletype - plaintext
> 6 : Online teletype - CRYPTO
1 Select Option : 1


ALICE_SD       ROOT
MUSIC     MP3  FILE     5,379,272
BOB       KEY  FILE    16,244,736
BOB       MIR  FILE           215
BOB       STT  FILE           219

Press any key...
```

Figure 4. Select option 1 to list the contents of the root directory.



```
> 1 : SD card directory listing
> 2 : Choose contact - initalize .key, .stt, .mir files
> 3 : Offline file ENcryption
> 4 : Offline file DEcryption
> 5 : Online teletype - plaintext
> 6 : Online teletype - CRYPTO
2 Select Option : 2


BOB   file name: BOB

.stt    KEY  BOB

.mir    STT  BOB

BOB     MIR

OTP key code file size [bytes]:   16,244,736
Current pointer to OTP key file [bytes]:           0
OTP key code file size read from .stt file [bytes]:   16,244,736
OTP key file size checks!
The last accessed cluster in key file [read from .stt file]:        2,722
CRC32 checksum read from .stt file: 3017,439,894
CRC32 checksum calculated from .key file: 3017,439,894
CRC-32 checksum OK! :)

Current keyfile cluster:        2,722
Position of keypointer inside that cluster:           0



Press any key...
```

Figure 5. To choose a contact from the address book, enter 2.



```
> 1 : SD card directory listing
> 2 : Choose contact - initalize .key, .stt, .mir files
> 3 : Offline file ENcryption
> 4 : Offline file DEcryption
> 5 : Online teletype - plaintext
> 6 : Online teletype - CRYPTO
3 Select Option : 3


MUSIC.MP3e name: MUSIC.MP3

MUSIC   MP3

Mirror file encrypted! Starting main file encryption.

Completed [%]:        100       %

Encryption completed!
The last cluster_k:      3,378

 Status file updated!


Press any key...
```

Figure 6. Encryption of the file completed.

An example

Let's use the traditional Alice, Bob, Eve and Mallory characters to explain how things work. Alice and Bob — the "good guys" — try to exchange some crucial information. Evil Eve will try to passively eavesdrop and crack their encryption, while malicious Mallory will try to jam the communication or alter the contents of transmitted messages or files.

Alice's SD card will contain the files BOB.KEY, BOB.STT and BOB.MIR, Bob's SD card will contain ALICE.KEY, ALICE.STT and ALICE.MIR. Both KEY files are identical — generated by the TRNG — and contain 16 MB to 4 GB of random data. The STT status files are small files (not more than 512 bytes) containing human-readable text that can be edited with a text editor like Notepad:

```
ALICE.STT:
POINTER :0000000000 [bytes]
TOTAL SIZE :0016244736 [bytes]
CLUSTER :0000007287 [clusters]
CRC32 :2989374417

Alice in Wonderland
Some additional info...

BOB.STT:

POINTER :0000000000 [bytes]
TOTAL SIZE :0016244736 [bytes]
CLUSTER :0000007287 [clusters]
CRC32 :2989374417

Sponge Bob Square Pants
Some additional info...
```

POINTER and TOTAL SIZE indicate when a new OTP key file is needed as the value of POINTER slowly advances to the value of TOTAL SIZE. CLUSTER holds the number of a cluster on the SD card where POINTER currently points at inside the key file. CRC32 is a checksum calculated from the first 16 bytes of the current CLUSTER and four bytes of the POINTER value (POINTER is a 32-bit long integer stored in the ATmega328PB SRAM in little endian format). This way alteration of these values is detectable. If you manually change the value of POINTER, then the MCU will search the SD for the correct CLUSTER and calculate the CRC32 and then update the .STT file accordingly. Note that the CLUSTER values on both SD cards do not have to be identical for equal POINTER values, because files will probably be placed in different clusters on different SD cards (and they can also be fragmented).

It is important that the STT file begins with the string 'POINTER ' (note the trailing space). The encrypted MIR file will be considered decrypted successfully (validating the key pointer in the process) if this 8-letter string (including the trailing space) appears at an attempt to decrypt the first eight characters of the MIR file. The four values in the STT file are 10-digit numbers, starting immediately after a space + colon ':' — this is how the MCU reads them from an STT file.

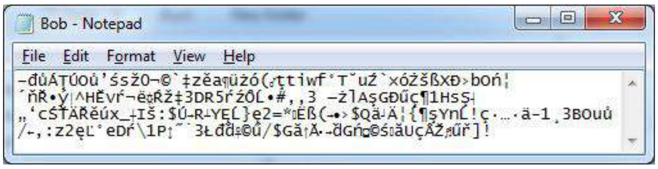
Offline mode

If Alice wants to send an encrypted file to Bob in Offline mode, the procedure is as follows:

1. Alice puts her SD card in her computer and stores the file to encrypt on the SD card. In this example it will be an MP3 file "MUSIC.MP3".
2. Alice copies her BOB.STT file to BOB.MIR file (deleting its old contents, they are not important now) on her SD card. BOB.MIR is now plaintext, and identical to BOB.STT.
3. She inserts the SD card into the OTP Crypto Shield and powers it up.
4. She lists the contents of the root directory (option 1, **Figure 4**).
5. She chooses the peer to send the encrypted file to. It is possible to have many different peer files on an SD card (different contacts in your address book - NAME.KEY, NAME.STT, NAME.MIR). Just enter the peer's name, the file extensions will be added automatically. (**Figure 5**)
6. She chooses option 3 — "Offline file Encryption" — and enters the complete filename including extension. The MCU will first encrypt the mirror file BOB.MIR and then proceed to encrypt the main file MUSIC.MP3. When encryption is completed, the MCU will update the status in the BOB.STT file (**Figure 6**).
7. Alice now removes her SD card from the OTP Crypto Shield and puts it in her computer. Playing MUSIC.MP3 won't work anymore because it is encrypted. Opening BOB.MIR will look something like **Figure 7** (it is now also encrypted). BOB.STT is still plaintext, but now looks like **Figure 8**. The POINTER has advanced, and so has the CLUSTER number; the CRC32 value therefore has changed too. The

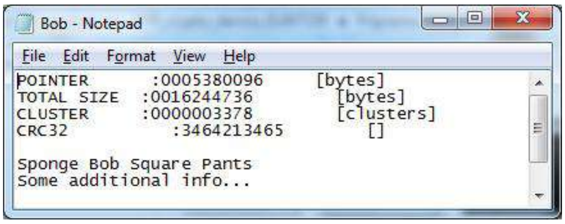
MUSIC.MP3 file was circa 5 MB which is reflected in the POINTER value.

8. Alice now sends an email with the encrypted MUSIC.MP3 and encrypted BOB.MIR files attached using any computer or smartphone or any other means of insecure communication (land/snail mail is also an option). Alice could rename BOB.MIR to ALICE.MIR but Bob can do it also when he receives it. It's better to rename both files for additional security.



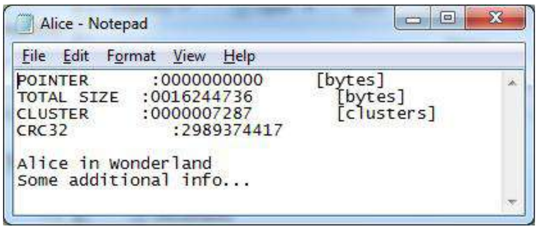
```
-đůÁŢÚOů'śsž0¬©`‡zěa¶üżó(⸗ţtiwf˚T˘uź`xóżšßXĐ›boń¦
´ňŘ•ý¦^HĚvŕ¬ë¤Řž‡3DR5ŕźôĹ•#,,3 –žlAşGĐűç¶1HsS┤
„'cŠTÄŘěúx_┼Iš:$Ú┴R┴YEĹ}ę2=*▯ĖB(→•›$Qä┘Ä¦{¶şYnĹ!ç·…·ä-1¸3BOuů
/←,:z2ęĽ˚eDŕ\1P↕˝˙3ŁđđǂOů/$Gă↑Ă·→ďGń▂©ś▯ăUÇĂŽ╗űř]!
```

Figure 7. The file BOB.MIR is now also encrypted.



```
POINTER      :0005380096      [bytes]
TOTAL SIZE  :0016244736        [bytes]
CLUSTER     :0000003378        [clusters]
CRC32           :3464213465        []

Sponge Bob Square Pants
Some additional info...
```

Figure 8. The updated status file BOB.STT invariably remains in plaintext.



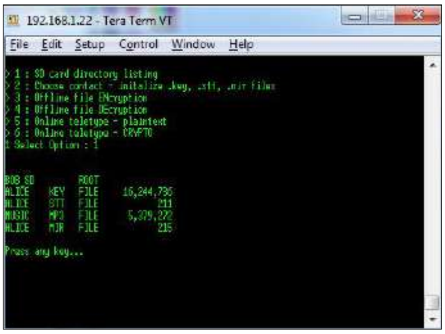
```
POINTER      :0000000000      [bytes]
TOTAL SIZE  :0016244736        [bytes]
CLUSTER     :0000007287        [clusters]
CRC32           :2989374417

Alice in wonderland
Some additional info...
```

Figure 9. The file ALICE.STT on Bob's SD card looks like this.

Receiving offline

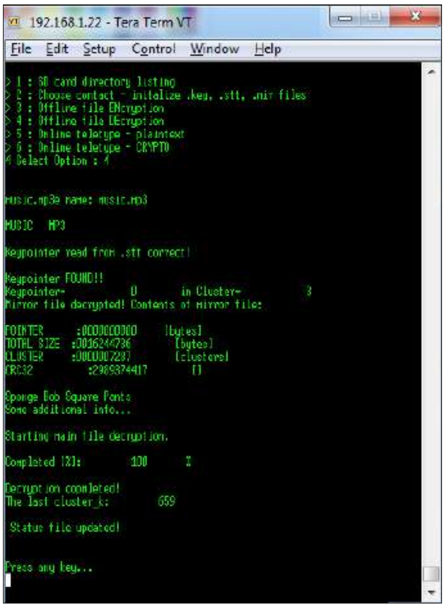
Bob's SD card contains files ALICE.KEY and ALICE.STT (**Figure 9**). Bob's POINTER is still on zero, and will reach 0005380096 after the decryption is completed. When Bob receives the files from Alice, the procedure to decrypt them goes like this:

1. Bob copies the two encrypted files received from Alice — MUSIC.MP3 and ALICE.MIR — to his SD card.
2. Bob puts his SD card into his OTP Crypto Shield. The SD card's root directory now looks like **Figure 10**.
3. After choosing the files with the peer's name (option 2) and some initialization the system is ready to start the decryption (option 4, **Figure 11**). The OTP Crypto Shield tries to decrypt ALICE.MIR with the key pointer value read from Bob's ALICE.STT file. The key file is scanned until the correct initial value of key pointer is found. Once again, decryption is considered successful if the first eight characters (remember the trainling space) of the MIR file read as 'POINTER '. After successful decryption, the contents of the MIR file are displayed. Now Bob can check several things:
   - The recorded position of Alice's key pointer (0000000000, before she started encryption) should be equal to the value of Bob's key pointer before he started decryption. If Alice's key pointer has a lower value than Bob's key pointer (the MCU will search and find the correct position anyway), then there is a possibility that security is compromised because Alice reused an OTP key sequence. Eve now has a chance to crack at least a part of their previous messages.
   - Gibberish characters appearing after decryption of the MIR file may indicate that Mallory has tampered with the message, for instance she altered the key pointer value inside the MIR file. One way to prevent such an attack is to hide a few short and meaningful authentication messages between the four 10-digit values before encryption of the MIR file (but without changing the formatting of the four values, and with-


192.168.1.22 - Tera Term VT
File Edit Setup Control Window Help

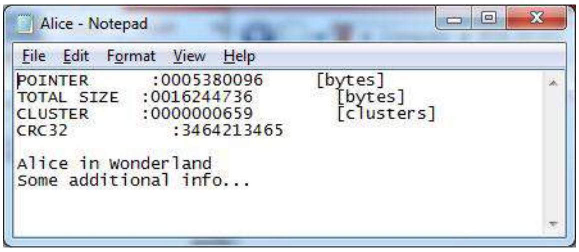
```
> 1 : SD card directory listing
> 2 : Choose contact - initalize .key, .stt, .mir files
> 3 : Offline file ENcryption
> 4 : Offline file DEcryption
> 5 : Online teletype - plaintext
> 6 : Online teletype - CRYPTO
1 Select Option : 1


BOB SD          ROOT
ALICE     KEY   FILE     16,244,736
ALICE     STT   FILE            211
MUSIC     MP3   FILE      5,379,272
ALICE     MIR   FILE            215

Press any key...
```

Figure 10. The root directory of Bob's SD card.


192.168.1.22 - Tera Term VT
File Edit Setup Control Window Help

```
> 1 : SD card directory listing
> 2 : Choose contact - initalize .key, .stt, .mir files
> 3 : Offline file ENcryption
> 4 : Offline file DEcryption
> 5 : Online teletype - plaintext
> 6 : Online teletype - CRYPTO
4 Select Option : 4


music.mp3e name: music.mp3

MUSIC   MP3

Keypointer read from .stt correct!

Keypointer FOUND!!
Keypointer-            0        in Cluster-           3
Mirror file decrypted! Contents of mirror file:

POINTER      :0000000000     [bytes]
TOTAL SIZE  :0016244736        [bytes]
CLUSTER     :0000007287        [clusters]
CRC32           :2989374417        []

Sponge Bob Square Pants
Some additional info...

Starting main file decryption.

Completed [%]:          100        %

Decryption completed!
The last cluster_k:         659

 Status file updated!


Press any key...
```

Figure 11. Decryption of the file completed. The file has to be played to find out if everything went fab / 10-4 / okey-dokey.



```
POINTER      :0005380096       [bytes]
TOTAL SIZE  :0016244736          [bytes]
CLUSTER      :0000000659          [clusters]
CRC32            :3464213465

Alice in Wonderland
Some additional info...
```

Figure 12. The status file on Bob's SD card should show an advance of the key pointer.

  out using colons ':' in these messages). This changes the position of the characters inside the MIR file, making it very likely for Mallory to change the wrong byte.
- All values in the decrypted MIR file can be manually checked by Bob for consistency in order to detect any tampering attempts from Mallory.

4. After decryption of the main file MUSIC.MP3, Bob can remove the SD card from his OTP Crypto Shield and put it into his computer. He can now listen to MUSIC.MP3, but he can also check the contents of his ALICE.STT status file (**Figure 12**). Both key pointers are synchronized, and the system is ready to securely encrypt the next communication session.

Online mode

In Online mode the OTP Crypto Shield constantly checks the value of the key pointer and alerts the user when its value decreased, indicating a reuse of an OTP sequence.

After selection of the contact (the same way as in offline mode) and initialisation of online encryption mode, the devices at both ends transmit the string "Rd" (ready). The one that receives it is clear to transmit the first message.

Each message sent by Alice contains:

- the key pointer (4 bytes);
- the message (payload, plaintext, *N* bytes);
- a CRC-16 checksum (2 bytes) calculated over key pointer and payload.

This 4+*N*+2 byte sequence is encrypted with OTP and a second CRC-16 checksum (again 2 bytes) calculated over the encrypted sequence is added to it.

Online reception

When Bob receives the message decryption goes like this:

First the checksum of the encrypted message is calculated to detect any errors that may have occurred during transmission. If the checksums don't match, the string "Er" will be sent back to Alice to

COMPONENT LIST

Resistors

Default: 0805, 125mW, 5%

R1,R18,R24 = 1.8kΩ

R2,R16,R17 = 10kΩ

R3,R21,R22 = 3.3kΩ

R4,R10,R11 = 680Ω

R5,R12,R14,R15 = 1kΩ

R6,R7,R23 = 4.7kΩ

R8 = 22kΩ

R9 = 7.5kΩ, 1%

R13 = 470Ω

R19 = 100kΩ

R20 = 1.5kΩ

P1 = 10kΩ trimpot

P2 = 1kΩ trimpot

Capacitors

Default: SMD 0805

C1,C2,C4,C5,C6,C7,C8,C9,C13,C16, C17,C18,C20 = 100nF

C3 = 100µF, 10V, Case B

C10,C12 = 1µF

C14,C15 = 33pF

C11 = 0.22F, 5.5V, supercap, 5mm pitch

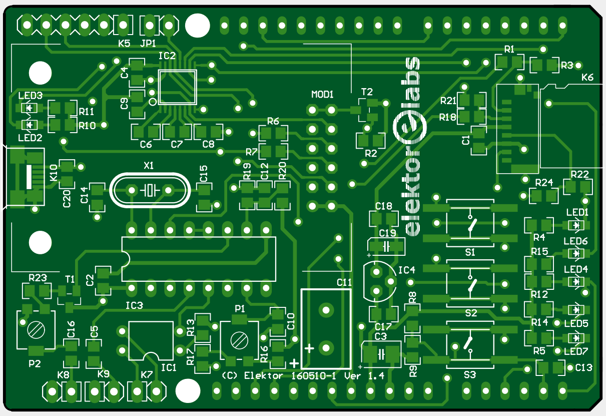
C19 = 2.2µF, 16V, Case A

Semiconductors

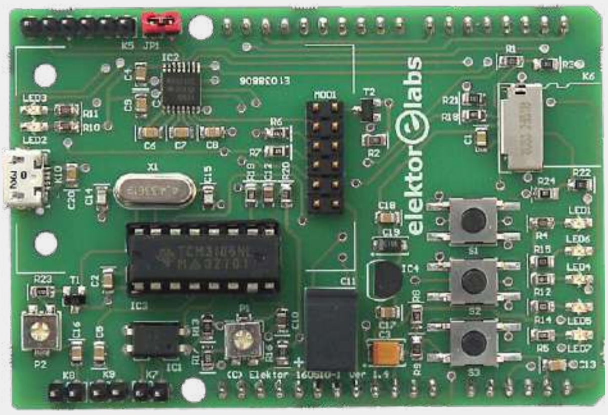
IC1 = HCPL-817-000E

IC2 = MAX3232CPWR

IC3 = TCM3105NL



elektor labs
(C) Elektor 160510-1 Ver 1.4
K5 JP1 IC2 MOD1 T2 R1 R3 K6
LED3 R11 R10 LED2 C4 C9 C6 C7 C8 R6 R7 R2 R21 R18 C1
K10 C20 C14 X1 C15 R19 C12 R20 C18 C19 R24 R22 LED1 R4 LED6 R15 LED4
R23 T1 C2 IC3 C11 IC4 R8 R12 S1 S2 R14 LED5 R5 LED7 C13
P2 C16 C5 R13 P1 C10 R16 R17 C17 C3 R9 S3
K8 K9 K7 IC1



IC4 = LE33CZ-TR

T1 = BC847C

T2 = TSM2302CX RFG

LED1,LED2,LED3 = green, 0805

LED4,LED5.LED6 = yellow, 0805

LED7 = red, 0805

Miscellaneous

K1 = 8-pin pinheader, 0.1" pitch

K2 = 10-pin pinheader, 0.1" pitch

K3,K5 = 6-pin pinheader, 0.1" pitch

K4 = 12-pin pinheader, 0.1" pitch

K6 = Micro SD card socket, Molex 47309 Series

K7,K8,K9,JP1 = 2-pin pinheader, 0.1" pitch

K10 = Micro USB type-B receptacle

S1,S2,S3 = tactile-feedback switch, 6x6.2mm, SMD

X1 = 4.433619MHz quartz crystal,

MOD1 = 12-pin (2x6) pinheader, 0.1" pitch

DIP socket, 16 contacts

PCB #160510-1, from Elektor Store

WIZ107SR Serial-to-Ethernet Module

Elektor Uno R4

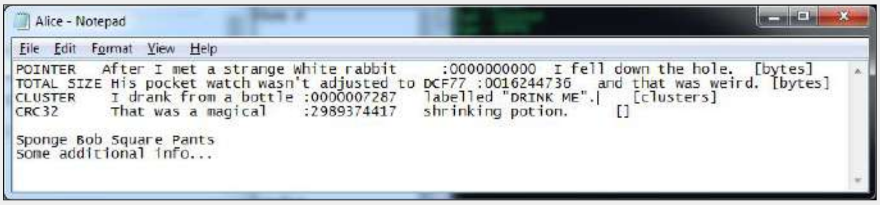
request a retransmit. If the checksum is correct, “Ok” will be sent. The string “To” indicates a time-out when confirmation from the remote peer has taken too long. Mallory could have changed some bytes in the message and spoof the last two bytes to make the CRC-16 match. This is very easy for her to do. If she simply tries to jam the communication channel, it will create an error in the checksum-matching step.

Bob’s OTP Crypto Shield will now try to decrypt the message by scanning for the correct initial key pointer in the current SD card cluster, and its next and previous clusters. Decryption is considered successful if the decrypted CRC-16 matches, and if the decrypted key pointer matches the one counted in Bob’s OTP Crypto Shield. The string “Dc” will be sent to Alice to indicate successful decryption.

It is very difficult for Mallory to spoof this protection as well. Any tampering with the message will result in a message that cannot be decrypted (after the checksums of the encrypted messages matched) and will alert both Alice and Bob of her presence and activity. If Mallory plays back a previously recorded message, it will be decrypted correctly, but the key pointer value will be too low, and Alice or Bob’s OTP Crypto Shield will flag this.

Tips 4 Even More Security

- Defragmenting the key file will make a key pointer search a lot faster when it becomes necessary to go to a previous data cluster on the SD card.
- Format the SD card to FAT32 with a cluster size of 8-16 KB.
- Start your online communication session with a short message to ensure fast key pointer synchronization. It takes more time to find a correct key pointer with a long message.
- When key pointer synchronization is lost, it is always useful to ‘flush’ the buffer with a meaningless sequence of substantial length depending on how much you suspect that your key pointer went back — this will protect you from Eve. Two meaningful plaintext messages encrypted with the same OTP key sequence are required to enable Eve to crack the code.
- If your key pointer has a significantly lower value than the other, increase its value manually in the STT file. It is always better to waste some of the random sequence than to use part of it twice.
- To improve ‘Mallory detection’ hide a few meaningful messages (without a colon) inside your MIR file before starting offline encryption. The MIR file could look something like this



```
POINTER    After I met a strange white rabbit     :0000000000  I fell down the hole.  [bytes]
TOTAL SIZE His pocket watch wasn't adjusted to DCF77 :0016244736   and that was weird. [bytes]
CLUSTER     I drank from a bottle :0000007287   labelled "DRINK ME".|    [clusters]
CRC32       That was a magical    :2989374417   shrinking potion.      []

Sponge Bob Square Pants
Some additional info...
```

- The Elektor Uno R4 bootloader may be useful to speed up development while debugging or trying to improve something, but for secure operation it must be disabled with correct fuse settings and removed from the flash memory. The final version of the firmware without bootloader must be flashed through the ISP port. A bootloader accidentally activated at startup may overwrite some flash memory pages and cause unstable and dangerously insecure operation.

Conclusion

Cryptography is one of the very few areas today where you can’t rely on plug-and-play, out-of-the box, off-the-shelf and foolproof principles without deeper understanding of the whole process. Data security is a complex process, not a product. Consequently, any crypto device will always remain a work in progress.

The crypto device presented here was designed to prevent many possible attacks that can be easily mounted against a general-purpose PC, the most commonly used crypto device today. It was designed to be more secure than a PC or a black-boxed cryptophone, and I believe it is. If you can think of security aspects that I have overlooked, please post your suggestions at the project’s Elektor Labs page [6]. For example, I used CRC, but some non-linear hash function would probably be better. I am also planning to protect it against TEMPEST attacks, but this is a complex problem that will have to be dealt with separately. ⏮

180543-01

Links and literature

[1] Elektor TV on Cryptography: www.elektormagazine.com/news/elektor-tv-historic-criptography

[2] Wim van Eck, “Electromagnetic Radiation from Video Display Units: An Eavesdropping Risk?”, 1985: https://cryptome.org/emr.pdf

[3] Bruce Schneier, “Secrets & Lies: Digital Security in a Networked World”, Wiley, 2004

[4] Wikibooks on OTP: https://en.wikibooks.org/wiki/Cryptography/One_time_pads

[5] Luka Matic, Elektor Magazine March & April 2017, “Truly Random-Number Generator”: www.elektormagazine.com/150116

[6] OTP Crypto Shield at Elektor Labs: www.elektormagazine.com/labs/one-time-pad-otp-crypto-system

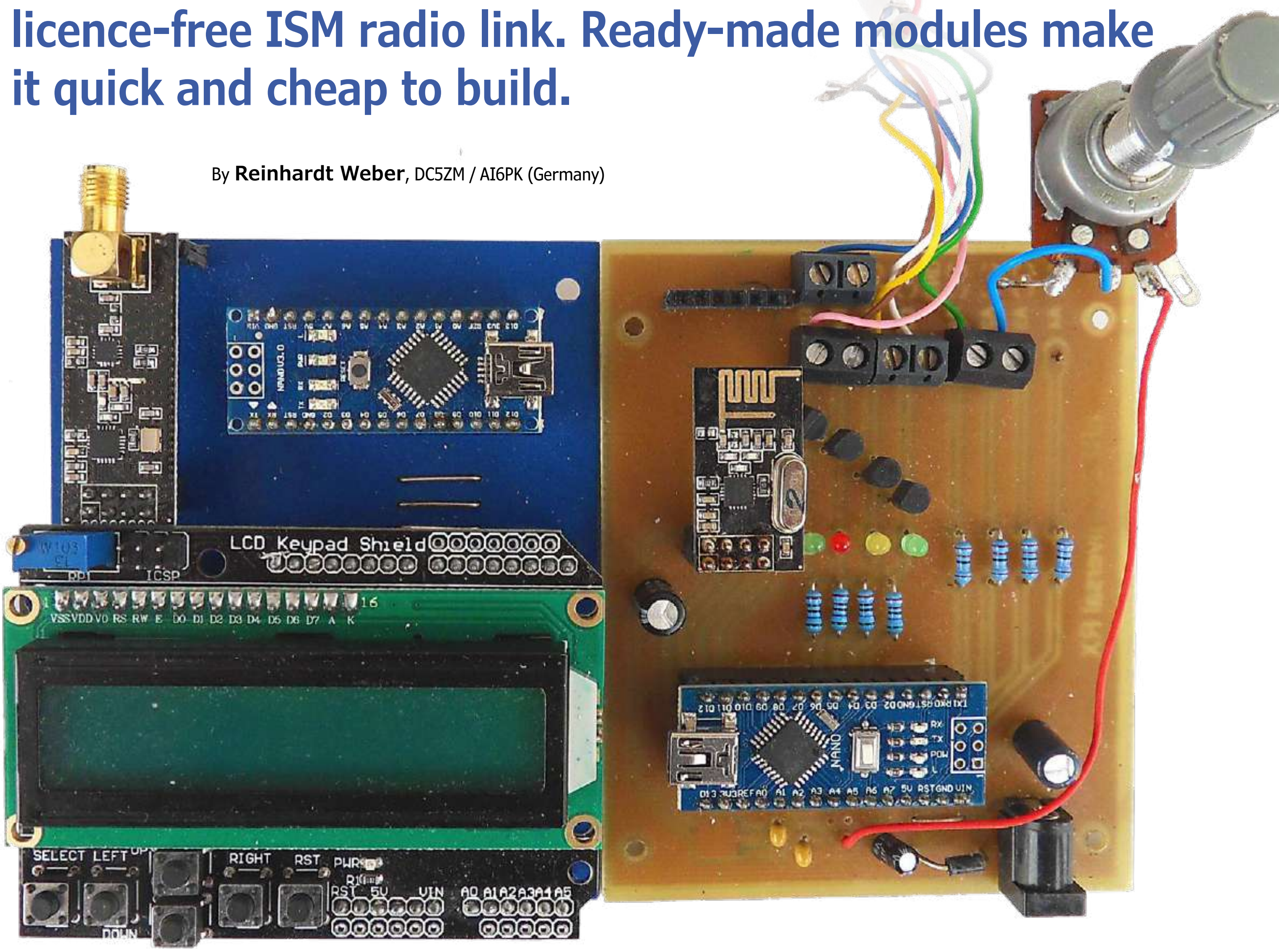
[7] Article resource & support page: www.elektormagazine.com/180543

HOMELAB PROJECT

2.4 GHz Half-Duplex Telemetry

Replace awkward remote-control cables with a licence-free ISM radio link. Ready-made modules make it quick and cheap to build.

By **Reinhardt Weber**, DC5ZM / AI6PK (Germany)



Plenty of devices use a cable (usually multicore) for remote control purposes. Unfortunately, installing wiring of this kind is often none too easy and the result seldom looks attractive. The author was faced with the task of replacing not one but two bulky six-conductor control cables between his amateur radio shack and a rotator control unit on the roof truss. The outcome was a solution that is applicable for all kinds of purposes, providing four digital and four analogue radio channels in half-duplex mode.

This project comprises two electronic assemblies, a transmit device in the shack and a receive devices at the rotator (rotor), linked by a single radio channel. Since each unit can both receive and transmit (albeit not simultaneously but only in turn), we are dealing with half-duplex communication, meaning that it would be more accurate to describe this in terms of a 'front end' or control unit (rather than the transmitter) and a 'back end' (the rotator control mechanism) rather than the receiver. Pictures are often clearer than words and you can visualise the basic arrangements in **Figure 1**.

Modules used in the front and back end units

The hardware used at each end is characterised by the fact that it consists mainly of ready-built modules. An Arduino Nano is used as controller in each case. A 2.4 GHz transmitter/receiver ISM radio module nRF24 takes care of each radio link.

Front and back end devices differ only in their user interfaces. Whereas the back end (**Figure 2**) features an analogue input and buffered digital outputs along with four LEDs (for the testing phase), the front end in **Figure 3** is equipped with pushbuttons and an LCD screen. A ready-built module is also used for this, namely a dirt-cheap *LCD Keypad Shield*.

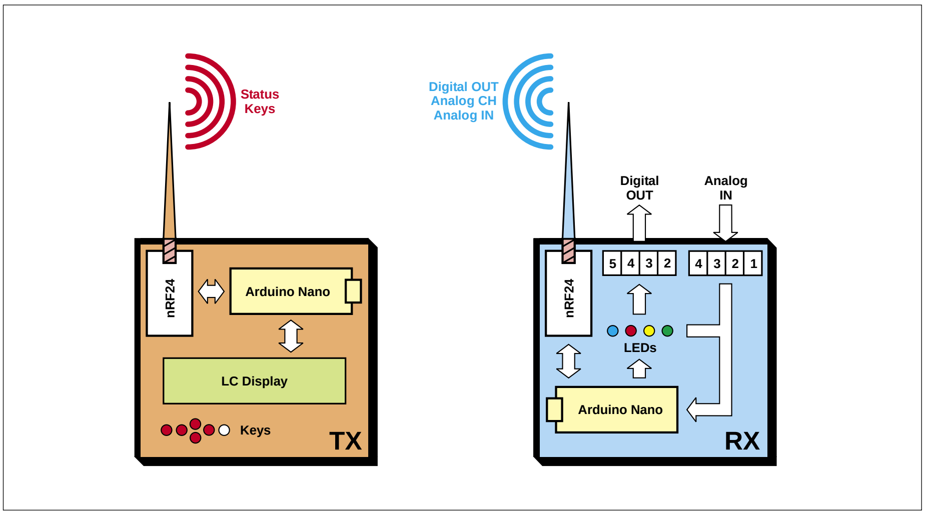


Figure 1. Block diagram of the telemetry system, showing the front and back ends.

Arduino Nano

Not much needs to be said about the Arduino Nano [1][2]. In function, it is largely compatible with the well-known, standard entry-level model Arduino Uno, but it is also smaller and cheaper. In this application both units are powered using plug-in DC adapters. The Arduino Nano can be powered by an unregulated supply voltage of 6 to 20 V, but because its internal voltage regulation relies on simple linear regulators, these get quite hot when they have to cope with relatively high input voltages. If the input voltage is restricted to between 7 and 9 V, things are a lot more bearable. If a higher voltage is used, a series resistor of 47 Ω should be included in the supply line; together, this and the reverse polarity protection diode dissipate part of the excess energy.

The nRF24 radio module

These popular radio modules, which transmit and receive on 2.4 GHz, are built into many wireless applications and have already been used in many Elektor projects. Not only are they inexpensive but there are also suitable libraries in the Arduino IDE that support these modules. The modules work largely without attention; all you need do is to integrate the library into the program, without the need to give any further thought to radio matters. The nRF24 modules are controlled via the SPI bus of the Arduino Nano. Up to 128 channels are available for use, although obviously the chips doing the communication must use the same channel to work with one another. The modules, which go by the title of breakout boards for the Nordic nRF24L01 chip, are available in a large number of versions: with a choice of printed onboard or external SMA antennas, high or ultra-low power, with or without onboard 3.3 V voltage regulator, with integrated power amplifier and LNA receiver amplifier plus above all, a host of different connection configurations. Which version you use depends on the space available. If you wish to transmit only through a wooden ceiling up to the roof truss, the module with printed antenna [3] is completely

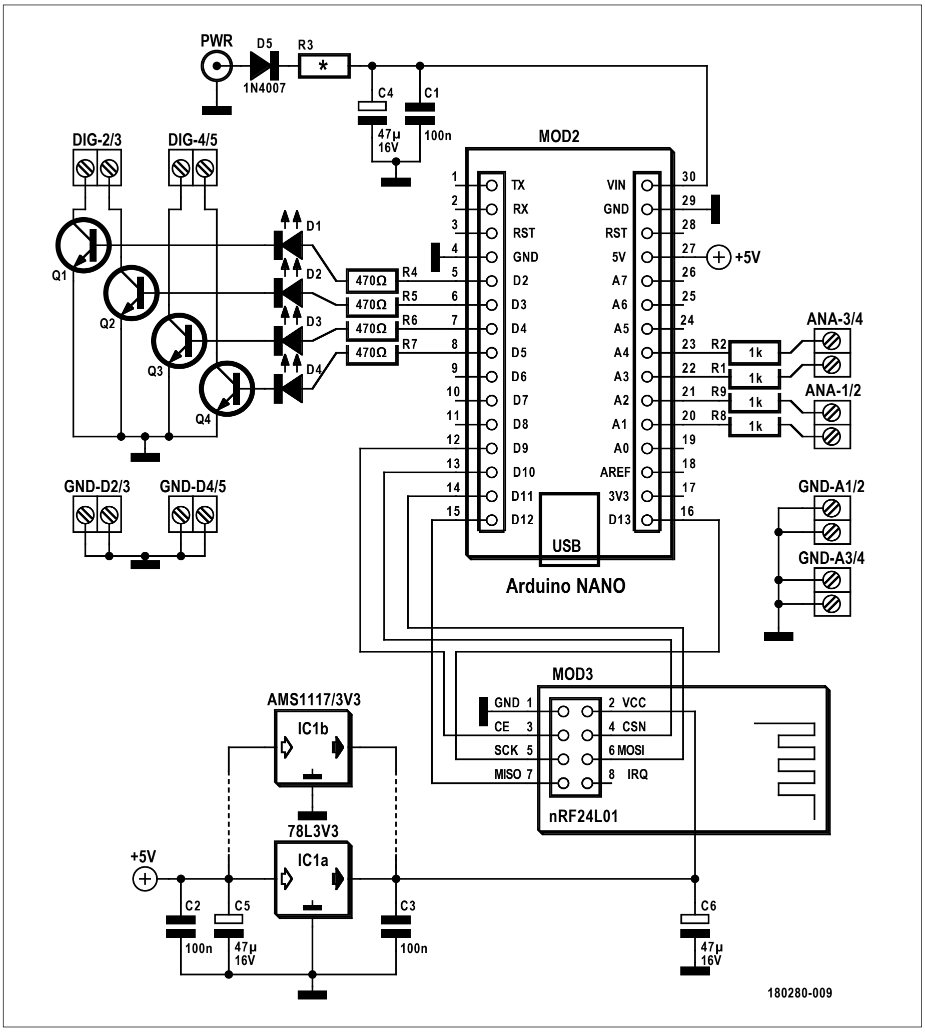


Figure 2. Schematic for the back end with Arduino and radio module.

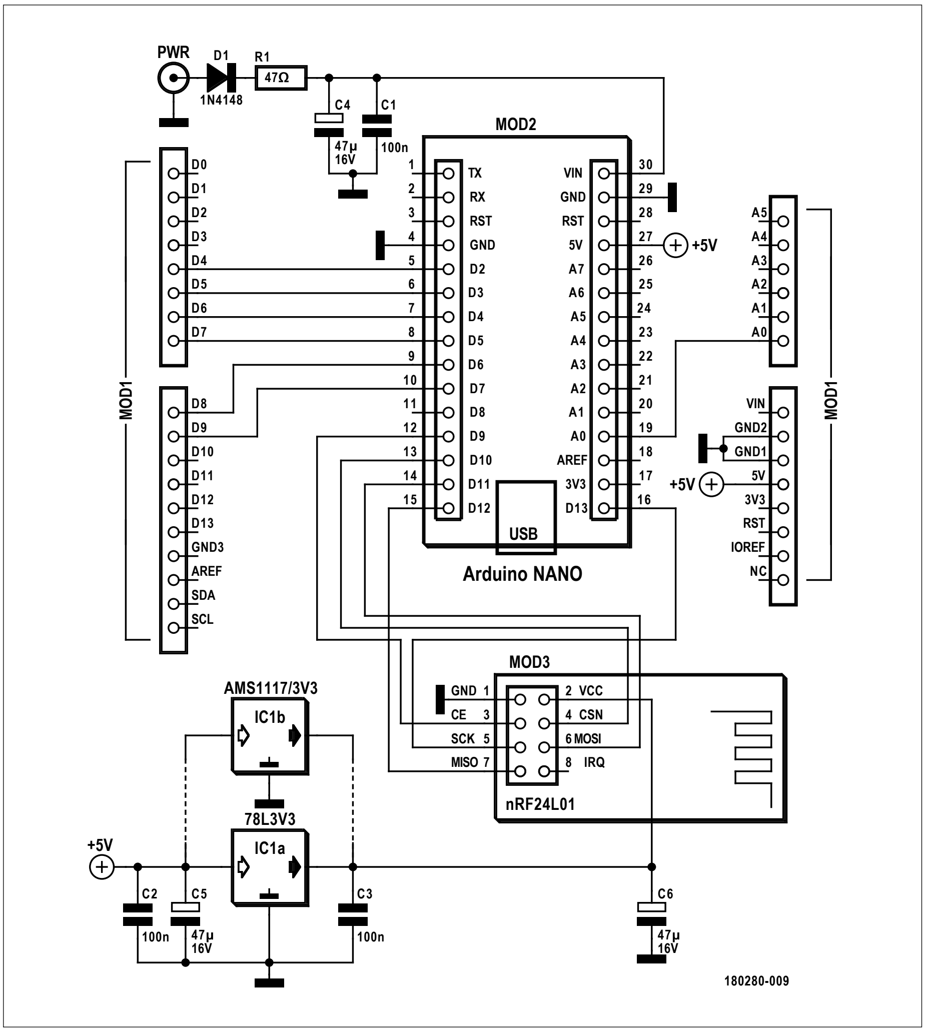


Figure 3. The front end comprises an Arduino, radio module and LCD Keypad Shield.

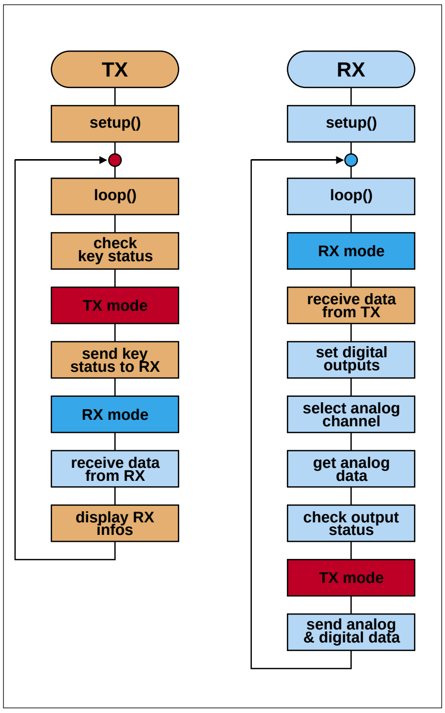


Figure 4. Communication between the two units in flow diagram form.

adequate. On the other hand, if the front and back ends are separated by several floors of reinforced concrete, you need to use a version [4] that promises maximum transmit and receive power. Note that the 1 km range specified by the manufacturer is achieved only under ideal conditions outdoors. In the situation described here, you can realistically expect a range similar to what a WLAN router would manage. Incidentally, Nordic itself no longer recommends the nRF24L01 for new designs, but refers you to the more powerful successor model nRF24L01+. All the same, stocks of the 'old' breakout boards are expected to remain available on the market for some years to come.

The transmit/receive module does not operate on 5 V like the Arduino, but with 3.3 V instead. In principle, the transceiver module could be operated directly using the regulated 3.3 V output of the Arduino Nano, which supplies up to 50 mA. This would be entirely sufficient for the simple radio module with printed PCB antenna (11 to 12 mA), but not for the version with amplifiers, since it draws up to 115 mA in transmit mode. For this reason, here we supply the radio module using an additional voltage regulator, which is connected to the 5 V connection of the Arduino. There is even a module custom-made for this purpose, but our preferred implementation employs the classic arrangement of a voltage regulator (SMD or TO92), together with the usual decoupling capacitors.

LCD Keypad Shield

The '1602 LCD Keypad Shield For Arduino', offered for instance by the Shanghai-based company DFRobot [5], is designed to match the Arduino Uno mechanically but of course, as seen here, it can also be used with the Nano. The display with 2x16 characters is not driven serially using SPI or I^{2}C, but instead using the parallel bus. Display pins DB4 to DB7 are led out to pinheader terminals D4 to D7, also control pins Reset and Enable are taken to D8 and D9 respectively. The display is equipped with a trimpot for adjusting the contrast of the LCD.

The circuitry of the five pressbuttons SELECT, LEFT, UP, DOWN and RIGHT is somewhat unconventional and not as you might expect. They are not routed separately to individual digital inputs, but instead, all five are commoned together to a single analogue connection(A0), as shown in **the text box**. Internally, the buttons are connected to a resistor divider, requiring the controller to evaluate the analogue voltage at A0 in order to detect when a button is pressed and if so, which one. This also means that only one button may be pressed at a time. The Shield has another button (RESET), which is connected to the RST pin of the Arduino, but it's not used in this application. Thomas Clausen has dealt with the Shield in greater detail and he describes it at [6].

How half-duplex communication works in practice

The communication process between the front and back ends is illustrated in two flow diagrams (**Figure 4**). In principle, a single keystroke is detected at the front end and this information is then passed to the back end, where processing takes place. The UP, DOWN, LEFT and RIGHT keys each cause one of the digital

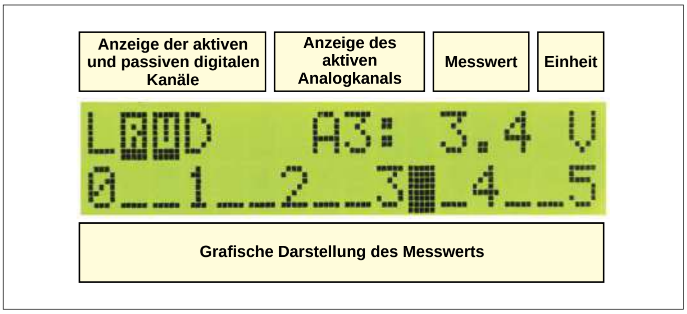

Anzeige der aktiven und passiven digitalen Kanäle
Anzeige des aktiven Analogkanals
Messwert
Einheit
L R U D A3: 3.4 V
0__1__2__3_4__5
Grafische Darstellung des Messwerts

Figure 5. This is how the LCD screen for rotator control looks.

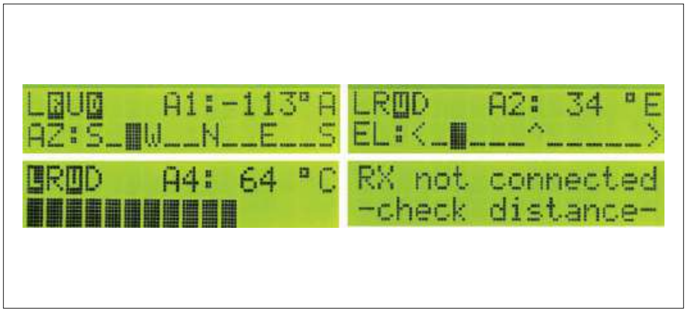


Figure 6. Sample custom characters designed for various applications.

outputs to be set to Low; the SELECT key enables the user to switch between four available analogue inputs in order to enter one of four measured values.

In the program for the front end (TX.ino), following initialisation, the controller evaluates the status of the keys in the main loop by measuring the analogue voltage at A0 and determining which of the five keys was pressed. Note that several variants of the Shield are available commercially and, depending on the variant purchased, the values of the divider resistors may differ. Accordingly, it may be necessary to change the threshold values in **Listing 1** so that the keys are detected reliably. The program then switches to transmit mode, informs the back end which key is pressed, and switches to receive mode.

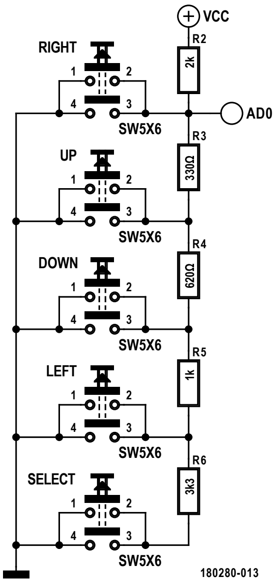
The back end, which is in receive mode when the RX.ino program is started, sets one of the four digital outputs or switches to another analogue input as appropriate on the basis of the received information. The program then reads the status of the four digital outputs, the analogue channel that has been set and its analogue voltage. The measured voltage is converted into a 10-bit digital value. The back-end switches to transmit mode and finally transmits all these data (in encoded form) to the front end in one go. After receiving this, the front end decodes the data and displays the received information on the LCD display (**Figure 5**).

During this bidirectional communication process, the nRF24L01 transmit/receive chip checks continuously for correct transmission. In the event of an error, a corresponding message is shown on the display.

Software variations

The two programs, which you can download from the project page [7], include sample scripts for a variety of applications. In the original application, the UP, DOWN, LEFT and RIGHT buttons actuate the antenna rotator by setting their outputs to Low. To illustrate the options, three different switching functions are shown in the listing for the front end: LEFT as a push button, RIGHT as a switch, and UP and DOWN as radio buttons (mutually interlinked pressbuttons). The SELECT button toggles between the four available analogue inputs. The LCD displays the number of the analogue input and its measured value. In the listing for the front end you can also see examples of how the display could appear if it showed the azimuth or the elevation (other analogue channels are not necessary with an aerial rotator). The program code also illustrates how a small display for a voltmeter or a thermometer could be produced (**Figure 6**). The author has used the practical Character Generator [8] to create the special symbols.

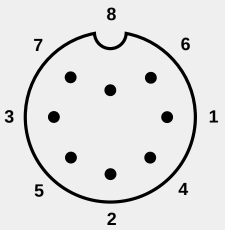
The two commented programs are intentionally quite simple and offer plenty of opportunities to adapt them to your own needs. All the code lines and functions



Listing 1. Evaluating keys pressed

```
void get_key_pushed()
{
   message_to_RX = analogRead(A0)/10;               // values from 0 to 102
   delay(100);    // key debounce

   if (message_to_RX<5){while(analogRead(A0)<5); message_to_RX = 2;} // Right
   if (message_to_RX> 5  && message_to_RX<20 ) message_to_RX = 3;        // Up
   if (message_to_RX>20  && message_to_RX<35 ) message_to_RX = 4;      // Down
   if (message_to_RX>35  && message_to_RX<55 ) message_to_RX = 1;      // Left
   if (message_to_RX>55  && message_to_RX<85 ) message_to_RX = 5;    // Select
}
```

Rotator control



Pin 6	Outputs 2 to 4.5 V_{DC}, corresponding to 0 to 450° (Azimuth)
Pin 1	Outputs 2 to 4.5 V_{DC}, corresponding to 0 to 180° (Elevation)
Pin 4	Rotate left (active Low)
Pin 2	Rotate right (active Low)
Pin 5	Elevate downwards (active Low)
Pin 3	Elevate upwards (active Low)
Pin 7	NC
Pin 8	Shared ground connection

The YAESU G-5600 rotator control unit has four buttons to control the rotator: UP and DOWN for elevation, LEFT and RIGHT for azimuth. Pressing a pushbutton grounds the respective control line. The control lines are also connected in parallel with the buttons to an 8-pin DIN socket at the rear of the rotor control unit. In addition, the rotator control unit displays the azimuth and elevation settings using two large pointer dials. The two analogue voltages displayed on these instruments are generated by the rotator motor itself. They are in the 0 to 4.5 V range and indicate the elevation of 0 to 180° or the azimuth of 0 to 450°. The two analogue voltages are also taken out to the DIN socket on the back of the device, enabling complete remote control of the device.

There is nothing to stop you from putting the control unit totally out of sight, requiring only short control cables to the rotator. You can connect an electronic assembly to the control unit, which is then linked only by an ISM radio link to a small control panel in the shack.

regarding the control of the nRF24 module should be changed only if you have sufficient experience with this particular radio module.

Practical implementation

Circuits and PCB layouts were created with the EAGLE program and can be viewed on the project page [6]. The relatively wide tracks used mean that the PCB layouts are easy to etch at home using simple equipment. After the construction of the circuits, both control programs are loaded with the Arduino IDE. The programs use the following libraries: `LiquidCrystal.h`, `SPI.h`, `nRF24L01.h` and `RF24.h`. One or another of these libraries is bound to be installed already in your Arduino IDE, but if not, you can install it by downloading from [9][10][11][12] and integrating it with your IDE using: Sketch Include Library (*Bibliothek einbinden*)

The use of ready-built modules means that you can fit either a transmitter or a receiver on one half of a single-sided Eurocard circuit board (10 × 16 cm) [13], if you don't mind using a few wire bridges. All five of the modules used can be had from Internet suppliers together for the bargain price of less than €30 / £27 / $35. Note that these modules are available in several variants and the (usually Chinese) vendors don't bother much about documentation, so it's your effort to work out what you have actually bought. For that reason, you may prefer to choose suppliers who provide schematics, PCB layouts and sample programs for their products.

180280-02

Editor's note:
This project previously appeared in the German Amateur Radio Club's magazine CQ DL (issue 4/18).

Web Links

[1] Arduino Nano: www.elektor.com/arduino-nano-3

[2] JOY-iT Nano: www.elektor.com/joy-it-nano-v3

[3] Radio module nRF24L01: www.elektor.com/nrf24l01-2-4-ghz-wireless-module-8-pin-150499-91

[4] nRF2401 with amplifier and LNA : www.elecfreaks.com/wiki/index.php?title=2.4G_Wireless_nRF24L01p_with_PA_and_LNA

[5] 1602 LCD Keypad Shield: www.dfrobot.com/product-51.html

[6] LCD Keypad Shield in detail: www.thomasclausen.net/en/walking-through-the-1602-lcd-keypad-shield-for-arduino/

[7] Project page: www.elektormagazine.com/180280-01

[8] LCD custom character generator: https://omerk.github.io/lcdchargen/

[9] LCD Library: https://playground.arduino.cc/Main/LiquidCrystal

[10] SPI Library: https://github.com/PaulStoffregen/SPI

[11] RF24 Library (1): www.arduinolibraries.info/libraries/rf24

[12] RF24 Library (2): https://github.com/maniacbug/RF24

Elektor Labs Pipeline

Here are another four pointers to interesting little projects posted by liberal contributors to Elektor Labs dot com, so you can see what's on the (electronics) minds of our readers.

From simple to complex, from measuring to controlling, Elektor Labs is a place for all projects about electronics. Check out this selection of projects, there may be something in it that you can use, or contribute to.

How steep is that slope?

If you are into cycling and follow big events like — in chronological order — the Giro (Italy), the Tour de France or the Vuelta (Spain), you will hear and read a lot about the 'categories' of hills and mountains and the steepness of slopes. But how to relate that information to the slopes you encounter when you ride a bike yourself? How tough is your workout compared to the pros? With this little project you can find out.

@ Elektor Labs: https://goo.gl/jjcw8x

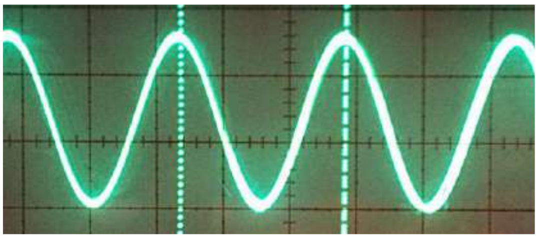
Send your secret messages in a secure envelope

Messages can be encrypted to prevent their contents to be intercepted, tapped or altered. However, to read the message the intended receiver must be in possession of the decryption key, which raises the question of how to send the key in a secure way to the destination? This project tries to solve this chicken-and-egg problem by adding tamper detection to the envelope.

@ Elektor Labs: https://goo.gl/rnk28d

Build a 3-component sinewave oscillator for €/£/$ 1

Sinusoidal waves are important in electronics. Many applications need sinewaves to function, and a lot of electronic circuits can be tested by firing a sinewave at them. Consequently, sinewave generators are handy things to have within arm's reach. If you don't have one, build this project. It only needs 3 (three) parts, and will not break the bank for sure.



@ Elektor Labs: https://goo.gl/YRmP1J

Programmable refrigerator watchdog

Building this little project is awarded with a device that monitors & displays the current refrigerator temperature, as well as logging the time the door was opened and for how long. Alarm thresholds can be set on maximum allowed temperature and door-open time. ⏮

(180571)

@ Elektor Labs: https://goo.gl/pBeCuH

PobDuino Board

Flowcode and Arduino together fight obsolescence

By **Jean-Noël Lefebvre** and **Stéphane Huet** (France)

Are you looking for a programmable and modular robot for working with your students? Construct one or many robots with the PobDuino board. With its help, your robots will be flexible and adaptable: they will be programmable in Flowcode (the basic license is sufficient) and will be able to use Arduino compatible boards to connect sensors and actuators.

Origins

Stéphane Huet is a teacher in Engineering Sciences at the Lycée Louis Aragon de Givors (France). He has been using the Pob robot from the company Pob Technology for several years to introduce his students into robotics. The school has also actively encouraged the design of the mechanical parts of the robot as an educational exercise. The school thus has a fleet of around 10 robots, like many other schools in France.

The principal features of this robotic platform are:

- robustness et mechanical modularity
- numerous accessories (servo motors and sensors)
- possibility to program in the C language, and above all with the

graphical language ORBEE (flowcharts), specially developed by Pob Technology

Unhappily, Pob Technology has gone out of business and Pob is now an orphan. Stéphane has sought to give a second life to his Pobs. He imposed these specifications, among others:

- keep the Mechatronic elements (mechanics, batteries, motors, sensors and actuators);
- allow programming with flowcharts, as with Orbee;
- open the platform to the Arduino environment, which will allow use of the plethora of hardware modules and source code libraries available.

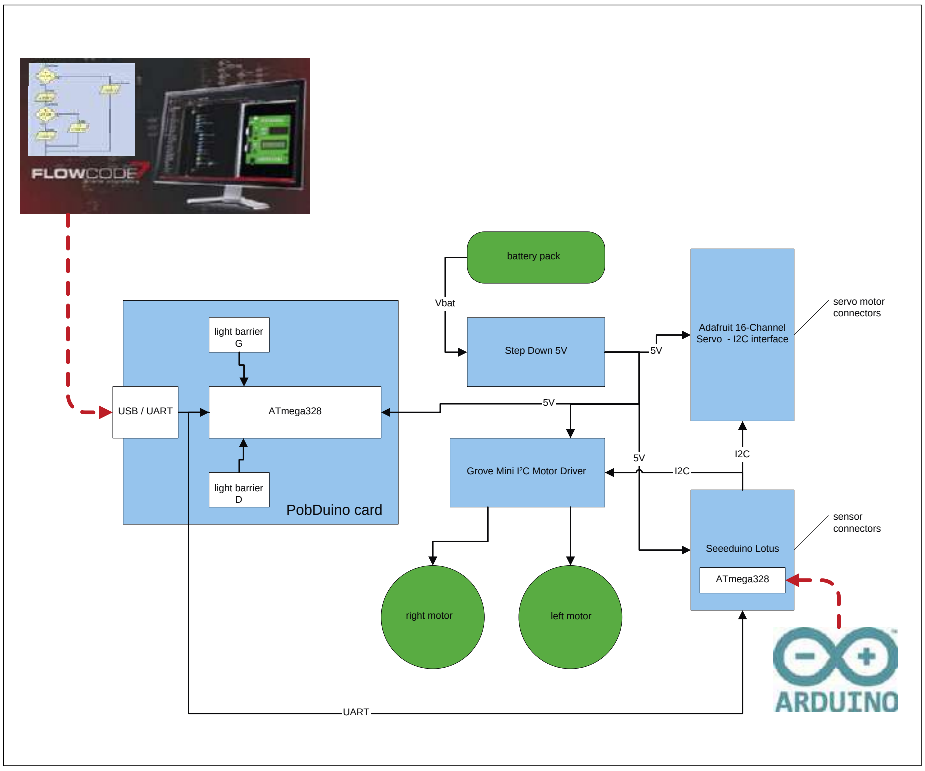


Figure 1. The various functional blocks at the heart of the dual control PobDuino system.

Dual control

For the software, Scratch was ruled out for two reasons. Firstly Scratch does not work with flowcharts, but with graphics called 'structograms' which do not correspond to the educational objectives. Secondly, scratch does not work by itself on an Arduino, it requires a link to a computer which can execute Scratch (or alternatively to use Scratch on an embedded Raspberry Pi board).
After some research, Jean-Noël suggested Flowcode, a serious candidate used by numerous schools. Nevertheless there were a few problems to resolve:

- The Flowcode editor mentions compatibility with Arduino, but only at the level of recognising the hardware of some Arduino boards. On the other hand, there is no software integration; it is not possible to use directly the Arduino EDI libraries:-(.
- Once you have acquired the basic Flowcode licence for Arduino / AVR, the possibilities remain very limited. In fact, to control peripherals (I2C, UART, servos, etc.), you must either get extra licences for the Flowcode

Principal commands already available

DIGITALREAD,Channel(4-8) → 0/1
Reading a digital input on the Lotus board microcontroller.

DIGITALWRITE,Channel(8-15),Value(0-100)
Writing to a PWM output on the Adafruit servo board.

ANALOGREAD,Channel(0-3) → 0-100
Reading an analogue input on the microcontroller of the Lotus board. In his practical exercises, Stéphane uses this input to read the line-following sensors.

DISTREAD,Channel(0-3) → D(cm)
Reading (in cm) of a Sharp GP2Y0A21 distance sensor, connected to an analogue input on the microcontroller of the Lotus board.

USREAD,Channel(4-7) → D(cm)
Reading (in cm) of a Grove (ref. 101020010) ultrasonic distance sensor, connected to a digital input on the microcontroller of the Lotus board.

SERVO,Channel(0-7),Degres(0-180),Speed(0-31/32)
Angular command of a servo motor connected on the Adafruit board.

MOTOR,Side(L/R),Sens(F/R),Speed(0-100)
Command for a motor connected on the Grove I²C motor driver board.

MOVE,Sens(F/R),Dist(mm 0-30000),Speed(0-100)
Command to move the robot in a straight line (distance measured by the coded discs / optical sensors).

ROTATE,Sens(L/R),Angle(1-360)
Command to pivot the robot (angle measured the coded discs / optical sensors).

STOP
Stop all motor commands in progress.

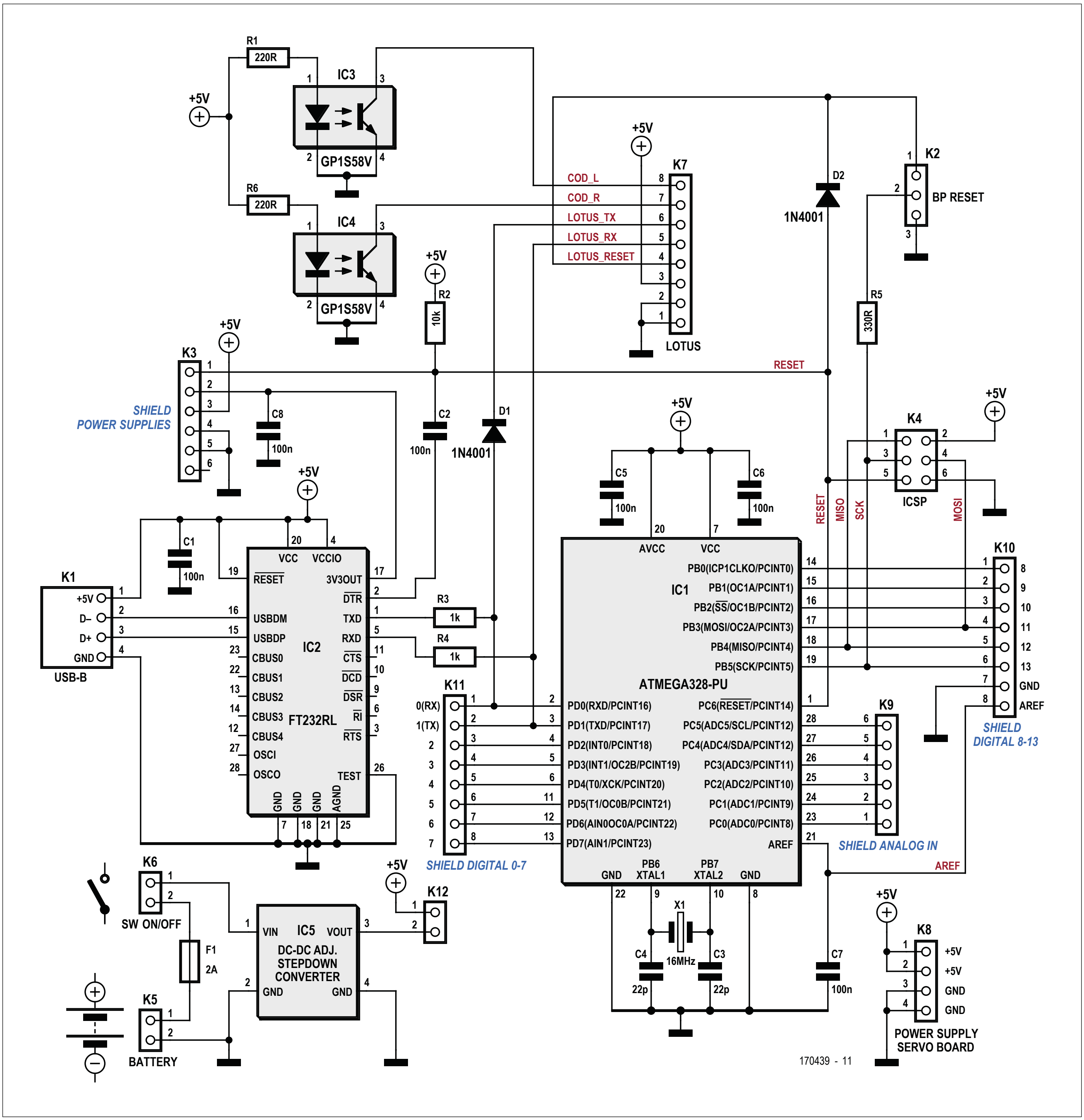


Figure 2. Circuit diagram of the PobDuino board.

libraries, or implement these functions in C language and then integrate them with Flowcode.

To overcome these obstacles, the PobDuino architecture uses two ATmega328 microcontrollers. The first runs the code produced with Flowcode, which does not call any particular peripheral library. However the basic UART functions are implemented.

The second microcontroller is housed on an Arduino compatible board which can use all the hardware peripherals of the Arduino environment. A connection between the UARTs of the two microcontrollers (simple service protocol exchange) allows unlimited use of the Arduino resources from within Flowcode. **Figure 1** gives an overview of the various functional blocks at the heart of the PobDuino system.

Hardware

The only board to make has been christened 'PobDuino'. It contains:

- An ATmega328 microcontroller
- Optical sensors (for the coding wheels of the motors)
- An FTDI interface for programming via USB

This card interprets the Flowcode pro-

gram, but also sends commands to the different parts of the robot.
Its circuit diagram (**Figure 2**) is directly inspired by Arduino Uno boards (old version). At the heart of this card we have the famous ATmega328P microcontroller, together with its 16 MHz crystal (X1) and power supply decoupling capacitors C5, C6 et C7.

ICSP programming is possible via the connector K4, for example to flash the Arduino bootloader. We also find the indispensable 'shield' connectors: K3, K9, K10 et K11. For UART communication by USB, this mission is carried out by a classic specialist circuit FT232RL; this is the only surface mount component which may require fine soldering.
The rest of the components are not from a classic Arduino, but have functions specific to PobDuino: IC5 is a DC-DC converter module which supplies 5V from a battery pack. IC3 and IC4 are the optical sensors for counting the rotations of the robot's wheels..
The other boards of the robot are commercially available.
The Seeeduino Lotus board carries the second ATmega328P microcontroller as well as twelve Grove connectors – to these can be connected sensors from the Grove series [2]. An Adafruit driver board controls the servos. An I2C Grove motor controller drives the DC motors. You can find information on all these boards online.

The assembly and the connection of all the boards are described in the menu and with numerous photos in the file 'Assembly of PobDuino' which may be downloaded from the article page [5]. **Figure 3** gives an overview of the connections between the various boards.

Code

The two ATmega328P microcontrollers communicate between themselves via a UART serial link. The microcontroller of the PobDuino board sends commands to move the robot, executed by the Lotus board (an acknowledgement is sent for every command executed).
On the PobDuino board, an Arduino bootloader must be installed. You can do this yourself by following the Arduino tutorial [6] or just buy a preprogrammed microcontroller. Thereafter, it is Flowcode that runs on the PobDuino board with the files PobDuinoLib.c and PobDuinoLib.h as

More about the authors

Jean-Noël Lefebvre, from his company OOTSIDEBOX, does commissioned work: interactive models, proof of concept, prototypes, small special effects. He also offers an electronic consulting and development service (hardware and software). Finally, he runs introductory workshops on electronics and in particular on Arduino. See www.ootsidebox.com

Contact **Stéphane Huet** on LinkedIn :
www.linkedin.com/in/stéphane-huet-ab552a61/

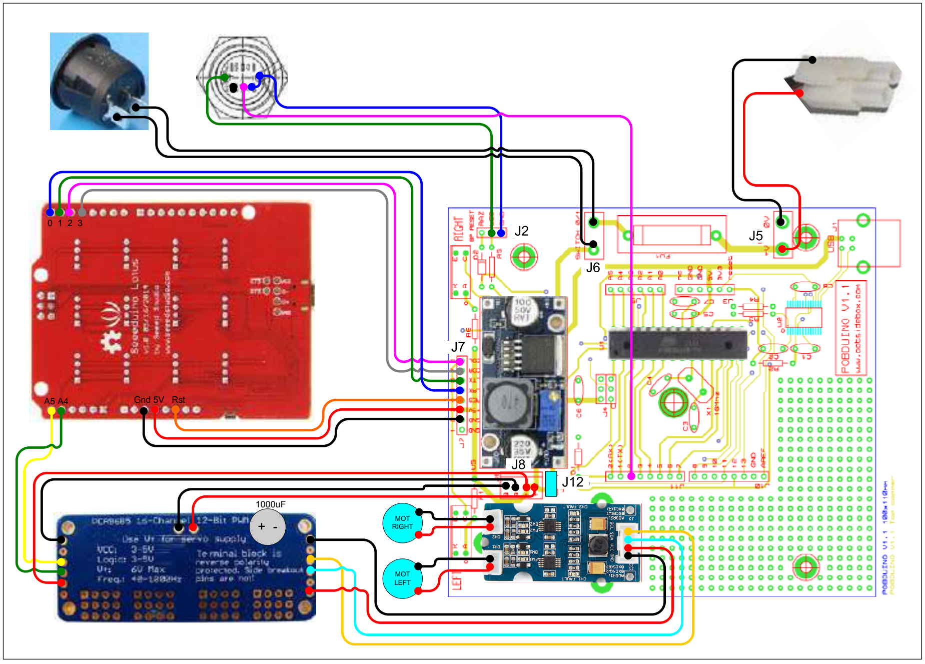


Figure 3. Links between the various boards.

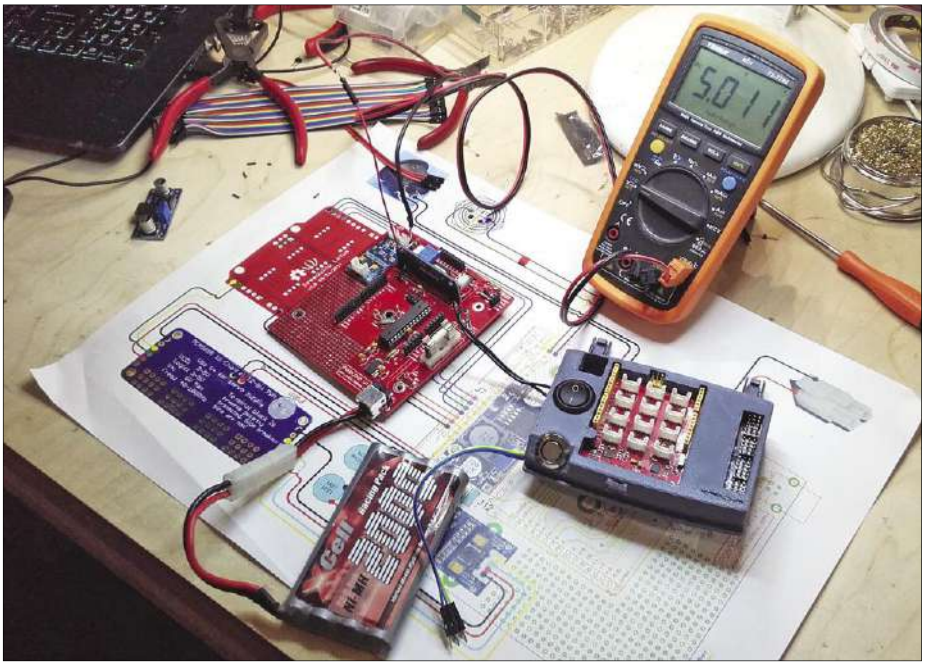


Figure 4. Regulation of the voltage converter.

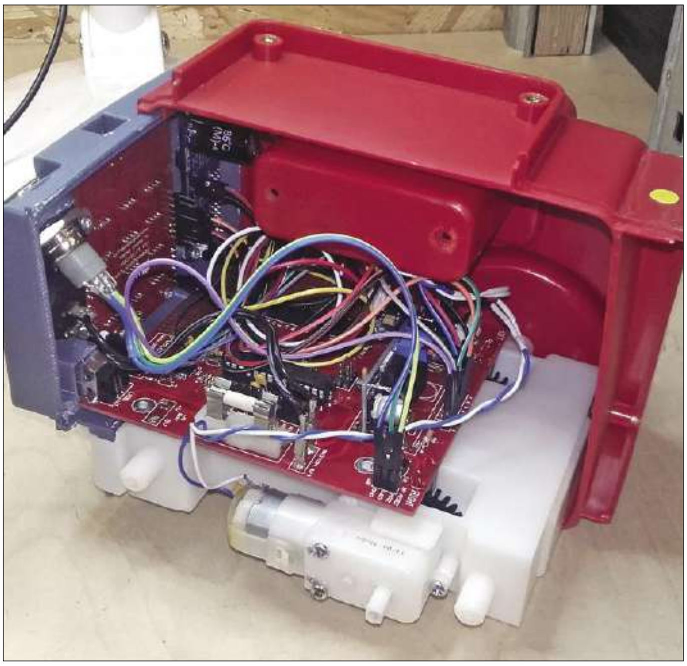


Figure 5. PobDuino with its guts exposed.

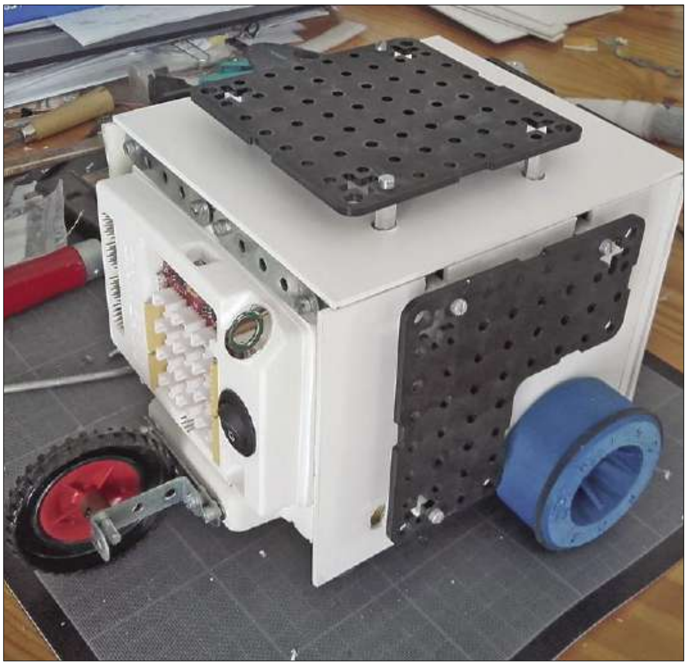


Figure 6. DIY PobDuino chassis built with Meccano parts and white plastic sheets.

well as the various macros (downloadable at [5]).

On the Lotus board, the Arduino sketch POBDUINO_B.ino is loaded; this calls several libraries. The board is thus ready to execute commands sent either from the serial monitor of the Arduino EDI or from the microcontroller of the PobDuino board with Flowcode.

To create a new programme for the robot in Flowcode, the simplest way is to use the file PobDuinoVierge.fcfx. It contains all the available embedded macros (see sidebar), and the options for the project are already configured. You only have to add commands.

You can also open the file HelloWorld.fcfx which tests the compilation, loading and execution by the PobDuino board. When run it sends the text 'Hello World' on the serial link, via the USB port. To make sure that everything is working ok, you need a serial terminal, such as TeraTerm, or even the one that comes with the Arduino IDE. The communication speed must be set to 9600 Bauds. After loading in PobDuino, you need to reset the robot, either by pressing the Reset button of the robot, or by connecting the serial terminal (automatic reset). The text 'Hello World' should thereafter be displayed on the terminal.

The file 'Programming PobDuino in Flowcode.docx' downloadable at [5] will guide your first steps in Flowcode and lists the various commands available.

If these commands don't fit your needs, all you need to do to add new ones is to code them in the sketch POBDUINO_B.ino and create corresponding Flowcode macros. This opens the door to using a huge range of sensors and actuators for Arduino, from Grove and Seeedstudio amongst others: MP3 readers, driving LED Neopixels, reading RFID chips, light sensors…

Conclusion

Thanks to the PobDuino board with open source code, clients of Pob Technology can resuscitate their POB robots. If you haven't got a Pob, that's no problem: Stéphane has supplied the 3D printing file to replace the back panel of the original POB robot. You can build the rest of the chassis with whatever you have to hand (**Figure 6**). You may equally use other available mechanical platforms (Pololu for example). Finally, for students, the file 'Beginning Programming' (under link [5]) contains seven beginners' programmes aimed at second year engineering students; Stéphane has put these at the disposal of his colleagues. ⏮

170439-02

Web Links

[1] Discover PobDuino: http://hackaday.com/2017/07/27/pobduino-makes-the-most-of-grove/

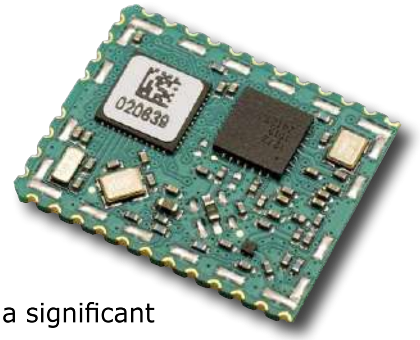
[2] Grove System: http://wiki.seeedstudio.com/Grove_System/

[3] Motor driver board: 'Grove - I2C Mini Motor Driver': http://wiki.seeedstudio.com/Grove-Mini_I2C_Motor_Driver_v1.0/

[4] Article page: www.elektormagazine.com/170439

[5] Installation of the Arduino bootloader: www.arduino.cc/en/Tutorial/ArduinoISP

(Nearly) Everything you always wanted to know about... LoRa



Answers by **Norbert Schmidt**, IMST (Germany)

In the coming Internet of Things (IoT) the LoRa radio data standard could have a significant role to play. Discover its potential in this FAQ file.

Q *What is LoRa?*

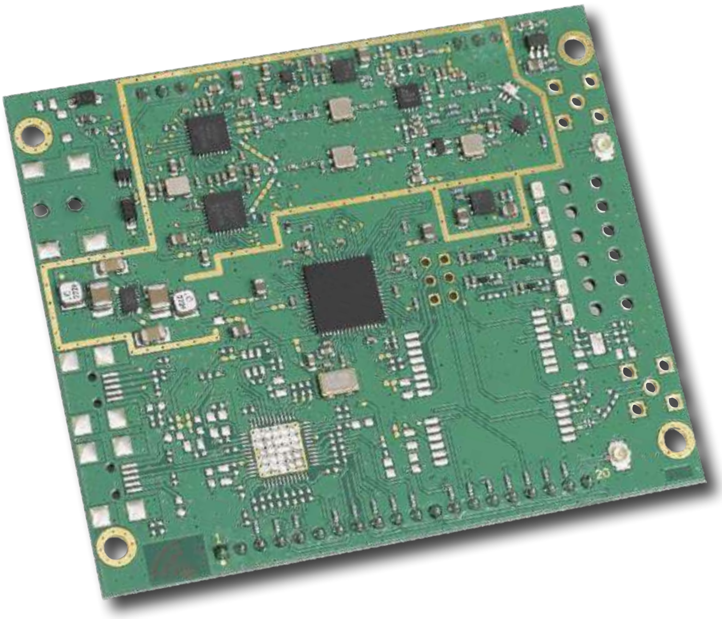
A LoRa™ (short for 'Long Range'), developed and patented by the Semtech company, is a proprietary solution for data transmission over radio in the European license-free 868-MHz ISM band. Because LoRa offers link distances up to 15 kilometers (approx. 10 miles), it makes feasible radio networks in which a single cell can cover an area of many square kilometers. A single gateway per cell then connects the end-nodes to the classic Internet.

Q *How does LoRa differ from rival systems in the WAN (Wide Area Network) domain?*

A The LoRa technology is differentiated by the extremely low power consumption of the end-nodes (particularly in standby mode), which can operate on the same battery for many years. What's more, the cost of end-nodes is very modest compared with other technologies. Unlike the rival SigFox system, there are many providers of LoRa WAN network service and equipment, with competition among operators. Unlike mobile data networks, there are no recurring charges for users.

Q *When can we expect to see network coverage for major conurbations in Europe?*

A The rollout of LoRa networks has already begun in many European countries. In Switzerland, the Netherlands, France, Poland, the Czech Republic, Russia and Germany major communications providers have started to construct public LoRa networks. In addition to these public networks, numerous private networks are cropping up for special applications. To ensure that end-nodes from differing manufacturers can communicate with one another without problems, the LoRa Alliance [1] has already established a certification program for LoRa WAN-compatible devices.



Q *Is LoRa intended only for industrial users or is it something also for fairly small startups, academic groups and so on?*

A LoRa is a radio technology for use in the most diverse range of applications and thus also for widely differing user groups. Alongside public LoRa WAN networks you could also have small private and proprietary systems, which could be implemented and commissioned rapidly thanks to comprehensive development tools and firmware versions offered. LoRa also still has vast potential for research, for example into issues such as network capacity and the scope for localizing the end-nodes.

Q *What sort of hardware is available already?*

A By now all components such as LoRa gateways and corresponding server infrastructure and user terminals needed to construct a LoRa networks are already available. For users who wish to develop their own products numerous 'building block' modules are available, for example the iM880B-L [2] for integration in terminal nodes or the concentrator module iC880A [3], which is a LoRa front-end for gateways. ◀

(150748)

Web Links

[1] www.lora-alliance.org

[2] www.wireless-solutions.de/products/radiomodules/im880b-l

[3] www.wireless-solutions.de/products/long-range-radio/ic880a

What's New in Android Things

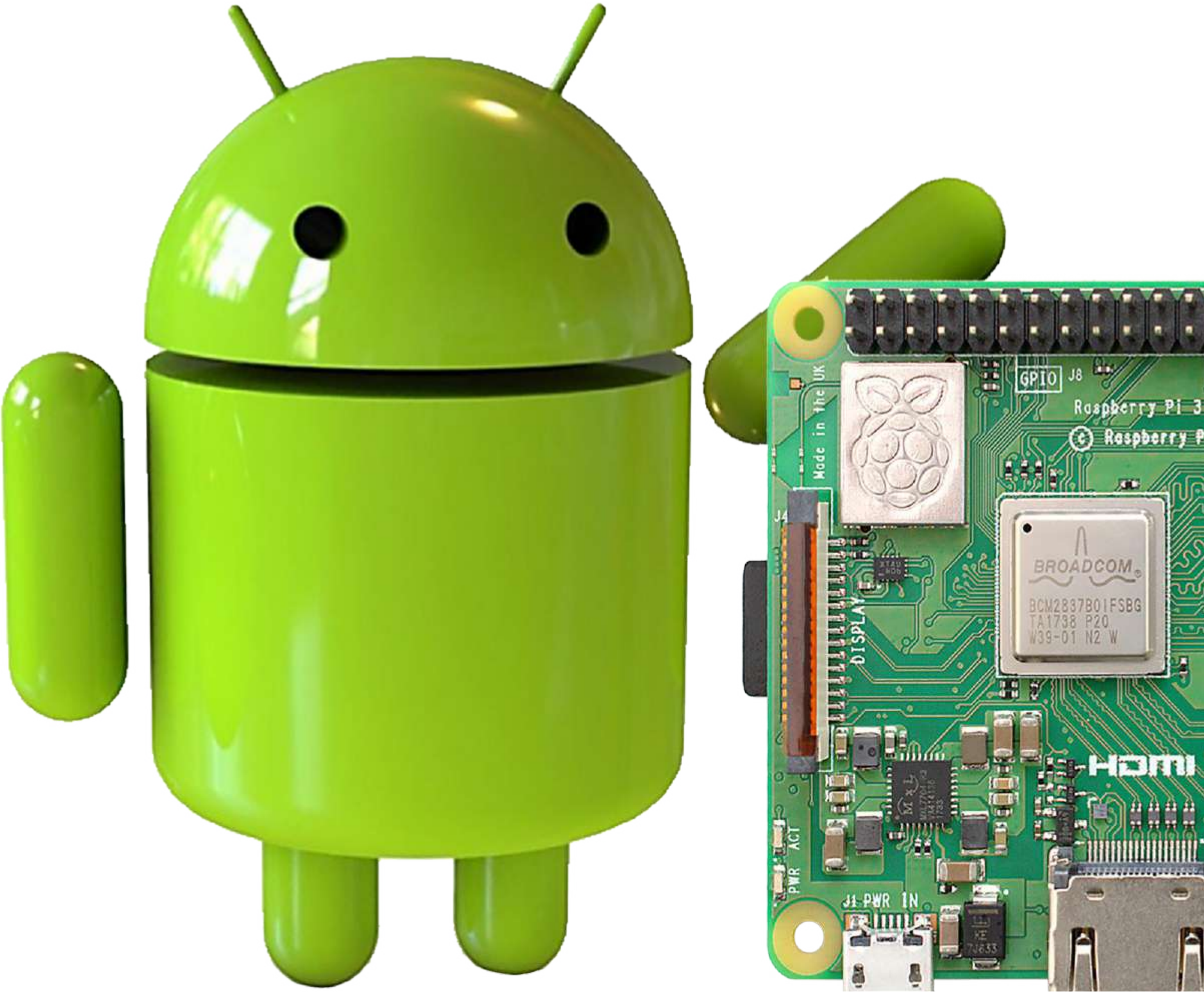
A boost from C++

By **Tam Hanna** (Slovenia)

When Elektor last reported on Android Things [1][2], the operating system was still in a state of active development. Since then Google has reached a milestone in their work with the release of a version with a stable API. This article will take a quick look at what's new. As a host we will, as before, be using a first-generation Raspberry Pi 3, and of course a Google account will be required.

There are many changes to be seen in Android Things 1.0, including a new permissions architecture and changes to the GPIO API. Google has also taken the opportunity to introduce a C++ programming interface and improvements to the update system.

For the experiments described below we require a Raspberry



Pi 3 equipped with a mouse, keyboard, screen and an Internet connection. Our starting point is the management console for Android Things described at [3], which can be used to configure devices and generate images [4]. It is a good idea to create a new 'product' in the Android Things Console for the purposes of these experiments.

Switch to the model automatically generated by Google in order to set up a build configuration. The concept of 'model management' was introduced by Google to allow developers to handle multiple hardware and software versions within a single product category.

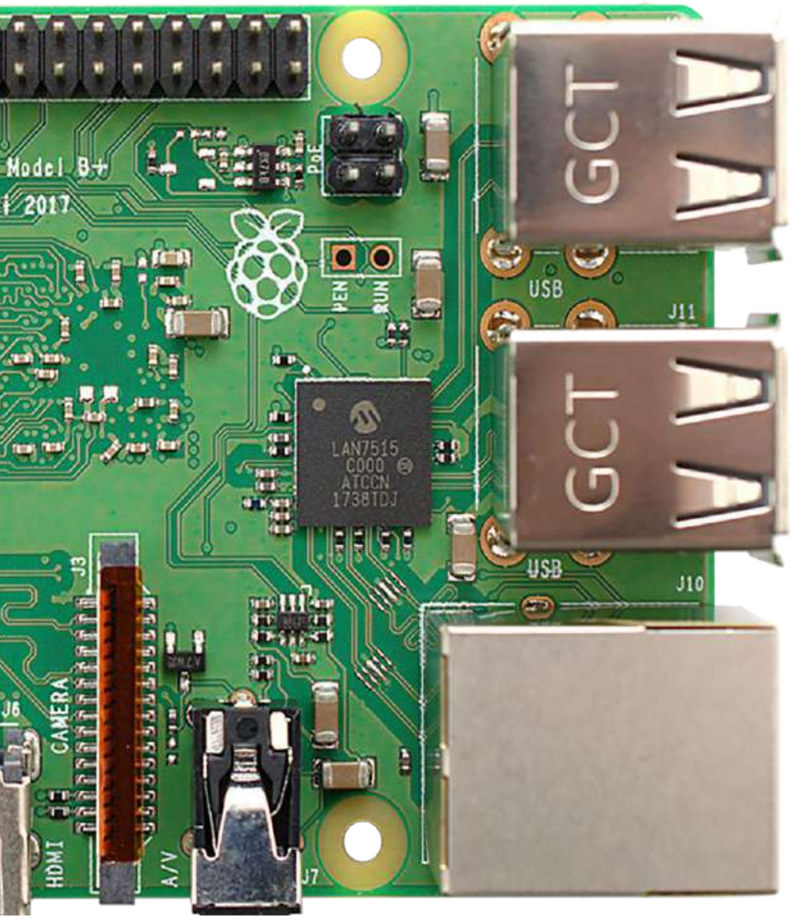
As in our previous experiments, creating the configuration requires a certain amount of time. Once the development image is ready, download it and install it as usual onto an SD card [1].

The new GPIO API

Open Android Studio and create a new project in the usual way. Current versions of the IDE (from version 3.2 onwards) offer an option in the project generator to create a project suitable for use with Android Things. For the target system select Android Oreo, and set up an activity of type Android Things Empty Activity.

Release versions of Android Things require the configuration of 'permissions' if the code to be executed is to have access to the hardware. For our experiments with GPIO pins it suffices to add the following rights to *AndroidManifest.xml*:

```
<xml version="1.0" encoding="utf-8"?>
<manifest . . .>
   <uses-permission android:name="com.google.android.
    things.permission.USE_PERIPHERAL_IO" />
```



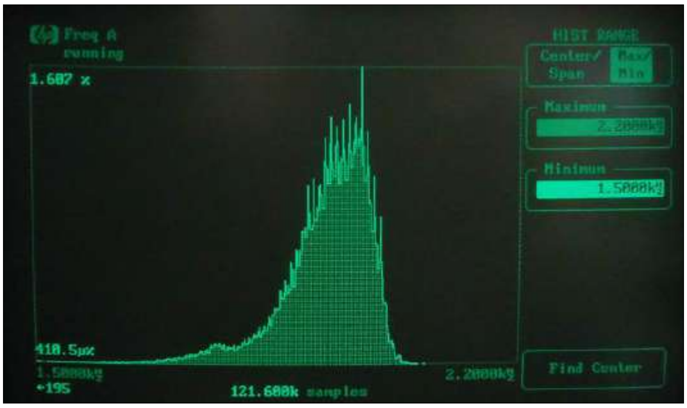


Figure 1. Not the most stable of waveforms.

As with the full-fat versions of the operating system, Android Things works in tandem with the GUI thread. We therefore need to create a class derived from Thread which will be launched in the `onCreate` method of *MainActivity*.

As in our previous articles we will generate a squarewave signal in order to get an idea of the execution speed and real-time stability of the system. We now look at the code required to output this waveform.

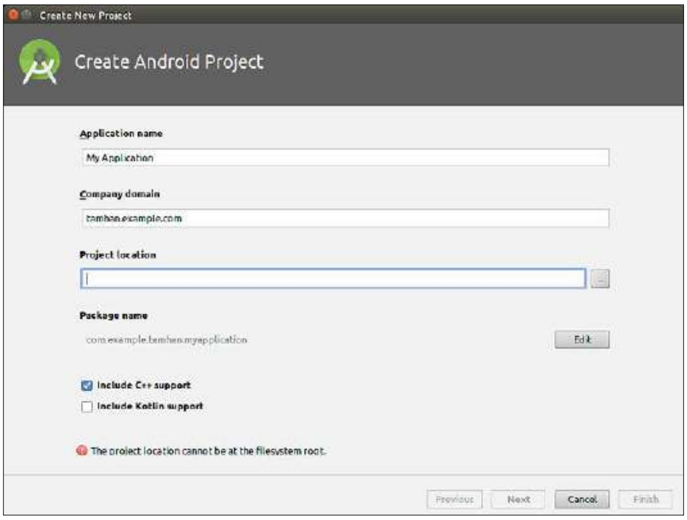
```
public void run() {
    Gpio myGPIO0;
    try {
        PeripheralManager manager =
        PeripheralManager.getInstance();
        myGPIO0 = manager.openGpio("BCM2");
```

Android Things 1.0 differs from its predecessors in how hardware components are initialized. Instead of the PeripheralManagerService there is a full-featured platform service that works in the same way as every other system service on Android. That means that we use getInstance() to obtain the instance that we need to use.

The rest of the function runs along the same lines as before.

```
        myGPIO0.setDirection(Gpio.
        DIRECTION_OUT_INITIALLY_LOW);
        while(1==1){
            myGPIO0.setValue(true);
            myGPIO0.setValue(false);
        }
    }
    catch (Exception e) { }
}
```

This code was run on a Raspberry Pi 3 operating at 1.2 GHz. The output was connected to a modulation domain analyser, and **Figure 1** shows the result.



Application name

My Application

Company domain

tamhan.example.com

Project location

Package name

com.example.tamhan.myapplication

☑ Include C++ support

□ Include Kotlin support

The project location cannot be at the filesystem root.

Figure 2. This check box enables C++ support.

Now with added C++

Anyone with in-depth knowledge of embedded Java systems such as MicroEJ will instinctively suspect that interworking using the JNI will not be entirely free of hurdles. Behind the acronym JNI hides the 'Java native interface', a system that allows native methods to be called from within programs running on the Java virtual machine. It goes without saying that passing parameters between Java code and native code is anything but speedy. Google therefore also offers a C++ API which in theory will circumvent this problem. Let us see if we can get code to output a square wave up and running.

Facilities for C++ development do not come as part of the standard distribution. In Android Studio open the installation assistant that you can find under Tools - SDK Manager, and select SDK Tools. Check the NDK entry and then download the components (about 1 GB in total).

Android Studio allows C++ support to be added into projects that already exist; but, in particular for applications that only involve a small amount of code, it is not worth the trouble to go through this process and it is easier to start a new project. When doing this, make sure the 'Include C++ support' check box is ticked. Android Studio will then use a project template that is suitable for use with the C++ compiler: see **Figure 2**.

One difference between a C++ project and its normal brethren is that an extra step is involved when using the assistant. Under C++ standard the setting Toolchain default should be selected in order to ensure that the correct compiler is used and to avoid further problems in the course of setting up the project. After clicking on Finish Android Studio will create a new project in the usual way, but with a native component alongside the familiar Java components.

Firefox still not welcome here!

If you encounter strange anomalies when working with the Android Things console, it is a good idea to check which browser you are using. Google's system continues to have problems with Firefox, Edge and Safari, and from experience the most reliable choice is their own Chrome browser.

Newly-downloaded NDK installations are often not complete. Before continuing it is best to resynchronize and/or recompile the application. When Gradle detects that a component is missing, the error message that it generates includes the option to download it. In most cases it suffices simply to click on the link in the error message: then, assuming you have an Internet connection, Android Studio will handle the rest.

Since at the time of writing the C API for Android Things has not been finalized, it does not form part of the NDK. Instead, you can open the URL [5] in a browser and download the most recent release. At the time of writing the download file has the name *native-libandroidthings-1.0.zip*. Inside this archive is a directory called *native-libandroidthings-1.0* which contains the compiled Android Things API.

This must be moved into the directory that contains the file *CMakeLists.txt*. This is a configuration file that controls the compilation process for the native part of the project. On the author's machine the project is called *ElektorCGPIO*, and the corresponding path is therefore *ElektorCGPIO/app*.

In order to update Android Studio's view of the project you should restart it at this point: the IDE is not always on the ball when it comes to detecting changes made externally to the file system. The next step is to click on *External Build Files - CMakeLists.txt* in order to load the makefile for the native component.

Just to clarify: all but the smallest C and C++ projects are no longer compiled manually, but rather with the help of a makefile. This is a kind of control file, which the eponymous build command make uses to determine the steps necessary to compile the project.

Some changes to the project structure are required in order to link in our library. First, below the line `cmake_minimum_required(VERSION 3.4.1)` we must add a stanza specified by Google: make sure that you change the path in the lines below to reflect your configuration.

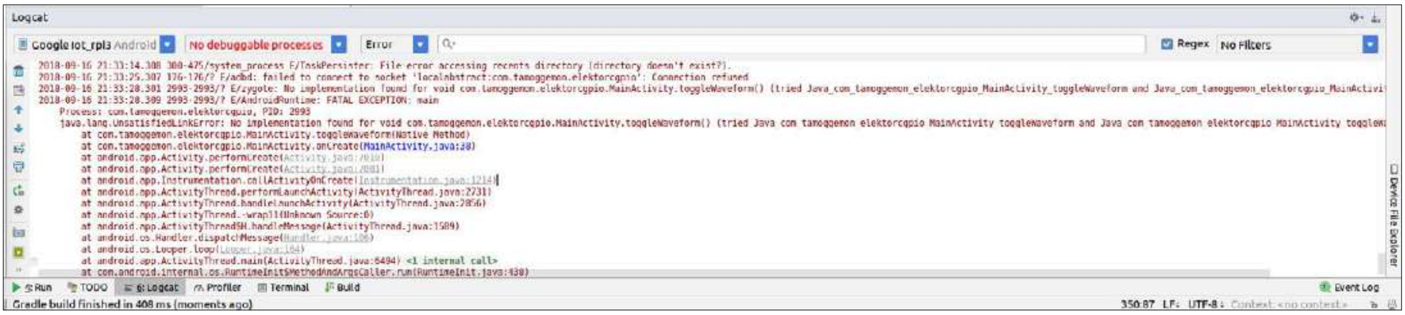
```
set(CMAKE_MODULE_PATH $ /home/tamhan/Desktop/
stuff/2018September/ElektorATINews/ElektorCGPIO/app/
native-libandroidthings-1.0)
find_package(AndroidThings REQUIRED)
include_directories($)
```

Although this snippet may seem baffling at first glance, its function is just to add the directory where the new components that have just been downloaded and extracted to the module load path. During the program compilation process *CMake* looks in various directories for native libraries which are linked into the code as required.

A further include file can also be found there, which contains information relating to how the library is linked in during the rest of the compilation process.

Further below there is a structure that lists the libraries to be loaded. Here again a modification is required for the development environment to work correctly.

```
target_link_libraries(
        native-lib
        $
        $)
```



```
2018-09-16 21:33:14.308 308-475/system_process E/TaskPersister: File error accessing recents directory (directory doesn't exist?).
2018-09-16 21:33:25.307 176-176/? E/adbd: failed to connect to socket 'localabstract:com.tamoggemon.elektorcgpio': Connection refused
2018-09-16 21:33:28.301 2993-2993/? E/zygote: No implementation found for void com.tamoggemon.elektorcgpio.MainActivity.toggleWaveform() (tried Java_com_tamoggemon_elektorcgpio_MainActivity_toggleWaveform and Java_com_tamoggemon_elektorcgpio_MainActivi
2018-09-16 21:33:28.309 2993-2993/? E/AndroidRuntime: FATAL EXCEPTION: main
    Process: com.tamoggemon.elektorcgpio, PID: 2993
    java.lang.UnsatisfiedLinkError: No implementation found for void com.tamoggemon.elektorcgpio.MainActivity.toggleWaveform() (tried Java com tamoggemon elektorcgpio MainActivity toggleWaveform and Java com tamoggemon elektorcgpio MainActivity togglew
        at com.tamoggemon.elektorcgpio.MainActivity.toggleWaveform(Native Method)
        at com.tamoggemon.elektorcgpio.MainActivity.onCreate(MainActivity.java:38)
        at android.app.Activity.performCreate(Activity.java:7010)
        at android.app.Activity.performCreate(Activity.java:7001)
        at android.app.Instrumentation.callActivityOnCreate(Instrumentation.java:1214)
        at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:2731)
        at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:2856)
        at android.app.ActivityThread.-wrap11(Unknown Source:0)
        at android.app.ActivityThread$H.handleMessage(ActivityThread.java:1589)
        at android.os.Handler.dispatchMessage(Handler.java:106)
        at android.os.Looper.loop(Looper.java:164)
        at android.app.ActivityThread.main(ActivityThread.java:6494) <1 internal call>
        at com.android.internal.os.RuntimeInit$MethodAndArgsCaller.run(RuntimeInit.java:438)
```

Gradle build finished in 408 ms (moments ago)

350:87 LF: UTF-8: Context: <no context>

Figure 3. Android Studio gives the name of the missing function.

Configuration details for the native part of an Android Studio project are not only stored in *CMakeLists*. Open the file *build.gradle* in the *App* module and insert an NDK stanza into the Android block.

```
android {
    compileSdkVersion 27
    defaultConfig {
        ndk {
            abiFilters 'armeabi-v7a', 'x86'
        }
        . . .
```

Being based on Java, Android Studio and Android Things are independent of CPU architecture at the application level. In the case of our API we are dealing with a precompiled module, which is therefore in the form of assembly code. We are thus restricted to operation on the processor architectures that can be targeted, namely ARM and x86. Google no longer sees value in supporting other processor architectures such as MIPS.
Changes are also required in the *MainActivity* to support working with native libraries. To begin with, we now need a `static` constructor.

```
public class MainActivity extends Activity {
    static {
        System.loadLibrary("native-lib");
    }
```

Static constructors are a special case of normal constructors, which are called when a program starts up. Here we employ the function `LoadLibrary` in order to establish connection with the C-code library at run time.
The reference to ‘at run time’ is important: the compiler does not carry out any verification of a combined C- and Java-based project at build time to check whether the methods required by the Java code are actually implemented in the C++ code. For this reason it is absolutely essential to test JNI-based code as thoroughly as possible, in order to avoid annoying run-time errors.
The call to a native method proceeds as follows.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    . . .
    TextView tv = (TextView) findViewById(R.
id.sample_text);
    tv.setText(stringFromJNI());
}
public native String stringFromJNI();
```

The important thing to note is that a native method must be declared in Java using the native keyword. The line tells the compiler that the developer undertakes to provide the method; if the block is missing, then an error will be reported at compile time.

A question of nomenclature

As programming languages, Java and C could hardly be more different. While Java uses memory management and offers complex abstraction classes for handling strings and the like, the C programmer is sitting right next to the hardware.
Unfortunately for reasons of space we cannot provide a complete introduction to JNI in this article. Interested readers are referred to the tutorial at [6].
In practice the main problem developers will face is the compatibility (or otherwise) of data types. Finding the correct syntax can be very trying: one example of this is the extent of the convolutions required to work out the correct name to use in the C code for the `stringFromJNI` method we called from Java above.

```
extern "C" JNIEXPORT jstring JNICALL
Java_com_tamoggemon_elektorcgpio_MainActivity_
stringFromJNI(
        JNIEnv *env,
        jobject /* this */) {
    std::string hello = "Hello from C++";
    return env->NewStringUTF(hello.c_str());
}
```

As well as the miscellany of flags that indicate that the method is to be exported, we also see the string *Java_com_tamoggemon_elektorcgpio_MainActivity_stringFromJNI*: this is an identifier generated automatically by the Java compiler that is used to name the method.

When working with Android Studio there is a neat trick to help find the correct names for methods. Open the file *MainActivity.java* and modify it as follows.

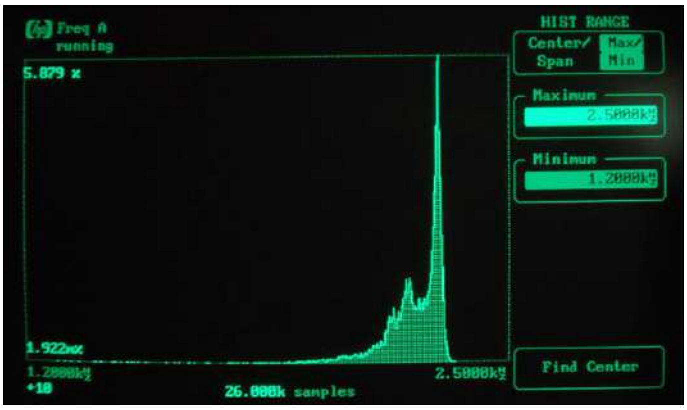


Figure 4. Even when rewritten in C++ the program is not significantly quicker.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    toggleWaveform();
}
public native void toggleWaveform();
```

The above code will call the method *toggleWaveform*. Now simple logic dictates that because this method has not yet been written, when the code is run on the Raspberry Pi 3 it will report an error. Nevertheless, click on Play to instruct the system to upload the program over the ADB.

The program will (at most) briefly flash up on the screen before you are dropped back into the Android Things launcher. In Android Studio switch to the LogCat tab and set the filter in the combo box to Error: this will cause Android Studio to filter out all messages other than those that report critical problems. The screen should now look like the image in **Figure 3**.

The information we have thus collected lets us create a basic framework for the method we need to write as follows.

```
#include <jni.h>
#include <string>
#include <pio/gpio.h>
#include <pio/peripheral_manager_client.h>
extern "C" JNIEXPORT void JNICALL
Java_com_tamoggemon_elektorcgpio_MainActivity_
toggleWaveform(
        JNIEnv *env,
        jobject /* this */) {
}
```

LogCat

A running Android device continuously emits messages that are logged by the operating system. In Android Studio, as well as in many other development tools, there is a window labelled ‘LogCat’ which allows the contents of the log files to be inspected.

The `JNIEnv` and `jobject` parameters are always present in JNI methods, and they can be ignored in this case. In more complex situations they can be used to allow access to the JNI helper object, which in turn allows you to access the services of the Java runtime; and the JNI can, for example, return objects that let your code can interact with Java objects and variables. The two include files starting ‘pio’ are required to allow access to the Android Things API.

In the next step we can look at the method that actually toggles the output pin. Its code is as follows.

```
extern "C" JNIEXPORT void JNICALL
Java_com_tamoggemon_elektorcgpio_MainActivity_
toggleWaveform(
        JNIEnv *env,
        jobject /* this */) {
    APeripheralManagerClient* client =
APeripheralManagerClient_new();
    AGpio* gpio;
    APeripheralManagerClient_openGpio(client, "BCM2",
&gpio);
    AGpio_setDirection(gpio,
AGPIO_DIRECTION_OUT_INITIALLY_LOW);
    while(1==1)
    {
        AGpio_setValue(gpio, true);
        AGpio_setValue(gpio, false);
    }
}
```

In principle there should be nothing too surprising here. We are using the C++ API described in detail at [7] to construct a GPIO pin object and then output a characteristic waveform. In theory our program should at this point be ready to try out. In practice, unfortunately, it grinds to a shuddering halt with an error message. The cause of this is that we have already installed a program on the Raspberry Pi that uses port pin BCM2 and which therefore blocks our program’s access to it.

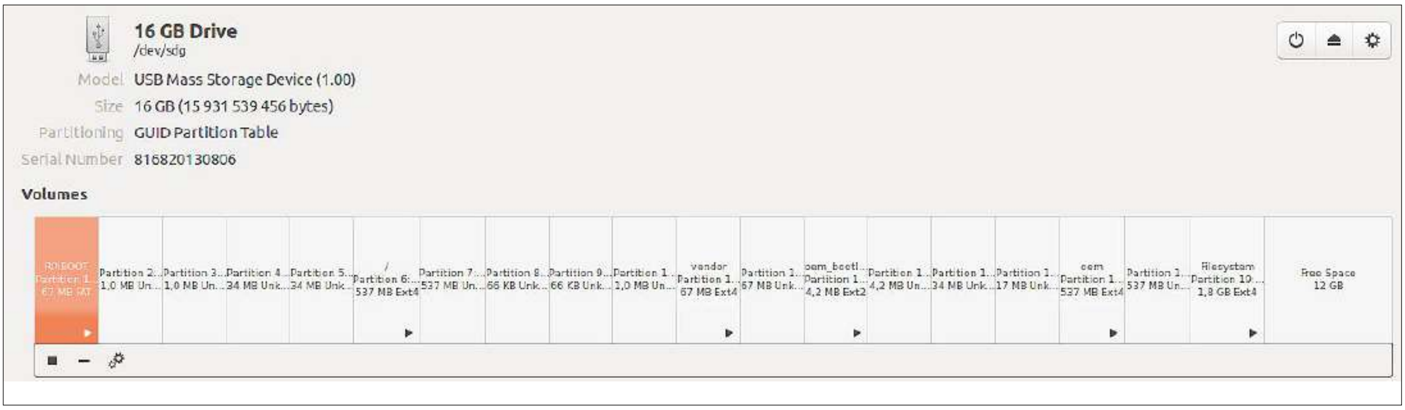
The simplest route to solving this problem is to connect a mouse to the processor board and under System tab carry out a system reset. A slightly more delicate alternative is to log on to the system using the ADB and remove the program from the Raspberry Pi.

Now our program can run successfully and we can look at the characteristics of the signal it generates in detail.

The plot on the modulation domain analyzer shown in **Figure 4** indicates that the improvement that we have achieved is at best marginal.

If you delve deeper into the Android Things API (or into one of the many gripes raised in the developer community) you will soon discover that access to the hardware, even under C++, is mediated by the Linux sysfs pseudo-filesystem. This delivers a great deal of security in interacting with hardware, but comes at a cost in performance. In comparison, APIs like *WiringPi* write directly to registers in the microcontroller. Such an approach is not possible under Android Things, as our program does not have superuser permissions.

Performance can be improved by dealing directly with the *sysfs* filesystem. However, the improvement is not great, and in practice is not worth the additional effort involved.



16 GB Drive
/dev/sdg

Model USB Mass Storage Device (1.00)
Size 16 GB (15 931 539 456 bytes)
Partitioning GUID Partition Table
Serial Number 816820130806

Volumes

RPIBOOT Partition 1 67 MB FAT
Partition 2... 1,0 MB Un...
Partition 3... 1,0 MB Un...
Partition 4... 34 MB Unk...
Partition 5... 34 MB Unk...
/ Partition 6:... 537 MB Ext4
Partition 7:... 537 MB Un...
Partition 8... 66 KB Unk...
Partition 9... 66 KB Unk...
Partition 1... 1,0 MB Un...
vendor Partition 1... 67 MB Ext4
Partition 1... 67 MB Unk...
oem_bootl... Partition 1... 4,2 MB Ext2
Partition 1... 4,2 MB Un...
Partition 1... 34 MB Unk...
Partition 1... 17 MB Unk...
oem Partition 1... 537 MB Ext4
Partition 1... 537 MB Un...
Filesystem Partition 10... 1,8 GB Ext4
Free Space 12 GB

Figure 5. Practically all the partitions are duplicated, which makes the update process safer.

Automatic updates

These days it is practically impossible to subscribe to an IT security mailing list and not be inundated with news of hacks compromising IoT devices. Compounding the problem is the fact that users seldom update their devices.

Google is trying to do something about this with the Android Things Console. The idea is to make it as straightforward as possible for developers to offer regular updates to their code. Since this functionality can be very useful in commercial situations, we will take a moment to describe how the new update system works.

The basic idea behind ‘A-B’ updates is illustrated in **Figure 5**. On the SD card practically all the partitions are duplicated. An incoming update is written to one partition of each duplicate pair, allowing the processor to carry on safely as normal using only the other partitions. If something should go wrong while receiving an update Android Things will automatically simply continue to use the working version, so avoiding the ‘bricking’ of the device.

An Android Things device is subscribed to a ‘channel’ for updates. Google declares four standard channels, and new devices start off subscribed to the stable-channel. Changing the channel entails a factory reset, which deletes all user data.

The new features in Android Things 1.0 include a comprehensive API that allows the developer to push updates. Among other things it is possible to specify when and how updates will be installed. Although at first sight this might seem unnecessarily finicky, it is an important feature: for example, it allows you to prevent an update interrupting a long-running sequence of measurements.

For reasons of space we will not go further into the update API here. A practical implementation can be found at [8].

Conclusion

The small changes in the GPIO API in Android Things 1.0 will not necessarily be welcomed as good news by developers: it would have been nice if existing code could have been reused more easily.

With these changes Google continues the trend that was already seen in previous versions. Getting the maximum performance out of a system is not so important to the ‘Big G’, especially if, as in the case of allowing direct access to hardware, this comes with compromises in system security.

The operating system provides a comfortable ‘playground’ environment for anyone who wants to build user interfaces for devices with a minimum of effort. If what you are looking for is a real-time operating system, then even this most recent version of Android Things is probably not for you. ◀

180305-02

Web Links

[1] Android on your RPi (1), Elektor 2/2017: www.elektormagazine.com/160361

[2] Android on your RPi (2), Elektor 3/2017: www.elektormagazine.com/160369

[3] Android Things management console: https://partner.android.com/things/console

[4] Android Things console documentation: https://developer.android.com/things/console

[5] C API: https://github.com/androidthings/native-libandroidthings/releases

[6] JNI tutorial: https://developer.android.com/ndk/guides/concepts

[7] Description of the C++ API: https://developer.android.com/things/sdk/pio/native

[8] Update API: https://developer.android.com/things/sdk/apis/update

Automatic Traffic Light Controller

for model railways

By **Piet Kralt** (Netherlands)

Model railway layouts often have roads as well as tracks. Placing traffic lights at road junctions is a nice touch, especially if they actually work. That makes everything look a bit more realistic. In this article we get you started and show you how to obtain the desired result with simple means.

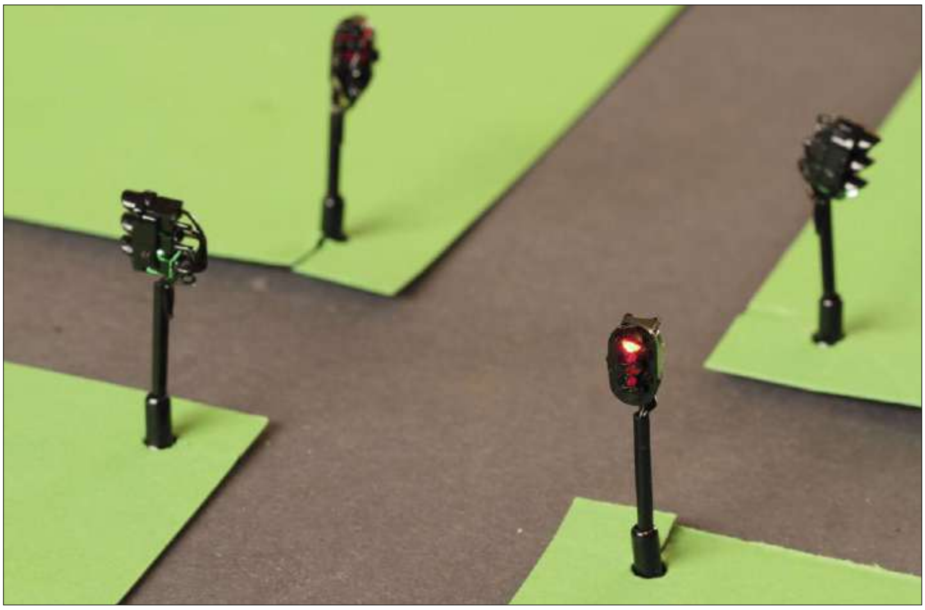


Figure 1. With the N-gauge version the three LEDs have separate cathode leads and a common anode lead.

Making your own traffic lights is not trivial, but you can also buy them ready-made. Figure 1 shows a pole-mounted traffic light for an N-gauge layout. With this version the three LEDs have separate cathode leads and a common anode lead. There is a 1 kΩ series resistor in the anode lead to limit the current. The LEDs are intended to operate from a supply voltage of 12 to 15 V. A simple calculation ($I = V/R$) shows that the resulting current through the LEDs is about 10 to 13 mA.

The hardware

There are many different ways to design circuitry to control these LEDs, but to keep things simple (at least as far as the hardware is concerned) the author opted for software control using a PIC16F526 microcontroller. With a maximum rated output current of 25 mA, each pin can drive two LEDs, but only at a voltage of 5 V. This requires some modification of the traffic light. The 1-kΩ resistor has to be removed and replaced by a 270-Ω or 330-Ω resistor in each of the cathode leads. See the schematic diagram in **Figure 2**. This circuit can control the four traffic lights of a normal intersection, with the LEDs of opposing traffic lights driven by a single output pin for each color.

Model railways usually have a 12 V supply voltage but rarely have 5 V available, so there is a 5 V voltage regulator in the circuit. There is also a connector for the external power source and a connector for in-circuit programming of the microcontroller. The jumper and the pushbutton are intended for possible functional enhancements.

An external supply voltage of 8 to 16 V is applied to K1, and the 7805 (IC1) and associated capacitors convert it into 5 V. Connector SV1 is the programming connector for in-circuit programming of the PIC microcontroller. If you prefer to use a separate programmer, you can omit this connector. SV3 and SV4 are the connectors for the four traffic lights. Each LED has its own terminal.

The software

To keep the hardware simple, most of the functionality is implemented in software. In the author’s opinion, that is a major advantage because software is a lot easier to modify than hardware.

The author first wrote a very straightforward control program in assembler language. It can be downloaded at [1]. The cycle times are hard-coded, but they can be changed in the .asm file if so desired. SV2 gives you the option of normal red → green → yellow → red or yellow blinking. The circuit is so simple that it can easily be built on a breadboard for a quick test.

Prototype PCB

The CAD files for the PCB layout and the Eagle design files are contained in the file 150710-11.zip available at [1]. The board has intentionally been kept reasonably small. It could be even smaller, especially if you use SMD components, but the gain is only marginal. The space

savings are hardly worth the trouble of fiddling with SMDs.
The fully assembled prototype is shown in **Figure 3**. The components are shifted slightly on the final version of the board, mainly to accommodate the box header. A normal 2x5 pin header would be easier and cheaper, but with the obvious risk of connecting the plug the wrong way.

As previously mentioned, each lead of the traffic lights has its own terminal. The connector pinout is shown in **Table 1**. Traffic lights 2 and 4 are connected to SV4 in the same way. In the prototype the leads were soldered directly to the PCB, mostly on the back side, mainly for the sake of convenience. The author originally intended to use PCB-mount screw terminal strips, but that turned out to be impractical because the wires are so thin.

Concluding remarks

With this design the author managed to build a satisfactory working prototype which meets the main objective. However, there are plenty of possibilities for further enhancements, such as:

- a version with heavy-duty drivers to allow larger LEDs or even incandescent lamps to be used;
- the option of configuring the lights for various sequences used in other countries, such as red → red/yellow → green → yellow → red;
- the capability to configure different green and yellow times (that’s why the pushbutton is included in the circuit).

In short, give your imagination free rein and modify the software (and the hardware) as you wish. ⏮

(150710-I)

Photo credits: Dirk Jan Kralt

Table 1. Connector pinout for the LEDs

SV3 pin	Traffic light	Lead/ color
1	1	common 1
3	1	green
5	1	yellow
7	1	red
2	3	common 3
4	3	green
6	3	yellow
8	3	red

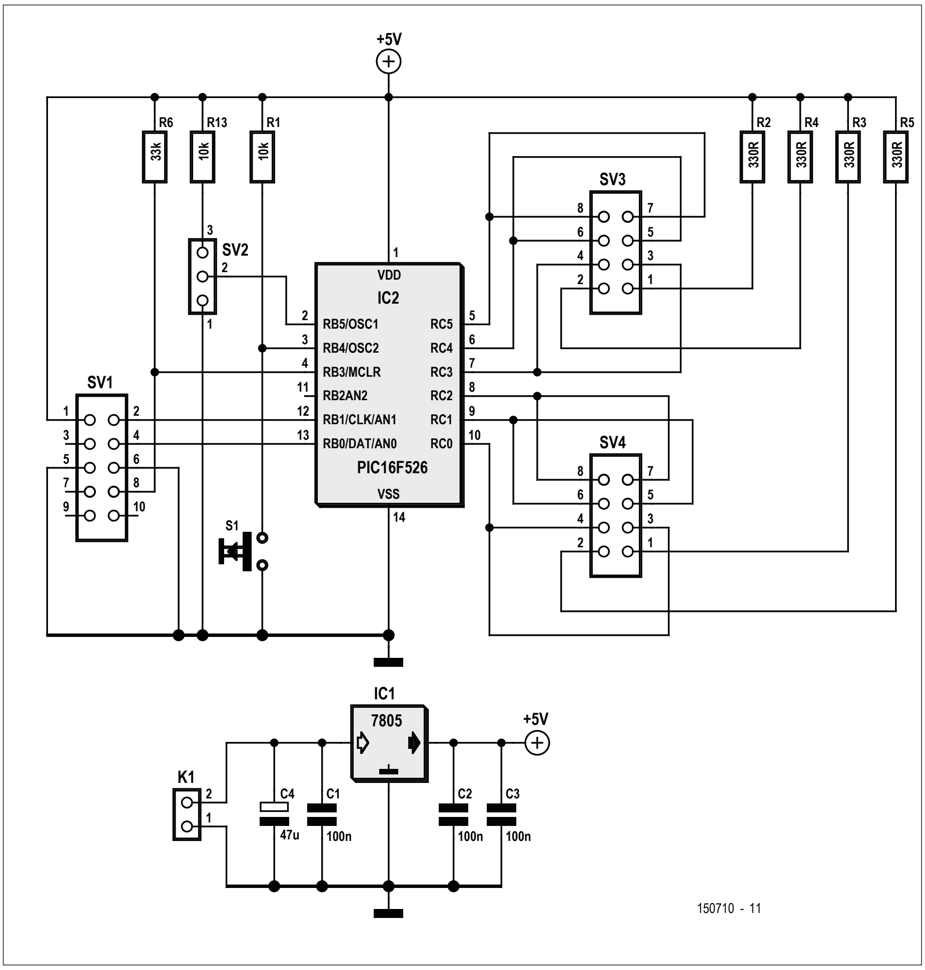


Figure 2. The schematic diagram shows the basic idea of the circuit.

Web Links

[1] Software and CAD downloads: www.elektormagazine.com/150710

[2] Project page: www.elektormagazine.com/labs/stoplichtautomaat-voor-de-modelbaan

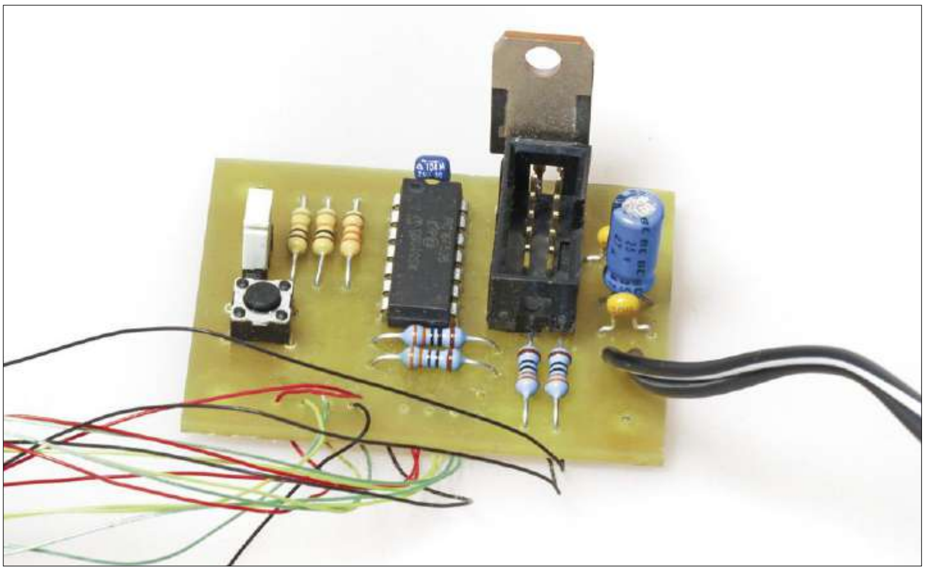
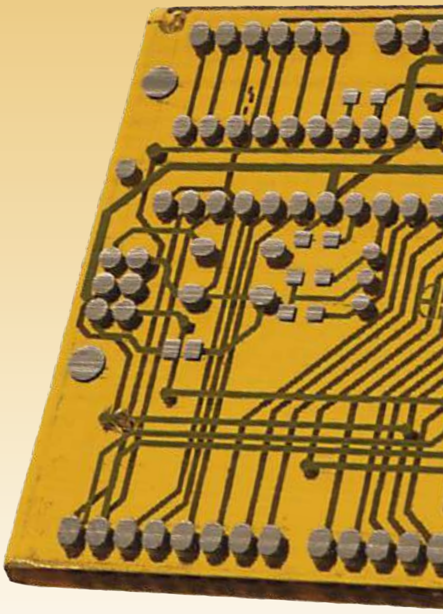


Figure 3. In the prototype the leads are soldered directly to the PCB.

Making PCBs with your 3D printer

Creative solutions using rubber, marker pens and lasers



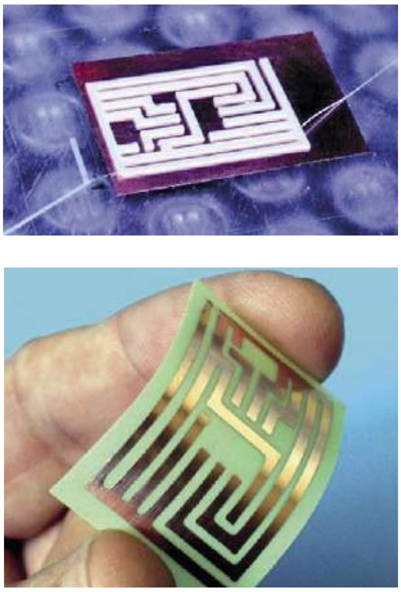
When you have a 3D printer at home, you obviously want to use it as much as possible. Some people have come up with innovative solutions that let them 'print' PCBs using their 3D printer. This usually involves a secondary process, but the resulting boards are useable and you also save time and money compared to using an online PCB manufacturer.

By **Harry Baggen** (Elektor Labs)

There are several methods you can use to make your own PCBs at home. The one that first comes to mind is where the board is covered with a photosensitive layer. This is then covered with the track layout, exposed, developed and finally etched. Another method is to copy the track layout onto an overhead transparency sheet using a laser printer, then use heat to transfer the toner from the sheet onto the board. An alternative is to use a PCB milling machine to create isolating channels onto the board. Unfortunately, these machines aren't cheap and are not of much interest to the enthusiast. More and more technophiles have obtained a 3D printer so they can print their own objects. When you also have an interest in electronics and design your own PCBs, there comes a point where you wonder if a 3D printer can be used for this. It certainly can, and several methods are described on the Internet.

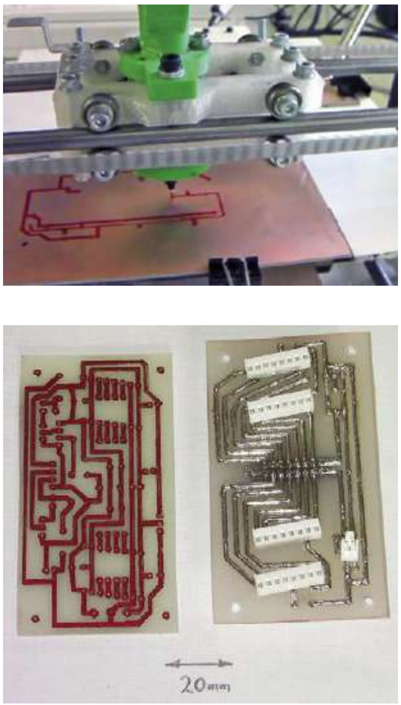
Using rubber filament

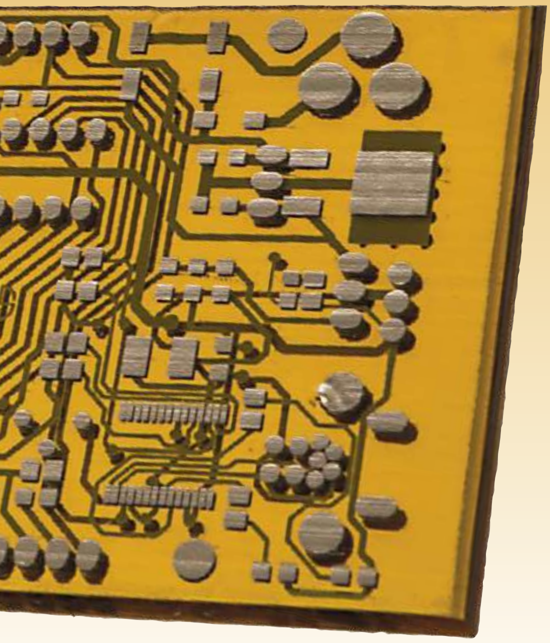
At first sight, it makes sense to simply print a single filament layer of the track layout directly onto the copper side of a blank board. The board would then be etched, leaving the copper underneath the filament. During experiments carried out by mikey77, a very active contributor on the well-known Instructables website [1], he discovered that this method didn't work very well because the usual PLA and ABS materials wouldn't adhere properly to the copper layer. Instead, he tried using a rubber filament, called Ninjaflex, that adheres well to virtually any material. The print board was fixed to the print bed of his Makerbot Replicator 2 using some glue from an aerosol can. With this method he could even create flexible prints by using very thin 'boards'.



Using a marker pen

Most other methods for making PCBs using a 3D printer use a modified print head in which a marker pen has been mounted. On the Plotting pages of RepRap [2] they describe how you can use this method to draw the tracks onto the board. In this instance, a fine-tipped permanent marker pen is used to draw the layout onto a blank board. The description is very detailed and they state which programs are required to convert the layout files into GCode, which can be processed by a RepRap or other printer. When the layout has been printed, the board can be etched in the usual way in ferric chloride, for example.



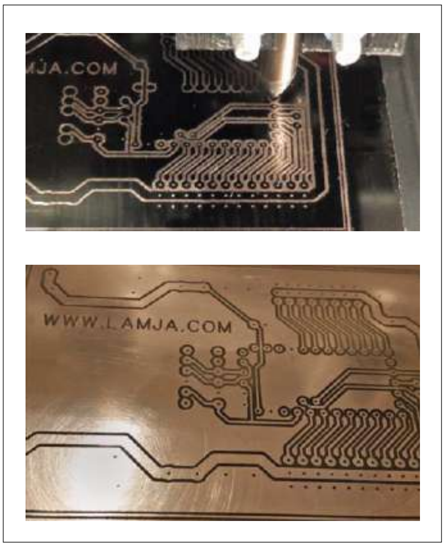


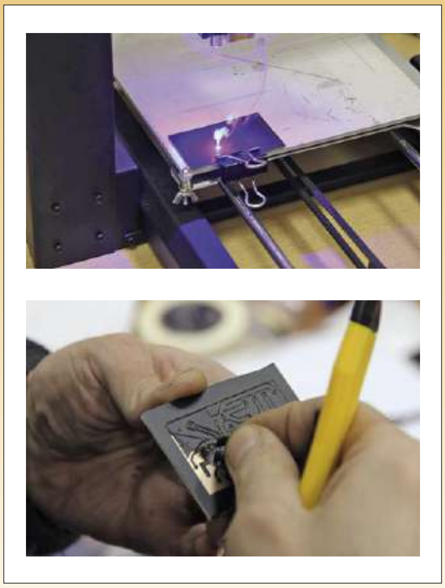
This method is only suitable when thicker tracks and pads are used, since the marker pen cannot be used to draw fine lines accurately. The article also has a warning about this limitation. However, it is perfectly suitable for a standard PCB where only ordinary through-hole components are used. You can of course make the pen holder yourself, and several designs are described at Thingiverse, such as the Elastico Pen Holder [3].

The reverse marker method

It's also possible the opposite way! By this we mean that the board is first completely covered with ink from a broad-tipped permanent marker pen. The parts that have to be etched are then scratched off using a steel pen with a sharp point, which has been mounted in a custom holder on the 3D printer. This method has been described by Arvid Mortensen on his Lamja website [4]. He uses this method to create only the isolating channels, and leaves the rest of the unused copper alone, the same way as with a PCB milling machine, but this obviously won't affect the operation of the PCB. The PCBs shown in the photos look very good, with clear tracks and a resolution that seems better than that obtained with the ordinary marker pen method.

The files needed to make the holder for the metal pen have been made available by Arvin. This holder has been designed for use with the K8200 printer made by Velleman.





Using a laser

It's won't come as a surprise to hear that some people have tried using a laser. However, in order to burn away the copper layer you would need a powerful laser and we haven't found any experiments in this area. But there are some alternative methods, where a laser with a power of a few Watts is sufficient to 'burn' the layout onto the board [5]. The laser used here is a semi-conductor device with a power of 2.1 watts, which is fairly easily obtainable. It is mounted in place of the extruder of the 3D printer. The PCB is covered with a self-adhesive vinyl film and then mounted onto the print bed of the 3D printer. The printer is then 'fed' with the inverted version of the print layout. This causes the laser to burn isolating channels in the vinyl material. Unfortunately, when PVC is burnt, harmful dioxins can be released, which means that good ventilation is essential. In view of this, it would be better to use a different material for the adhesive film.

> Of course, some people couldn't resist using a laser

The vinyl film on the isolating channels must then be removed manually, leaving the copper layout. The prepared board can then be etched as usual (now we've mentioned it, this also requires good ventilation). The quality of boards made this way seems quite reasonable.

So you can see that with some creativity and a few modifications you can use a 3D printer for a totally different purpose. ⏮

(160191)

Web Links

[1] https://3dprint.com/11367/3d-print-copper-circuit-board/

[2] http://reprap.org/wiki/Plotting

[3] www.thingiverse.com/thing:671711

[4] www.lamja.com/?p=635

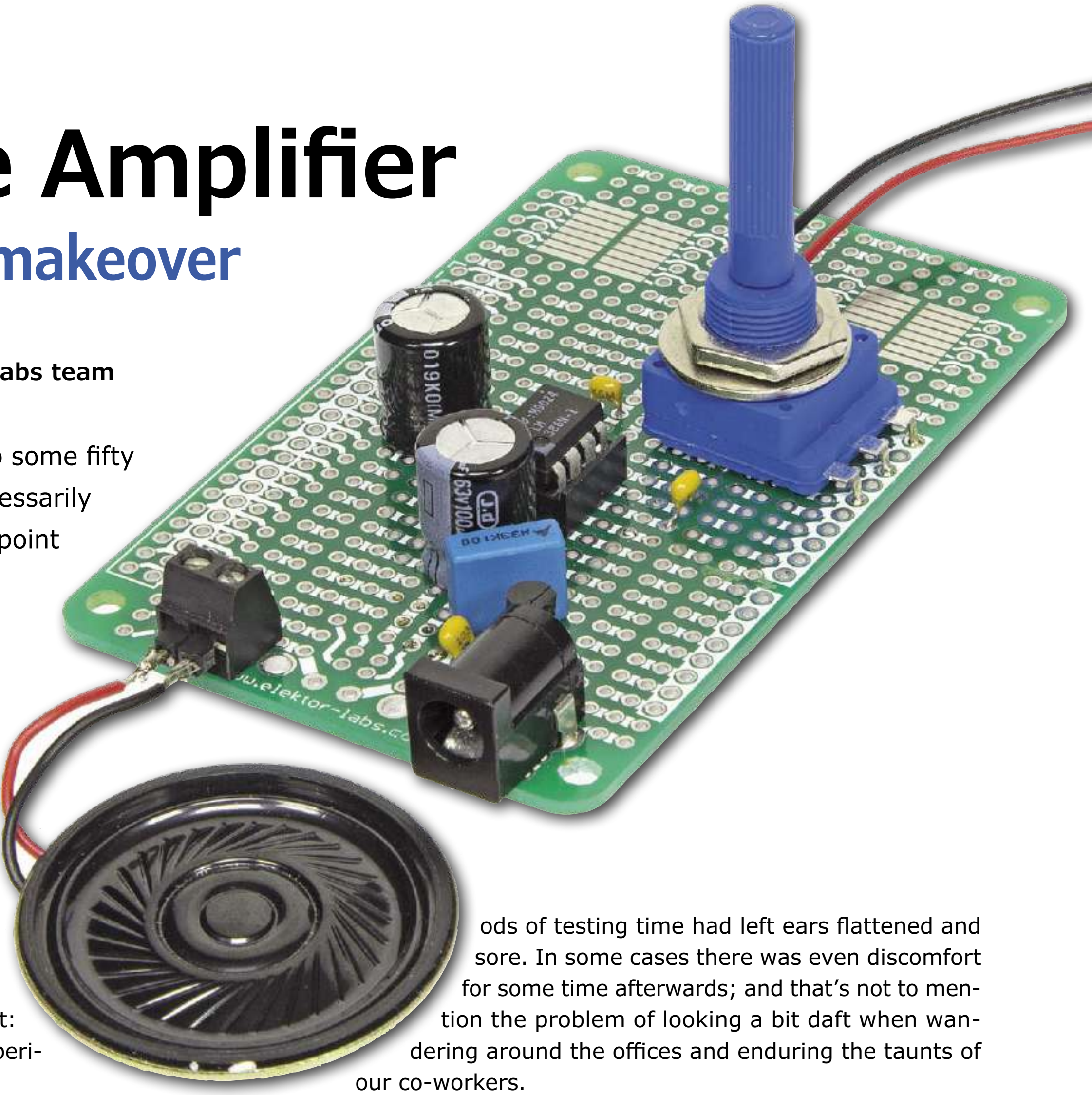
[5] http://3dprintingindustry.com/news/z-present-diode-laser-etching-pcb-82860/

My Little Amplifier

LM386 gets a makeover

By the tolerant & considerate **Elektor Labs team**

Even an IC that has clocked up some fifty years on the market is not necessarily hopelessly obsolete. A case in point is that several semiconductor companies are still making the LM386 power operational amplifier, and even now are bringing out new and improved versions.



We were working in the lab late one night (on projects like the 'talking sonar' and 'tiny radio') when our eyes beheld an unfortunate sight: wearing headphones for extended periods of testing time had left ears flattened and sore. In some cases there was even discomfort for some time afterwards; and that's not to mention the problem of looking a bit daft when wandering around the offices and enduring the taunts of our co-workers.

Not ones to wallow in self-pity, we decided that the best solution would be to make a small and simple audio amplifier. Not some vast bass bin or complex PA system, but perhaps something just big enough to gently get on the nerves of our oh-so-delightful and thoughtful colleagues. With this noble motive in mind we had a look in the component cupboard and chanced upon (you guessed it) our old friend the LM386.

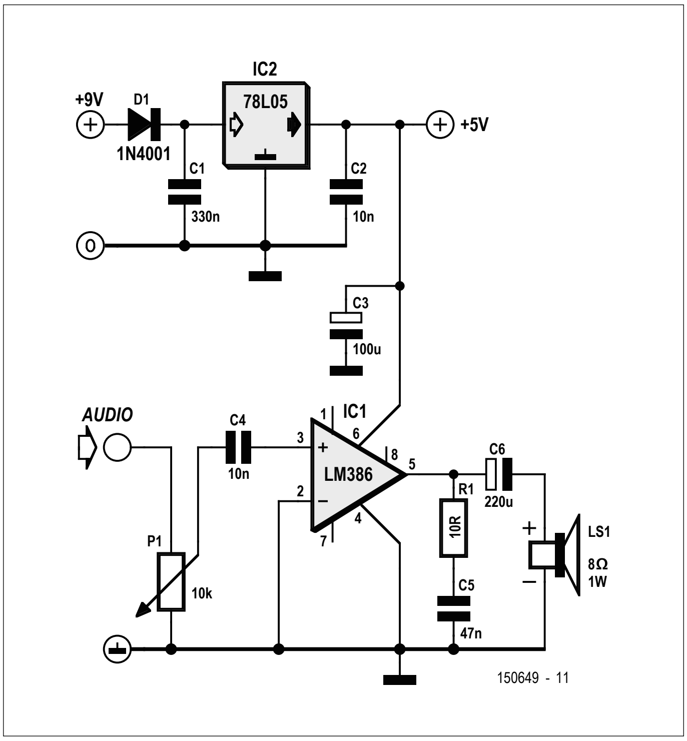


Figure 1. The LM386 power opamp is an old friend.

Old wine

This integrated audio amplifier designed for low-voltage operation has been scuttling around on its eight legs for nearly fifty years. It was manufactured by National Semiconductor, along with its sister opamps the LM741 and LM324.

The LM386 was used in a huge number of battery-operated appliances, from portable radios and guitar amplifiers to hobby projects. It could approach a watt or so of output power without complaint. Although the device dates back to the early days of integrated circuit technology, it bears its age well and today there are versions on the market produced by several different manufacturers. More recently remixed editions have appeared with operating voltages of up to 12 V or 18 V, with output powers of 0.25 W to 1 W, depending on the device variant, load impedance, and acceptable level of THD.

The LM386 is certainly a versatile chip. It only needs a small number of external components, just the odd resistor and capacitor, to make a complete audio operational amplifier. There are various tricks that can be used to adjust the gain of the amplifier or to improve the bass response. The LM386 also turns up unexpectedly in other audio circuits: for example it can be used as an oscillator generating sinewaves or squarewaves.
Basically operational amplifiers have a simple job: take a small input voltage and generate an output voltage which is higher by a factor of 10, 100 or even 1000 than the input. In this kind of circuit the LM386 can take a run-of-the-mill audio input signal and amplify it by a factor of something between 20 and 200: in the trade this amplification factor is called ‘voltage gain’.
The most important connections on the device are of course the two audio signal inputs, which are on pins 2 and 3 (the audio input to the LM386 is differential rather than being referenced to the ground connection), and the audio output, which appears on pin 8. Also important are the power supply connections, pin 4 for ground (relative to which the audio output signal is referenced) and pin 6, to which, as mentioned above, up to 18 V can be applied, depending on the variant of the device in use. Less important are the pins marked ‘gain’, pin 1 and pin 8. These two pins are connected internally via a resistance of 1.35 kΩ. If a 10 µF capacitor is connected between these two pins (that is, in parallel with the internal resistance) the voltage gain will be set to 200. If now a potentiometer is connected in series with this capacitor, the voltage gain can be freely adjusted between 20 and 200. It is also possible to dispense with the capacitor, as we have done here. Not because we are particularly tight-fisted, but rather because we were more than satisfied with the default voltage gain of 20.

New wineskin

As can readily be seen from **Figure 1**, the circuit includes just the LM386 and a (very small) handful of passive components. We power the LM386 from a single (non-symmetric) supply rail, and as a consequence of this we need to provide AC coupling between the input signal and the input to the device, as well as at the output of the device: this is the job of capacitors C4 and C6. Capacitor C4 takes the input audio signal to pin 3 of the device, and the other signal input, on pin 2, is connected to ground.
At the output of the device we can see a somewhat odd-looking combination of components in R1 and C5. This is a snubber network, or, to use the elegant French expression, a Boucherot cell. Its job is to damp high-frequency oscillations that can be caused as a result of the inductance of the coil in the loudspeaker.
Our tiny amplifier runs from a supply voltage of +5 V. IC2 is a garden-variety 78L05, and this ensures that the power supply is well regulated. At its input it can accept an unregulated voltage of anything from +7 V to (at least according to the datasheet) +35 V; however, it might be a good idea to keep the input voltage to at most +12 V. We even offer the user the luxury of being able to connect the input power the wrong way around: then the circuit will not work, but diode D1 will at least prevent any damage to it.

We now finally come to our new wineskin: the modern ELPB-NG Elektor prototyping board, on which the LM386 and its surrounding components can comfortably be accommodated. **Figure 2** suggests one way that the components can be mounted on the board. The red and black wires that come from the board carry the audio signal to the circuit’s input, while the supply voltage is provided over the power jack (the black lump to the bottom left). To connect the 8-Ω loudspeaker to the output we splashed out on a proper terminal block.

And that’s it built: we now await with eager anticipation the results of testing the circuit in our harsh laboratory environment! ◀

(150649)

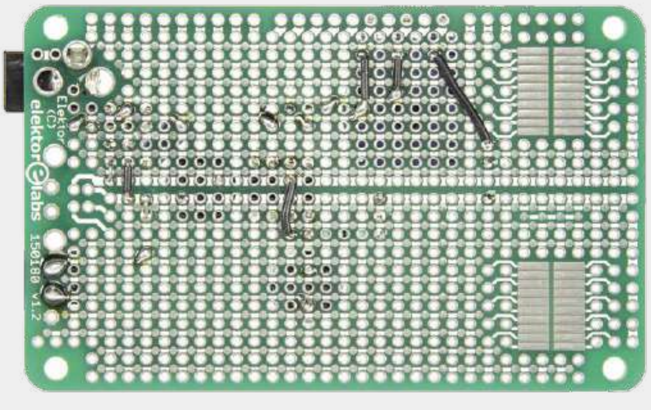
COMPONENT LIST

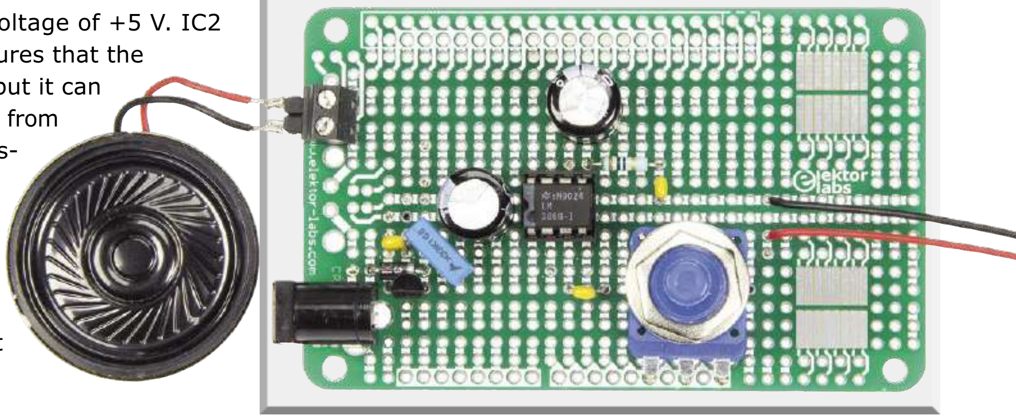
Resistors
R1 = 10Ω
P1 = 10kΩ log. law potentiometer

Capacitors
C1 = 330nF, 5mm pitch
C2,C4 = 10nF, 0.1” pitch
C3 = 100µF 50V, 3.5mm pitch, 8mm diam.
C5 = 47n, 0.2” pitch
C6 = 220µF 50V, 3.5mm pitch, 8mm diam.

Semiconductors
D1 = 1N4001
IC1 = 7805
IC2 = LM386

Miscellaneous
ST1 = PCB screw terminal block, 5mm pitch
DC adaptor socket
DIP-8 IC socket
Loudspeaker, 1W, 8Ω
PCB type ELPB-NG, Elektor Store # 150180-1





(Nearly) Everything you always wanted to know about...

Enclosures

Photo: Bopla Gehäuse Systeme GmbH

By **Thijs Beckers** (Elektor Netherlands)

When it comes to an enclosure for your PCB, it is always best to choose one in advance, and then design the PCB accordingly. That way you can ensure that everything will fit. This article highlights several things you should consider when choosing an enclosure, helping you pick the right one for the job.

When you're involved in electronics you should always keep in mind that electricity and components should be treated with a certain level of respect. Components can be very sensitive to (static) voltages and currents can be deadly in the worst case. Almost everybody knows that a (too) humid environment should be avoided. However, an environment that's too hot can be just as damaging, particularly when the electronics are in a sealed enclosure. We can slowly begin to see that there are a number of conditions that should be satisfied when selecting an electronics enclosure.

Q *In what kind of environment will the enclosure be used?*

A The main function of an enclosure is to protect its contents from any external factors at the normal place of use that could affect the proper operation of the electronics. For a suitable level of protection against mechanical factors and liquids you should refer to the IP rating (Ingress Protection, see tables). On the other side of the Atlantic the NEMA rating (National Electrical Manufacturers Association [1]) is often used instead.

Q *Which material should the enclosure be made of?*

A Broadly speaking, there are two choices: metal or plastic. Plastic enclosures are often used at hobbyist end of electronics, since they have the most advantages, such as electrical isolation and price. In industrial applications, however, metal often wins the day due to its superior mechanical strength. You can also consider properties such as the thermal conductivity, resistance against fire, oils, acids, alkalis, corrosion, scratches, and the temperature range within which the material remains stable.
Another type of material is biodegradable plastic. The ASTM [2] has issued a set of standards that these materials should satisfy. "Biodegradable" could otherwise be widely interpreted; after 1000 years even 'ordinary' plastic will have decomposed.

Q *What should the enclosure look like?*

A You normally want your enclosure to look attractive and there are several factors that should be taken into account when you consider its appearance. From a commercial point of view it sometimes is the case that the appearance is more important than the ergonomics. It may be too expensive to have a unique, professionally styled enclosure for your own design, and you will be better off using an off-the-shelf one. One way to improve the appearance is to find a firm that will manufacture a front-panel to your specifications. It will add to the cost, but it will give a much more professional look. If you're more of a DIYer you should definitely have a look at a Fablab. For those of you not familiar with this concept: a Fablab is a workshop with (professional, industrial) equipment and tools that can be used by anybody to make almost anything.

It’s often application-specific

When we look a bit closer to home and take some of our circuits as an example, we can be more specific regarding the requirements for an enclosure. If we take a look at the *Improved Radiation Meter* from the November 2011 issue of Elektor [3], we’ll see that the sensor board needs to be in a shielded, light-proof enclosure, which should also be easy to open so the sample can be placed inside.

The shielding suggests that we should use a Faraday cage, which makes a metal enclosure the obvious choice. A metal enclosure won’t let any light through, so the second requirement is met as well. The circuit operates at a low voltage and current, so there are no specific safety-related issues. The choice for our prototype turned out to be a biscuit tin.

A bit more recent is the *New Precise Nixie Clock* from the May & June 2016 issue of Elektor [4]. For this project we thought it important that both the electronics as well as the Nixie tubes should be in the spotlight. Since the circuit has a GPS receiver, it would be impossible to use a metal enclosure. Both of these requirements are met by a transparent acrylic enclosure.

Another example is the *Nixie Bargraph Thermometer* from the July & August 2018 issue of Elektor [5]. With this project we also thought that its appearance was very important. Here we also used acrylic, and used its intrinsic light transmissive properties to create a stylish temperature display.

As you can see from the previous examples, there can be a large variation in the requirements of an enclosure for each project. It’s not for nothing that we say: start with the enclosure, and then design the insides so they fit!

180460-02

Web Links

[1] NEMA: http://www.nema.org

[2] ASTM: http://www.astm.org

[3] Elektor Improved Radiation Meter: www.elektormagazine.com/magazine/elektor-201111

[4] Elektor New Precise Nixie Clock: www.elektormagazine.com/magazine/elektor-201605

[5] Elektor Nixie Bargraph Thermometer: www.elektormagazine.com/magazine/elektor-201807

IP ratings for enclosures

First digit

IP	Protection against:	Meaning
0x	No protection	
1x	Large objects(>50 mm)	Protection against contact with a hand. Protection against ingress of solid objects > 50 mm
2x	Medium-sized objects(>12.5 mm)	Protection against contact with a finger. Measurement equipment is touch-safe. Protection against ingress of solid objects >12.5 mm
3x	Small objects(>2.5 mm)	Protection against contact with tools. Measurement equipment is touch-safe. Protection against ingress of solid objects >2.5 mm
4x	Tiny objects(>1 mm)	Protection against contact with tools. Protection against contact with wires. Protection against ingress of solid objects >1 mm
5x	Dust protection	Touch-safe due to completely closed enclosure. Not completely protected against dust, but sufficient to not affect normal operation.
6x	Dust-proof	Touch-safe due to hermetically sealed enclosure. Complete protection against dust.

Second digit

IP	Class	Protection against
x0	None	-
x1	Drip-proof Type I	dripping water
x2	Drip-proof Type II	dripping water onto equipment at an angle of 15°.
x3	Spray-proof	spraying of water (10 l/min) at an angle between -60° to 60°
x4	Splash-proof	spraying of water (10 l/min) at any angle
x5	Spray-proof II	spraying of water (12.5 l/min) at any angle
x6	Waterproof	water ingress when sprayed (100 l/min) at any angle
x7	Submersion-proof	water ingress when submerged (30 min at 1 m)
x8	Waterproof	water under stated conditions
x9	Moisture-proof	humidity of more than 90% or high power water jets

Bad Circuit Howler

It works...
but not the way it should.

By **Dr. Thomas Scherer**

A faulty circuit has found a gap in Elektor's quality assurance procedure in the HOMELAB PROJECT category. This is unfortunate, but there is another aspect. The circuit sort of works, but not in the way the author intended. The story is interesting and you can definitely learn from it.

What's that all about? In Elektor edition 5/2018, a 'Sinusoidal FM-LF Amplifier' was featured [1]. Subtitled: ‚Analogue is not dead'. Any electronics enthusiast with an interest in that branch of electronics will have checked that one out for sure. But then „hey — hows that supposed to work?" is sure to be their next thought.

The story begins...

We know that many of our younger readers were educated in the post-analogue era of electronics and to them any circuit which does not rely on ones and zeroes must surely have been taken from a wizard's book of spells. We are also lucky that our readership includes many analogue experts with years of experience under their belt. Shortly after the article was published we began to receive critical emails and comments, especially from German readers. We took another look and couldn't believe what we read — it looks like we truly dropped a clanger this time!

It started out like this: Elektor reader Hans-Norbert Gerbig sent us a circuit he had been working on for publication. It happened that one of our editors — who later admitted that analogue and RF were not his strong points — reviewed and approved the submission. Normally when there is some aspect of a circuit that anyone is not sure about we discuss it with colleagues who have more specialised knowledge in the field and then talk it over with the author. Sadly in this case it didn't happen.

According to Murphy, anything that can go wrong will go wrong, so in this case the article somehow missed out on its next level of scrutiny to carry out a detailed factual examination. As a result, the circuit and its description got all the way through to publication in four languages. Needless to say it should not have happened, but it did; we can only apologise.

To make a virtue of necessity we can analyse the author's original design thoughts and take a closer look at his circuit. You never know, we might learn something.

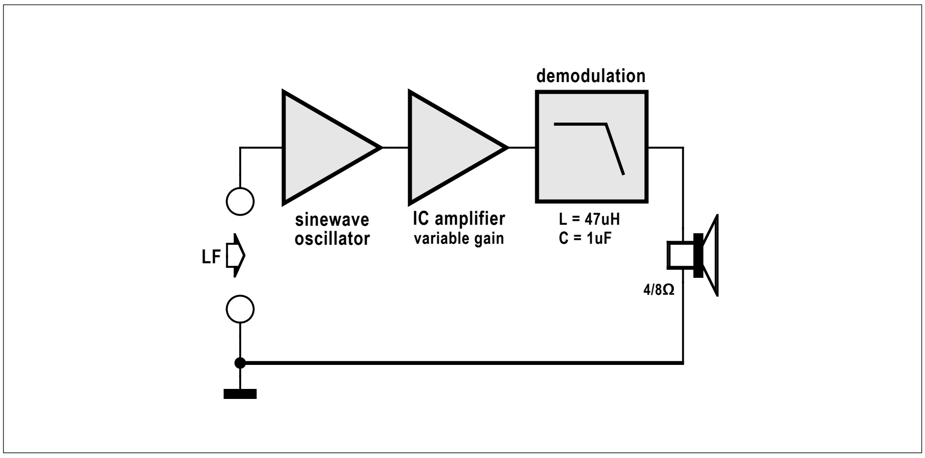


Figure 1. The block diagram outlines the flawed circuit concept.

The circuit concept

The 'sinusoidal FM-LF amplifier' was intended to offer an analogue alternative to the digital PWM audio amplifier principle. The circuit, as originally described consists of a frequency modulated RF sine wave generator, a standard opamp IC followed by a low-pass filter for demodulation purposes. A block diagram [1] of the concept was given and is shown again here as **Figure 1**. Any reader with a background in analogue electronics may already begin to suspect that the basic concept is a little flawed. How come? Well it's not just one simple error.

1. A low-pass filter is not able to act as a demodulator for RF signals with a symmetrical waveform. When the signal frequency is way above the low-pass filter's cut-off frequency rapid changes of the RF signal are simply averaged out. A low pass filter does what it should: it filters out RF and lets low frequency signals through. This applies regardless of the wave-

form; it does not matter whether it is sinusoidal, triangular, square or any other wave-shape. The type of signal modulation used also has (almost) zero effect; no matter if it is frequency or amplitude modulated. The signal modulated onto the RF is only recovered by a low-pass filter if it somehow changes the pulsewidth of the RF signal, i.e. if the integral of the waveform is influenced by the modulation. That is the function of a low-pass filter at the output of a digital PWM amplifier: to integrate.

2. Any audio amplifier has a limited bandwidth. According to the LM386 datasheet operating with a fixed voltage gain of 20 (26 dB) the output 3-dB roll off point occurs at 300 kHz and for 1 MHz signals the gain falls to 10 dB. The opamp in this circuit therefore acts as a low-pass filter to RF signals. The gain falls by around 30 dB / decade for frequencies above 1 MHz so that at 10 MHz, it is about 0.3. The opamp cannot amplify the RF signal, but acts as an RF attenuator in the circuit. The amplifier gain/bandwidth product applies also to the TBA820 and all amplifiers in general. **Figure 1** could not work even if it were possible to demodulate a frequency-modulated RF signal with a low-pass filter.

The basic concept has many flaws. The sinusoidal FM-LF amplifier concept cannot work, no matter which way you look at it.

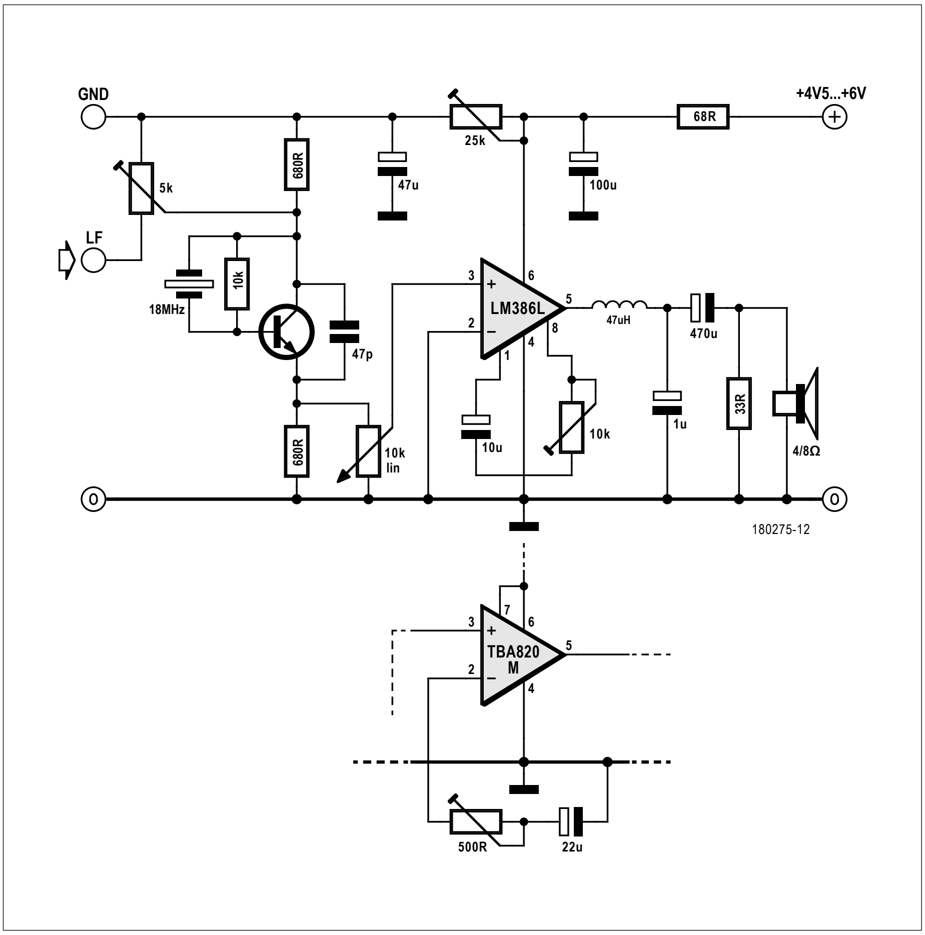


Figure 2. The 'sinusoidal FM-LF amplifier' circuit.

Despite everything, it amplifies!

When you actually build the circuit in **Figure 2,** connect it to a 5 V power supply, hook up an audio signal at the input and a speaker at the output you will most likely and surprisingly hears sounds coming from the speaker. What's going on? The amplifier actually amplifies. Any good circuit technician will see that pretty quickly. While critical appraisal of a circuit design is valid it gives support if we can actually build the circuit and take measurements to see if our interpretation is correct — more on that later.

The circuit contains a one-transistor crystal oscillator. According to the original script, a low-frequency signal applied across the collector resistor will modulate the oscillator frequency. The modulated RF signal at the emitter resistor is then fed via a 10 kΩ potentiometer to an opamp input. The voltage gain of this opamp can be adjusted from 20 to 200 by using another 10 kΩ potentiometer (at pin 8). The output signal is then fed to a low-pass filter made up of a 47 μH coil and 1 μF electrolytic capacitor. The 470 μF electrolytic capacitor connected in series with the output signal ensures any DC component of the output signal is isolated from the speaker. A 25 kΩ potentiometer is used to set the operating point of the FM sine wave generator. At least that's how it's supposed to work.

Why does the circuit amplify? Even if the oscillator was able to be modulated by the LF signal a significant level of the raw LF input still makes its way through to the opamp input. At audio frequencies the effective resistance of the combination of transistor and base resistor is given by the resistance (10 kΩ) divided by the current gain (typically 100 for a BF494) to give 100 Ω. The signal applied to the collector resistor therefore appears to be 'amplified' by a factor of 0.8 and fed to the 10 kΩ potentiometer. The resulting waveform at the opamp input is the sum of the LF and RF signals. There is no high-pass filter in the signal path to filter out the LF components. This IC, as already mentioned, will only amplify LF components in the signal, so the low-pass filter at its output is totally redundant.

Long story short: The amplifier provides some gain to the LF signal because it's just working as a normal audio amplifier, surrounded by unnecessary components. You could omit everything to the left of the 10 kΩ pot, as well as the 25 kΩ pot, the 68 Ω resistor and the low pass filter, you would be left with an amplifier that works better than the original. The sinewave oscillator simply performs no useful function.

The author had assumed his concept was correct when he saw an amplified LF signal at the output despite the presence of RF signals elsewhere in the circuit.

And in the real world?

Just so no one can say: „Yes, but that's all just theory!" I built a version of the

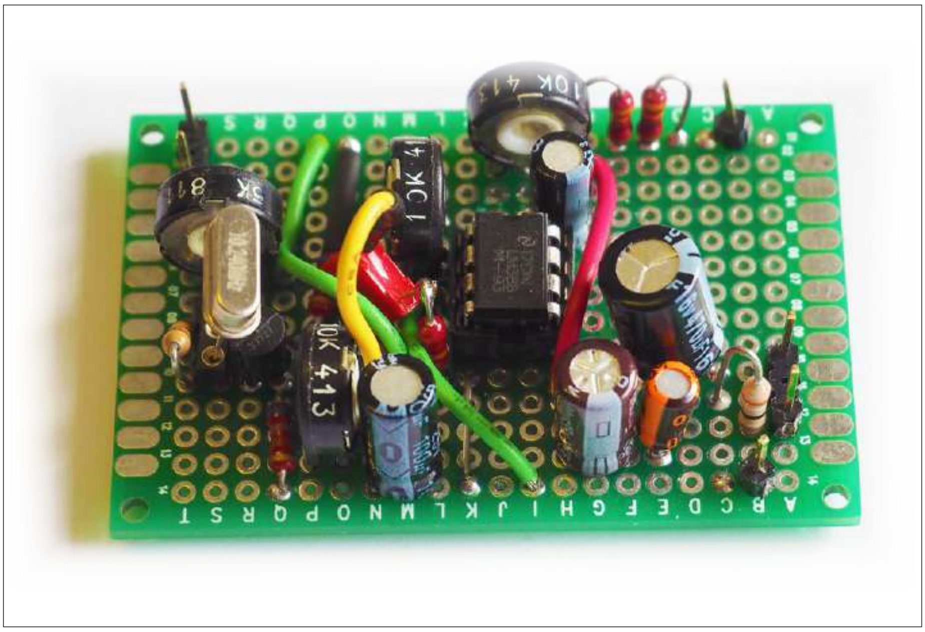
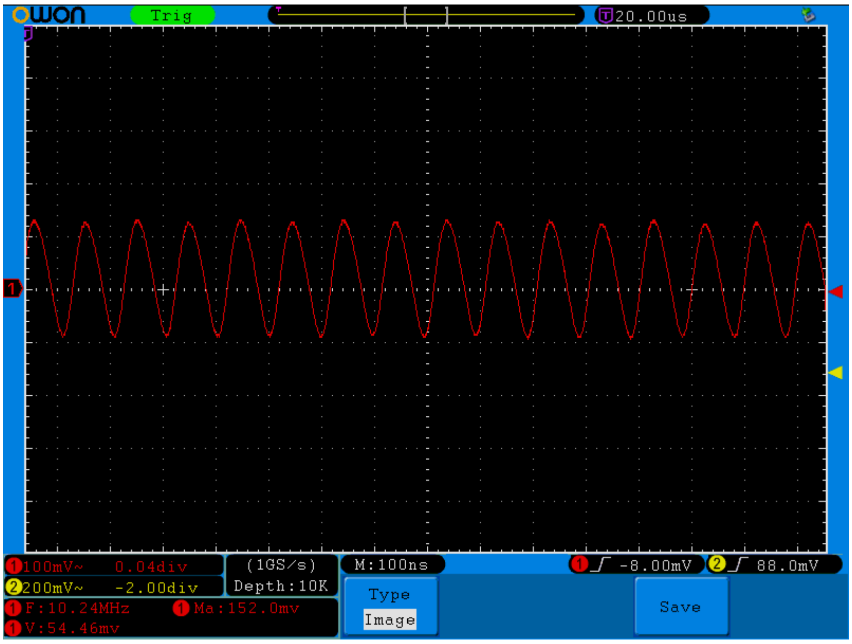


Figure 3. My „quick ‚n dirty“ prototype using the circuit from Figure 2.

circuit as shown in **Figure 2** on a piece of breadboard. As you can see in **Figure 3** it's not pretty but it does the job. I didn't have an 18 MHz crystal handy so I used a 10.24 MHz crystal that was in my box of bits. I also had a BF494 but needed to order an LM386. On power up I measured the signal at the collector of the transistor and was able to confirm that the oscillator was indeed working at the expected RF frequency with a signal level of about 55 mV_{rms}. The signal amplitude can be set within limits using the 25 kΩ potentiometer.

Figure 4 shows the RF signal after connecting a 600 Hz audio source at the modulation input. As claimed by the author you can see no amplitude modulation on the RF signal, but also, unfortunately with the naked eye, no frequency modulation either, even with the scope timebase adjusted to show up any modulation components.



```
1 100mV~     0.04div    (1GS/s)      M:100ns     1 ∫ -8.00mV  2 ∫ 88.0mV
2 200mV~    -2.00div    Depth:10K
1 F:10.24MHz       1 Ma:152.0mv
1 V:54.46mv
```

Figure 4. The 600 Hz and 10.24 MHz signals added. No modulation visible here.

By slowing the scope timebase down to resolve audio frequencies you can see in **Figure 5** that the signal consists of a fundamental 600 Hz sine wave from the LF signal generator summed (not mixed) with the 10.24 MHz oscillator signal. It can't be anything else.

From the opamp output you can also see the 'amplified' 600 Hz signal at the speaker. The noise levels seen in **Figure 6** are due to my untidy circuit layout and other circuit deficiencies — nothing to do with the circuit principle.

As you can see in this case, results of a practical investigation support the theory.

A closer look

There are some features of the circuit (**Figure 2**) that could be described as not exactly in accordance with 'best practice in circuit design'. To begin with, the opamp supply contains a low-pass filter made up of a 68 Ω resistor and 100 µF electrolytic capacitor, presumably to filter RF. The voltage drop across the resistor means that the LM386 output signal clips even at relatively low volume settings. If you want to output more than a few tens of mW its better to link pin 6 of the IC directly to the supply.

The fact that the input signal GND connection is not connected to the amplifier ground makes the circuit almost unusable. The test circuit did not work at the beginning because my function generator signal ground is not floating, just like the output of the bench power supply which powers the circuit. Connecting the function generator shorted the collector resistor to ground so that there was no sound from the speaker or measurable RF at the emitter resistor. Two 1 µF capacitors in series with the function generator signal output terminals resolved the problem, at the expense of the interference spikes visible in **Figure 6**, which were injected through the resulting ground loop. Note: Never use separate ground connections in a circuit unless there is a very good reason.

All good things come in threes: even with the two capacitors providing AC coupling of the input signal there was still no sound at the speaker. By adjusting the 25 kΩ potentiometer it was sometimes possible to hear a quiet tone at high resistance settings. How come? The transistor needs a minimum level of current to sustain oscillations and pass the signal on to the next stage, but this current also produces a voltage drop across the emitter resistor. Depending on the setting of the 10 kΩ potentiometer, however, just a few hundred mV DC is enough to send the opamp input into saturation so that the output goes up to the supply rail voltage. What's missing here is AC coupling so that the DC component of the signal cannot influence the signal level at pin 3. A capacitor between the transistor collector and the hot end of the 10 kΩ potentiometer leading to pin 3 is needed. After fitting the capacitor, I was able to use the 25 kΩ potentiometer to set an optimal current for the highest possible RF amplitude without saturating the opamp input stage.

Finally, during a pleasant telephone conversation with the author we were able to agree that his design will not work as a 'pulled oscillator'. Even if it did, it is not possible to modulate the oscillator frequency in this way with an LF signal to produce spectral frequency mixing. Despite all the above you can actually pick up the LF signal using an FM and a short wave receiver tuned to the oscillator frequency so there must be some FM and AM components in the signal but the modulation levels are very low and are likely to be the result of non-linearities of the transistor operating characteristics.

Conclusions

There is always lots of room for error and misunderstanding when you experiment in areas of electronics where you are not so confident, especially with analogue electronics. Advice from more experienced colleagues is always very helpful and don't forget that there's no such thing as a stupid question. The 'Sinusoidal FM-LF Amplifier' really is not such a rare example. Let's take a poll: Hands up anyone who's messed up with one of their own circuit designs… I've got to admit, both my hands are up or at least they were before I put them back on the keyboard. It's easy to do, and after all, mistakes are just essential diversions on the path to ultimate success. ◀

(180589-02)

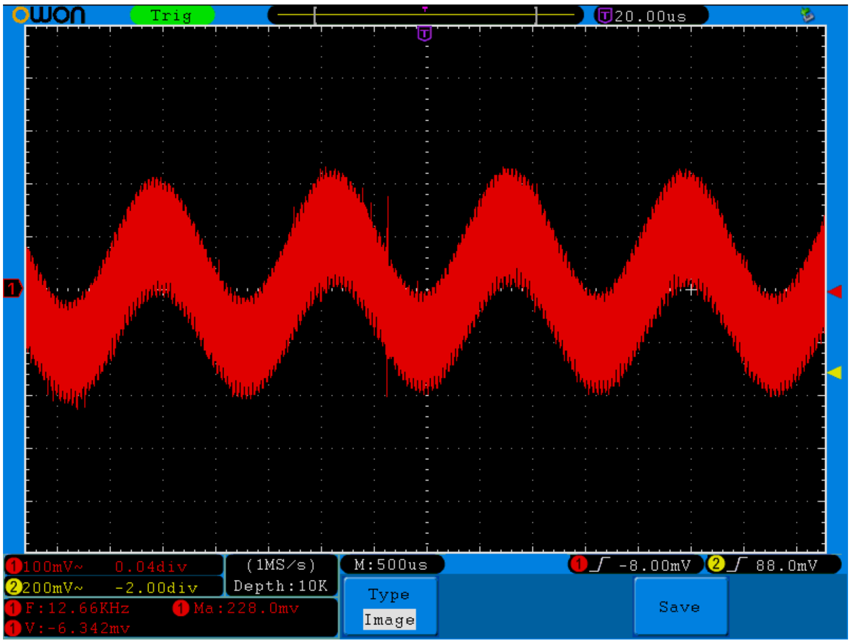


Figure 5. The opamp input is a waveform consisting of a 600-Hz signal summed with a 10.24-MHz signal (it filters out any RF components).

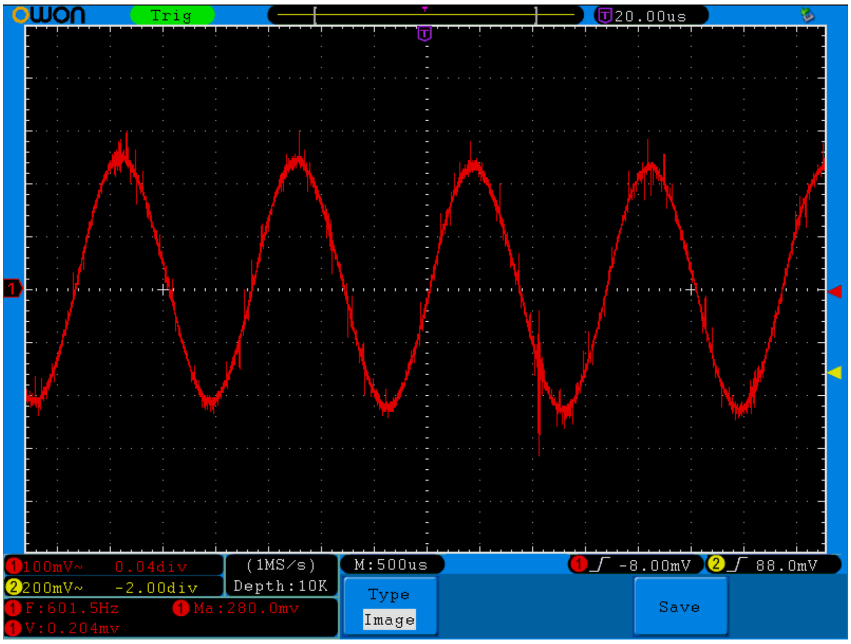


Figure 6. The 600-Hz signal at the circuit output is only slightly amplified. The RF has already been completely attenuated before the low pass filter.

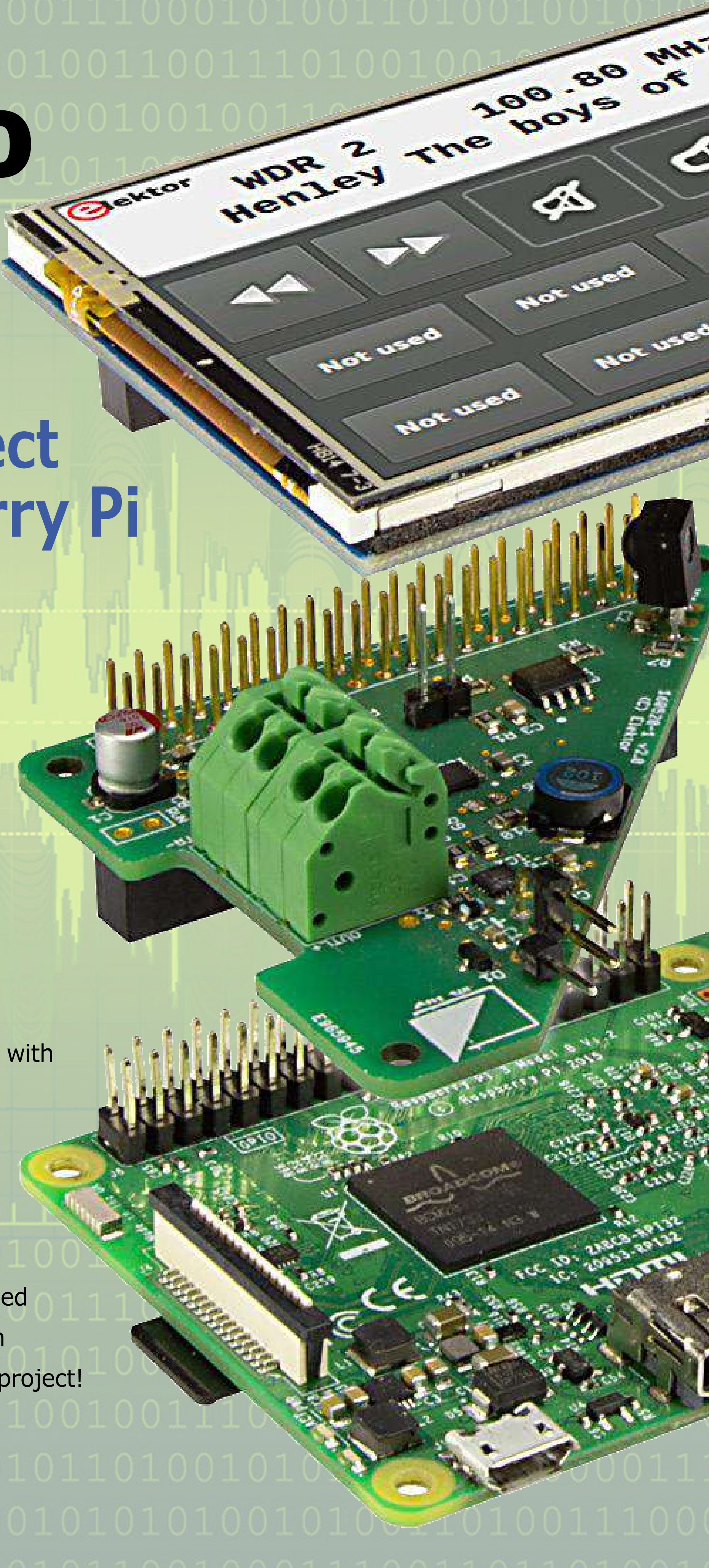
Web Link

[1] Article ‚Sinusoidal FM-LF Amplifier': www.elektormagazine.com/180275-01

FM Radio with RDS

A top HAT project for the Raspberry Pi

By **Fabian Bugelmüller, Christoph Fornezzi, Franz Parzer** (HTL Steyr, Austria), and **Ton Giesberts, Mathias Claußen** (Elektor Labs)



A Raspberry Pi plus a homebrewed HAT with driver software and a custom graphical user interface created specially for this application combine to make a unique FM radio that is not only eye-catching but a fascinating learning experience, thanks to extensively detailed documentation. Nobody can fail to learn something new from this wide-ranging project!

PROJECT INFORMATION

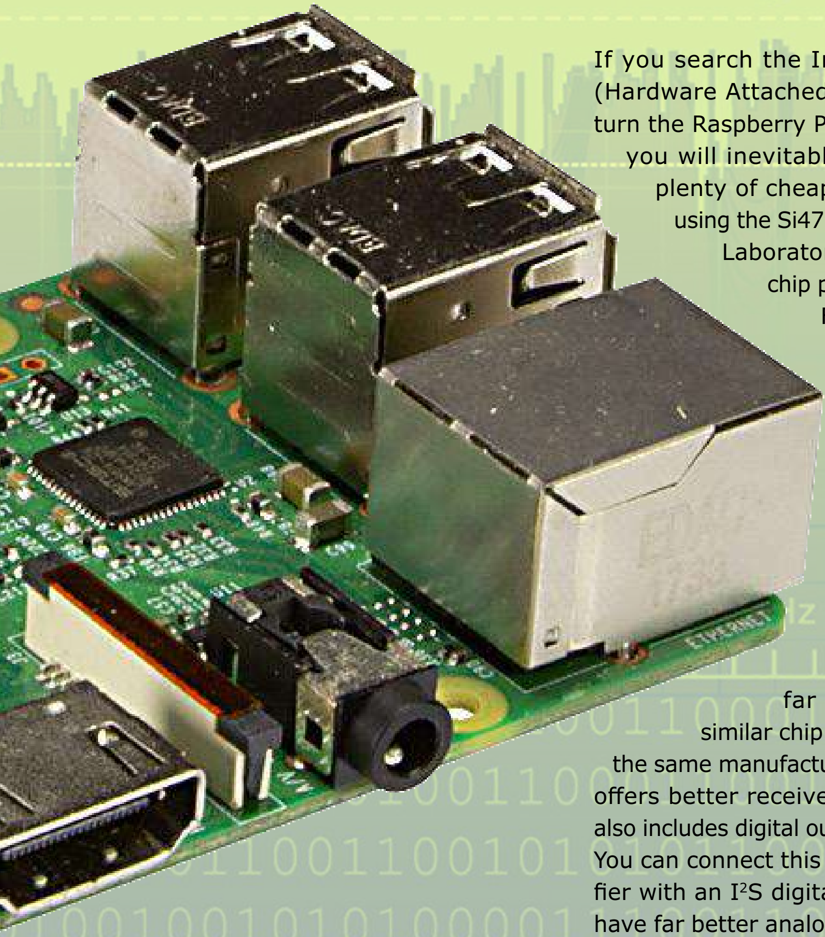
Raspberry Pi 2 or 3 (B+)
HAT
FM receiver
Touchscreen

entry level
[✓ intermediate level]
expert level

4 hours approx.

Equipment for
SMD soldering;
reflow oven recommended

€65 / £60 / $75 approx.



If you search the Internet for a HAT (Hardware Attached on Top) that can turn the Raspberry Pi into an FM tuner, you will inevitably stumble across plenty of cheap breakout boards using the Si4703 chip from Silicon Laboratories (SiLabs). This chip provides a complete FM radio with RDS (from antenna input all the way to analogue stereo output) and is ideal for use in mobile devices. But why stop there? What you might not realise is that you can do far better by using a similar chip called Si4731 from the same manufacturer, which not only offers better receive performance but also includes digital outputs for I²S audio. You can connect this to an audio amplifier with an I²S digital input, which will have far better analogue-to-digital converters than the Si4703. And the feature list doesn't stop there. Another desirable bonus with our HAT is an infrared receiver, enabling you to control the radio with a standard remote-control handset. For the user interface (GUI) we have a 3.5-inch touchscreen LCD.

Then, to go with this hardware, we have a Linux software bundle that contains the drivers for the ICs used, as well as a special radio app that offers some alternative, deluxe or indulgent methods of operating the radio. What's not to like now?

Incidentally, our FM Radio HAT works with all 'normal' versions of the Raspberry Pi from Type 2 onwards; the use of Type 1 is neither supported nor recommended. The HAT has also been tested with the Pi Zero and does run on it — but it's at your own risk alone...

Wide-ranging receiver

In the schematic of **Figure 1** you can see the small number of components that comprise our FM Radio.

Si4731

For the radio receiver chip, we're using the Silicon Laboratories Si4731-D60

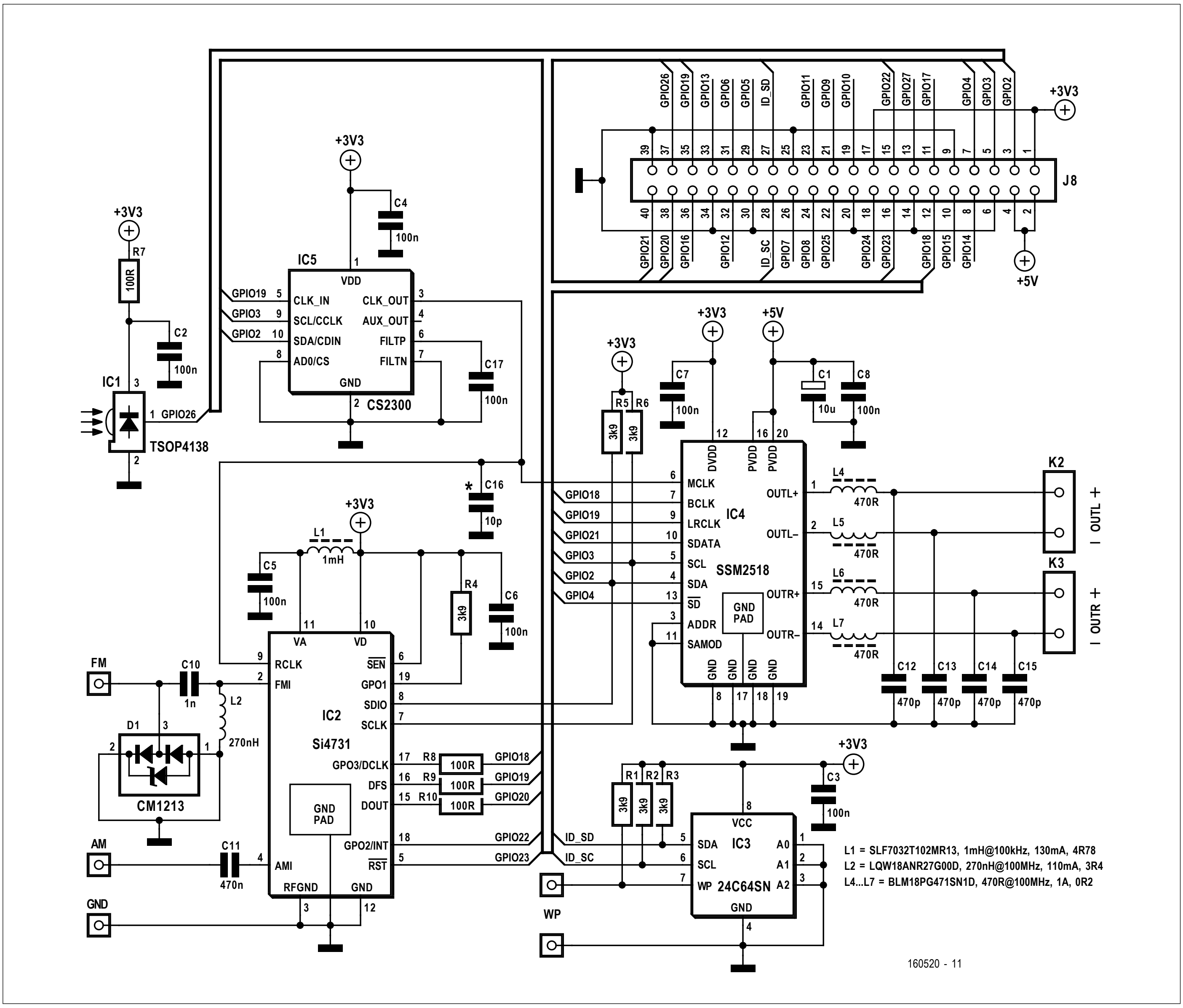


Figure 1. Schematic for the FM Radio project.

'Broadcast AM/FM Radio Receiver' [1] in its tiny 20-pin QFN package (firmware version D60). The receiver handles not only FM in the full worldwide frequency range (from 64 to 108 MHz, meaning that it also covers Japan and certain former Soviet territories) but also AM medium wave (520 to 1710 kHz) (**Figure 2**). The Si4731's digital audio interface operates in slave mode and supports a variety of audio data formats (transmitted with MSB first), including I2S and left-aligned modes. The chip can evaluate RDS and output the audio data in I2S format in the popular sampling rates of 32 kHz, 44.1 kHz and 48 kHz. The Si4731 is configured directly from the Raspberry Pi over the I²C interface.

The IC has two separate antenna inputs, one for FM and the other for AM. With FM, L2 is used to achieve resonance. The suppressor diode D1 looks after ESD protection, whilst C10 filters the direct component from the signal. Incidentally, for the FM antenna we have achieved better results with 75 cm of thin wire than with a customary telescopic whip antenna. We have plenty of experience with FM reception but have not yet had time to experiment with AM. The AM antenna needs to be a ferrite rod or frame antenna of 180 to 450 µH, which the data sheet says must be connected to the AM input of the Si4731 via a 470 nF capacitor (C11). The number of turns used depends of course on the properties of the particular type of ferrite rod used. Checking out the Internet will supply possible solutions for a suitable antenna.

C5 and C6 are decoupling capacitors; L1 ensures a stable supply voltage. R8 to R10 are the protective resistors specified in the datasheet.

SSM2518

For outputting the audio, we selected the SSM2518 from Analog Devices [2]. This Class D amplifier has a digital serial audio interface that accepts data in I2S format and is also controlled via the I²C bus. Both stereo output channels supply 2 W into 4 Ω. The circuitry of the IC corresponds to the specifications in the data sheet. C7 decouples the 3.3 V supply (Digital Supply Voltage, DVDD), C1 and C8 the 5 V supply (Speaker Supply Voltage, PVDD).

A polymer capacitor with a very low ESR of just 30 mΩ is used for decoupling the 5-V power supply for the power section of the IC. According to the data sheet, filtering at the loudspeaker outputs is required only if the speaker leads are longer than 20 cm. In practice, it's evident that, even if this stipulation is observed, the speaker leads radiate so much RF hash that RDS then becomes un-receivable. Fortunately, this annoyance can be resolved simply by providing filtering at the outputs (ferrite beads L4 to L7 and capacitors C12 to C15). R5 and R6 are the pull-up resistors required for the I^{2}C bus.

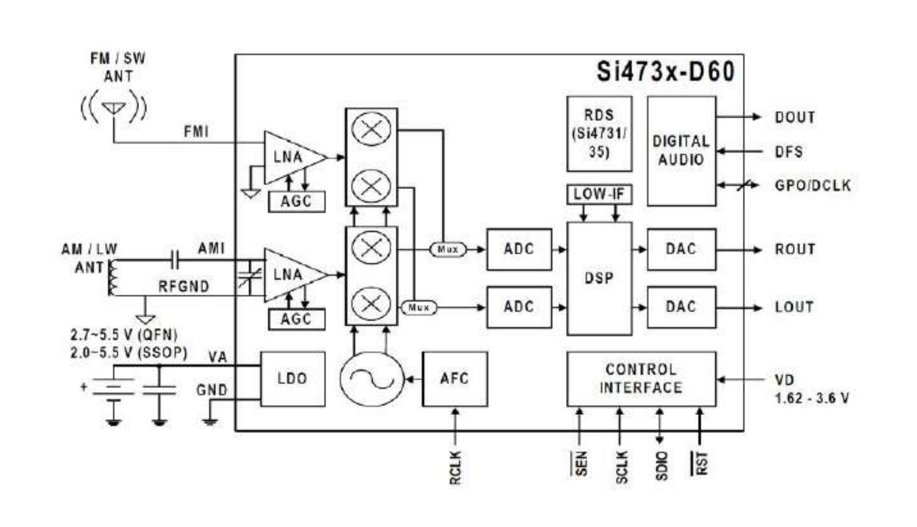


Figure 2. Internal circuitry of the Si4731 (Silicon Laboratories).

IR receiver

A 38-kHz IR receiver (IC1, a TSOP4138) picks up signals from an infrared transmitter of RC5-type remote controls (also 36 kHz ones used with TVs and stereo music systems). It then amplifies, filters and demodulates the signals, outputting the information to the GPIO26 line of the Raspberry Pi. The Raspberry Pi interprets the infrared signal and executes the relevant commands for controlling the radio.

EEPROM

In order for the hardware to comply with the HAT specifications (see below), an EEPROM with specific information must be included. IC3 is a common 24C64 EEPROM that also communicates over the I^{2}C bus.

CS2300

In our initial test setups, an oscillator was used as clock generator. However, problems arose because the master clock signal generated by the oscillator did not exactly match that of the I2S clock, and bits were lost repeatedly. Since the I2S clock should be exactly 1/256-th of the system clock, we substituted a CS2300 frequency multiplier for the oscillator and this multiplies the I^{2}S clock by 256. This makes everything jitter-free and the system and I^{2}S clocks match perfectly.

The CS2300-CP-CZZZ from Cirrus Logic is configured and controlled over the I2C bus. The circuitry is in as given in the data sheet. However, the 10-pF capacitor C16 at CLK_Out for attenuating harmonics is not required and is therefore not fitted.

Power supply

We hardly need mention that for powering the radio, you should use a good quality power supply that can actually deliver 3 A (not just on paper!). The power supply has to supply not only the RPi, the display and the radio receiver, but also the audio power amplifier with 2 x 2 watts. And how can the quality of a power supply be that good if it only costs a couple of pounds?

Software for the hardware

In order to make the hardware usable with Linux, we need to have special drivers for the individual hardware components available in (and integrated with) the Linux kernel. Conveniently, drivers for the class D amplifier are already included in the kernel, whilst the existing driver for the radio tuner was modified for the Si476x device family. Here is where most of the changes needed to be made, since these are two rather different device families. Drivers from a different device had to be taken and modified for the CS2300 frequency multiplier as well.

Developing drivers is laborious work and requires programming experience. Integrating drivers into the Linux kernel is equally taxing, as is compiling them without errors. But don't panic! We have created an image for the FM Radio hardware that is ready to go and can be copied easily to the SD card of the Raspberry Pi. This works just like any other distribution, such as Raspbian or Kodi. The custom-made application software piRadio is already included in the image.

But if you wish to penetrate further into the software entrails and/or want to cre-

Web Links

[1] HAT specifications: https://github.com/raspberrypi/hats

[2] Si4731 datasheet: www.silabs.com/documents/public/data-sheets/Si4730-31-34-35-D60.pdf

[3] SSM2518 datasheet: www.analog.com/media/en/technical-documentation/data-sheets/ssm2518.pdf

[4] Project page: www.elektormagazine.com/180119-02

[5] Elektor-Labs, FM-Radio: www.elektormagazine.com/labs/fm-radio-receiver-with-rds-for-raspberry-pi

[6] Elektor-Labs, PiRadio: www.elektormagazine.com/labs/piradio-for-fm-radio-receiver-with-rds-for-raspberry-pi-160520-1

[7] FM Radio using Volumio: https://github.com/rpi-Receiver/

[8] FM Radio atGithub: https://github.com/ElektorLabs/160520-FM-Radio-Receiver-with-RDS-for-Raspberry-Pi

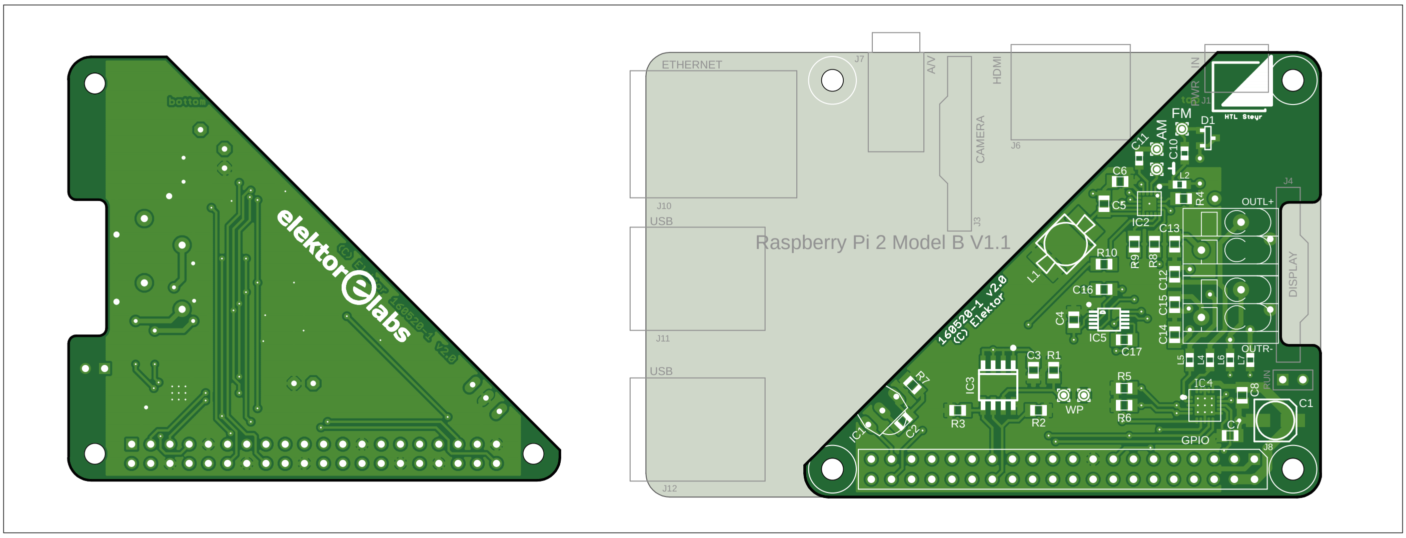


Figure 3. Upper and lower views of the FM Radio HAT.

ate your own distribution (possibly with a different user interface), you will find hints, examples, drivers, instructions and much more among the numerous documents on the Elektor project page [4].

Three-cornered HAT

A HAT is an add-on board for the Raspberry Pi that meets the HAT specifications [1]. This HAT is attached to the 40-way GPIO headers of the Raspberry Pi. Two of the 40 pins are reserved exclusively for connecting an 'ID-EEPROM', in which information is stored about the board. This data includes which functions the board has, which GPIOs have to be configured how and whether power for the Raspberry Pi and HAT is supplied using the Micro-USB connector on the Raspberry Pi or via the HAT.
Essentially a HAT must conform to the following conditions:

- various fundamental add-on requirements;
- valid data in the EEPROM (vendor info, GPIO map and device tree);
- 40-way female header;
- when the supply voltage is fed in via the HAT (backpowering, backfeeding), at least 1.3 A must be made available for the Raspberry Pi.

However, nowhere is it stated that the PCB of a HAT needs be rectangular (**Figure 3**)! Which is why our Hardware Attached on Top is formed of a triangular PCB (with a small rectangular notch cut for the touchscreen connector). Besides its unconventional looks, this shape also saves production costs. Essentially, we laid out the components in separate, logical blocks, wherever possible. Since the infrared receiver and the EEPROM have nothing to do with the main function, they are clearly separated from the other component groups. The radio receiver is located in the upper part of the board, the frequency multiplier in the middle and the Class D amplifier in the lower right corner. In the radio receiver section, the power supply peripherals, plus the FM, AM and digital output peripherals, are all arranged separately (as effectively as is possible on such a small board). In the amplifier section, the filters are placed so that the ferrite beads are as close as possible to the IC, and the capacitors as close as possible to the speaker output terminals. All auxiliary capacitors are attached directly to their ICs. A jumper is provided for write-protecting the EEPROM and the EEPROM can be written to only when the jumper is inserted.
The board of the FM radio receiver is attached to the Raspberry Pi with 17 mm-long combined male/female standoff pillars (M2.5) (**Figure 4**). For mounting the display, a fourth 17 mm-long combined male/female standoff pillar (also M2.5) is required next to the Ethernet jack as well as an M2.5 nut that compensates for the missing thickness of the PCB at this location. You then attach the radio board to the Raspberry Pi and screw four 14 mm-long M2.5 spacer pillars (again combined male/female) into the lower spacers. The four spacer bolts that support the touchscreen are of the same length; the inward thread points upwards. This relieves the loading on the 26-pin display connector.

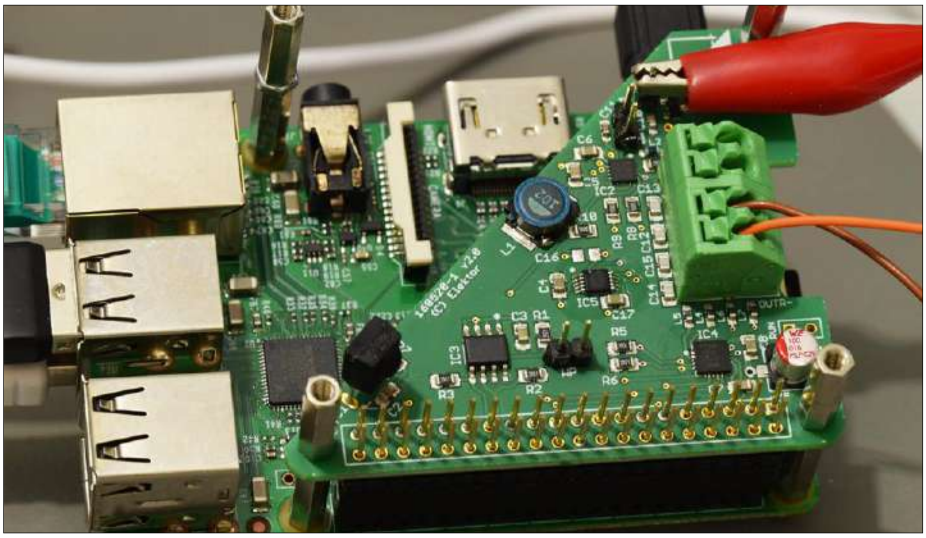


Figure 4. Mounting the HAT on the Raspberry Pi.

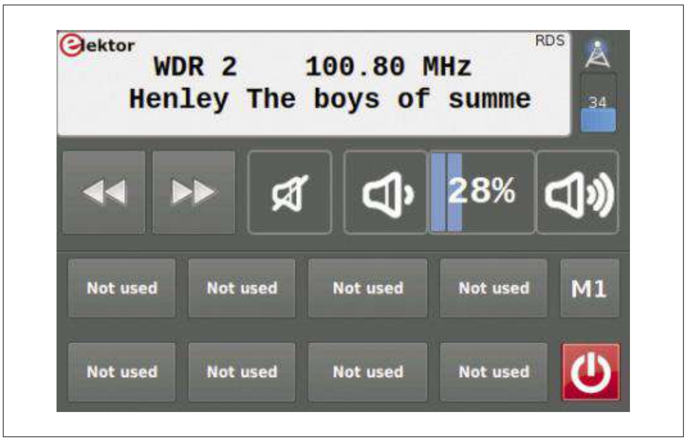


Figure 5. Main display of piRadio.

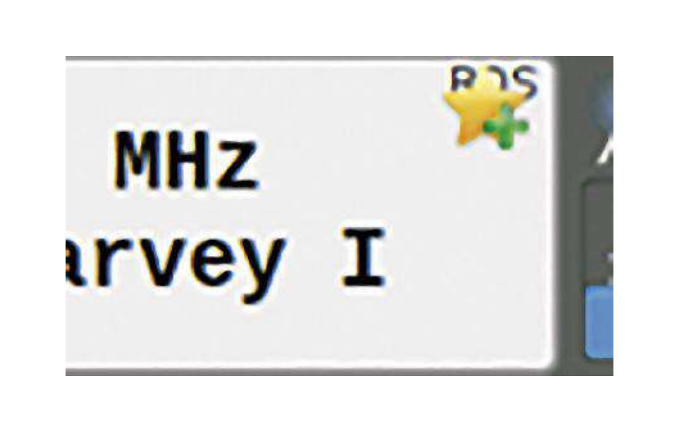


Figure 6. After storing a station.

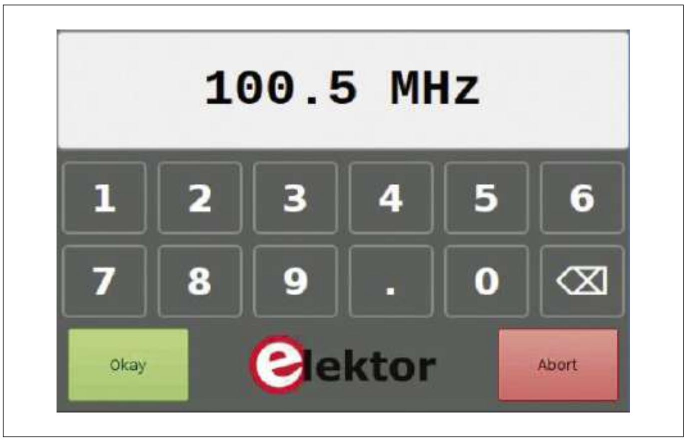
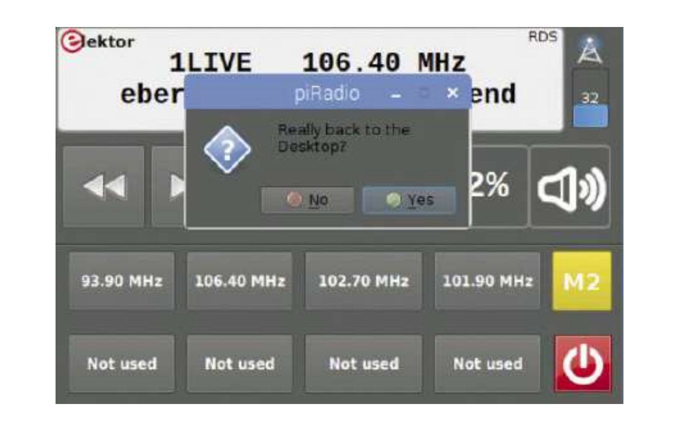


Figure 7. Entering a frequency direct.



Really back to the Desktop?

Figure 8. Quit radio listening and return to Raspbian?

If the LCD is not fitted, but instead replaced by an external (touch) screen, the 14 mm spacers and nuts are of course omitted, requiring the board to be fixed with M2.5×6 screws.

Software for the radio

If you are looking for radio apps, you will quickly discover KRadio, Gnomeradio or RDS Surveyor. There are also some TV apps that you can use for radio listening. Each app has its own strengths and weaknesses, but they're not ideal for use with IR remote controls and touch screens, because they were designed for a desktop system, mouse and keyboard. With a small 3.5-inch low-resolution touch screen, they are barely usable at all.

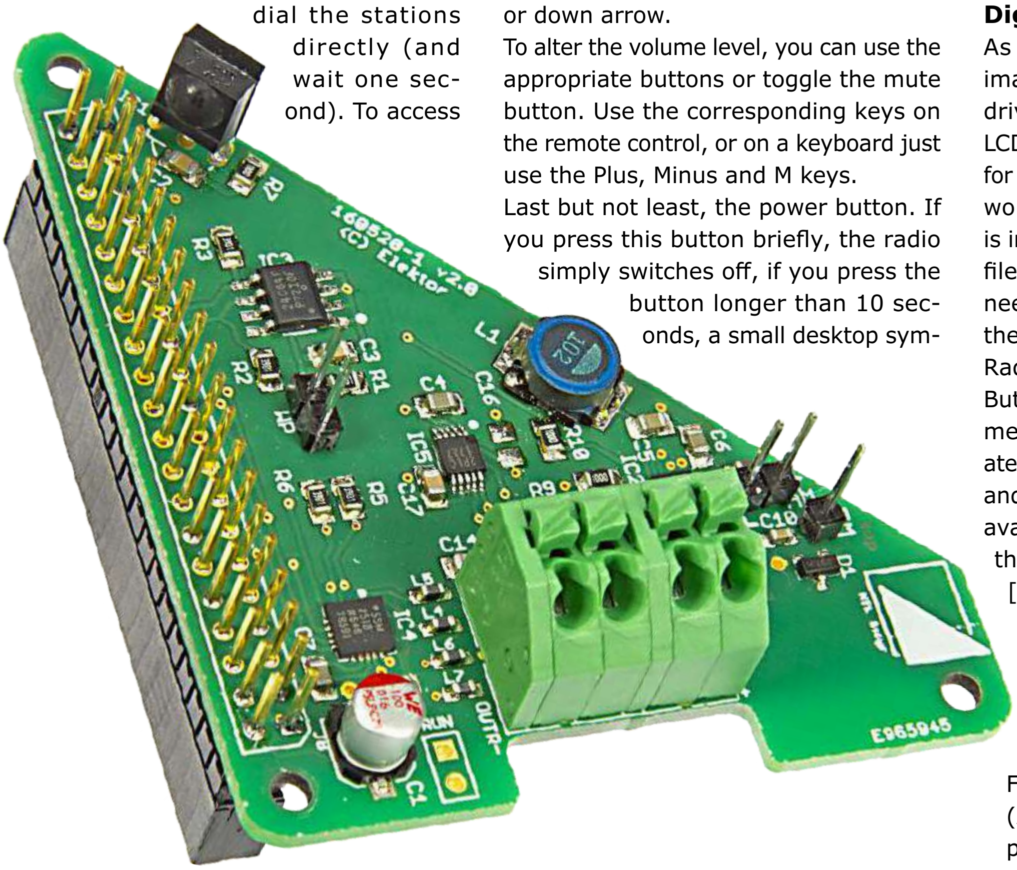
That's why Mathias Claussen at Elektor Labs developed a program called piRadio to control the FM receiver. The software was developed specifically for this hardware and for operation using a 3.5-inch touch screen with 480×320 pixels (like the popular Waveshare35a model). The control is totally intuitive.

Figure 5 shows the piRadio display. In the upper area (if the 'RDS' logo is shown at top right) the Program Service name (PS) is displayed to the left of the current tuning frequency. The current RDS radio text (RT) appears on the second line. If only the station name can be decoded, the display will use lighter characters for better readability. If no RDS information is received, just the current frequency is displayed. Received signal strength is displayed in a small bar graph on the far right below the antenna symbol. The higher the value, the better the reception.

In the lower area there are eight buttons that are labelled 'Not used'. This is the station memory with eight buttons. There are four such pages, between which you can switch using the M[x] key, meaning that you can save a total of 32 stations. To store a frequency, first go to the page on which the station is to be stored, select the frequency with the arrow keys and press the desired key on the touch screen for 5 seconds. A 'Bookmarked' star (**Figure 6**) appears in the upper right corner of the display under the RDS logo when the station is stored.

Since changing frequency using the arrow keys is usually awkward, we have implemented an interface for direct input (**Figure 7**). When you tap the display, you are given an entry form for the frequency (use the 'Menu' button on the remote control, the [E] key on the keyboard). The frequency in current use will be displayed and you can simply start entering a new frequency, as the first digit entered deletes the old frequency. For 98.55 MHz you simply enter [9][8][.][5[5]. If you make a mistake, simply press the Backspace key. If the frequency you entered is within a valid range, the 'Okay' button appears. If you are not satisfied with your choice, you can press 'Abort' and return to the main user interface. Keyboard entry is carried out using the same settings, but with a remote control the Enter key corresponds to the Backspace key, the A/B key to the decimal point and the OK key naturally to 'Okay'.

After saving a station, you can access all of the stations memorised with a remote control or keyboard by pressing [xy] on the remote control, where x is the station page and y is the station number on that page. For example, to select the second station on the first page, press 12 and to get the fourth station on the second page, press 24. For page 1, however, you do not have to press page 1 first, but can dial the stations directly (and wait one second). To access another station on the same page, you can also enter [0y]. If you select a preset that does not exist or has nothing stored, the radio will not change frequency.



Between the text window and the station memory on the left are the tuning controls (arrow keys) mentioned previously, which must be pressed for three seconds before a search up or down starts. On a remote control you start the search with the fast forward or fast reverse key, or on a keyboard with the left or right arrow. To increase the frequency up or down one step at a time, press the up or down arrow.

To alter the volume level, you can use the appropriate buttons or toggle the mute button. Use the corresponding keys on the remote control, or on a keyboard just use the Plus, Minus and M keys.

Last but not least, the power button. If you press this button briefly, the radio simply switches off, if you press the button longer than 10 seconds, a small desktop symbol appears in the upper right corner of the display. If you then release the button, a message window will appear asking if you really want to return to the desktop. If you click on 'Yes', the application closes and you can access the desktop (**Figure 8**). Otherwise, the message will disappear and the application will continue to run.

An overview of all commands and inputs you can make on the touch screen and their equivalents on the keyboard and remote control can be seen in **Table 1**.

Digging deeper

As already mentioned, we provide an image based on Raspbian, in which the drivers for the radio hardware and the LCD are already installed as well as LIRC for remote control via RC5. The Qt framework employed as the basis for the GUI is installed and adapted to the autostart file. The image is ready to go and only needs to be copied onto the SD card of the Raspberry Pi in order to use the FM Radio with all its functions.

But it doesn't end there. In the development phase of the project we also created numerous spin-offs, text documents and program code, which we have made available on the project page [4] and at the two related pages on Elektor Labs [5][6].

● The basis of the FM Radio project is the diploma thesis (RpiReceiver.pdf) of the authors Fabian Bugelmüller and Christoph Fornezzi at HTL college in Steyr (Austria). This describes the complete development of the hard-

Table 1. PiRadio software commands for touchscreen, keyboard and remote control handset.

Function	Keyboard	Remote control
Volume up	+	Vol+
Volume down	–	Vol–
Mute	M	MUTE
Quit	Q	POWER
Frequency up	Up arrow	CH+
Frequency down	Down arrow	CH–
Search frequency up	Right arrow	Videotext rot
Search frequency down	Left arrow	Videotext blau
Display/hide frequency	E	Menu
Erase frequency	BACKSPACE	ENTER
Decimal point	Full stop or comma	A/B
Confirm frequency entered	ENTER	OK
Discard and terminate	ESCAPE or E	Menu

COMPONENT LIST

Resistors
R1-R6 = 3.9kΩ 0.125W 1%, SMD 0805
R7-R10 = 100Ω 0.125W 1%, SMD 0805

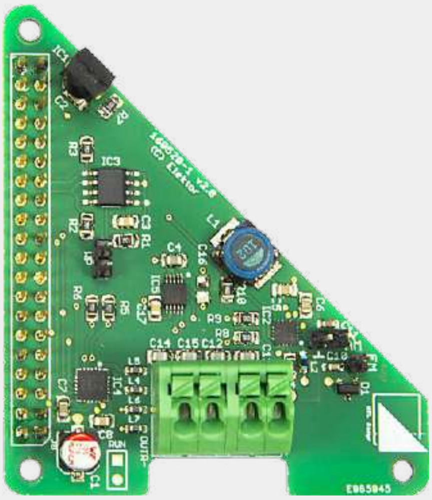
Capacitors
C1 = 10µF 16V, 20%, 5mm radial SMD, 0.03Ω/2.2A
C2-C8,C17 = 100nF 50V, 10%, X7R, SMD 0805
C10 = 1nF 50V, 10%, X7R, SMD 0603
C11 = 470nF 16V, 10%, X7R, SMD 0603
C12-C15 = 470pF 50V, 5%, C0G/NP0, SMD 0805
C16 = 10pF 50V, ±0.5pF, C0G/NP0, SMD 0805, not fitted

Inductors
L1 = SLF7032T102MR13-2PF, 1mH @100kHz, 130mA, 4.78Ω (TDK)
L2 = LQW18ANR27G00D, 270nH @100MHz, 110mA, 3.4Ω (Murata)
L4-L7 = BLM18PG471SN1D, 470Ω @100MHz, 1A, 0.2Ω (Murata)

Semiconductors
D1 = CM1213A-01SO, SMD SOT-23 (ON Semiconductor)
IC1 = TSOP4138, 3-pin, wired (Vishay)
IC2 = SI4731-D60-GM, SMD 20-pin, QFN (Silicon Labs)
IC3 = CAT24C64WI-GT3, SOIC-8 (ON Semiconductor)
IC4 = SSM2518CPZ, LFCSP 20-pin -WQ (Analog Devices)
IC5 = CS2300-CP-CZZ, MSOP-10 (Cirrus Logic)

Miscellaneous
K2,K3 = 2-way PCB screw terminal block, 3.5mm pitch
J8 = 40-way (2×20) GPIO receptacle strip for RPi (extra long)
4 pcs standoff 17mm, M2,5 (male-female), e.g. Ettinger 05.12.173
4 pcs standoff 14mm, M2,5 (male-female), e.g. Ettinger 05.12.143
5 pcs M2.5 nut
5-pin SIL pinheader, vertical, 0.1" pitch, distributed for AM/FM/PW pads
J8 = 3.5" touchscreen-LCD, e.g. Waveshare 3.5inch RPi LCD (A)
Raspberry Pi Model 2 or up
PCB # 160520-1 v2.0

C9,L3,K1: not fitted



ware and all drivers, the content of the EEPROM, the integration under Raspbian (and much more) in far greater detail.

- The project download contains an overlay for Raspbian (rpi-receiver-linux-rpi-4.9.y), specifically the special firmware required for FM Radio, which must be integrated into the Raspbian kernel. Detailed installation instructions explain the procedure.

- For the radio to work as desired, adjustments must be made to Raspbian to install the LIRC IR remote control and redirect the HDMI user interface to the touch screen LCD connected to the GPIOs. Both are explained in the installation guide.

- The piRadio application was developed at Elektor Labs. The source code is available, enabling modifications (different layout, additional functions, number of station memories, use of the RC5 remote control) to be made easily.

- The app was developed with the free C++ toolkit Qt. More information about software development (including a WebSocket API) can be found on the Elektor Labs webpage [6].

- If you don't wish to make any modifications to FM Radio with piRadio and, as far as software is concerned, want to get it going straight out of the box, you will find a ready-to-use image for the SD card of the Raspberry Pi.

- The radio can also be used with other apps, such as RDS Surveyor. This Java program is available for download along with installation instructions. The RDS Surveyor really reveals every fine detail that the Radio Data System broadcasts.

- A video shows the operation of FM Radio and its GUI.

- There is a guide on how to integrate the hardware with the Volumio open source audiophile music player [7].

- There is also a guide on how to integrate Qt into Raspbian [8].

So, whether you just want to build an FM radio, modify some details in the hardware and software or else use the knowledge gathered in the course of this project for your own developments, the project page has something for every user! ⏮

180119-02

@ WWW.ELEKTOR.COM

→ **Unpopulated board # 160520-1 V2.0**
www.elektor.com/rpi-receiver-160520-1

→ **Raspberry Pi 3 B+**
www.elektor.com/rpi3b+

Verilog Basics

Fundamental features of the hardware description language

By **Jörg Zollmann** (Germany)

Alongside VHDL, Verilog is one of the best known and most widely used hardware description languages (HDLs) for programming CPLDs and FPGAs. In this article we give an overview of the most important aspects of the language.

To recap: a hardware description language allows you to specify a digital circuit with the help of a textual description. The circuit can subsequently be functionally tested using a simulator. Also, a description written in an HDL can be converted into real hardware using software tools in a process called 'synthesis'. Verilog and VHDL both have extensive user communities, both in the industrial world and among enthusiasts. Anyone who works on CPLD- or FPGA-based projects will find it advantageous to learn at least the basics of both languages. In this article we will present the basics of Verilog, to the point where you should be able to use it in your own applications and get to grips with larger-scale designs such as the 'DIY processor' to be published shortly in ElektorLabs magazine.

Conventions

The syntax of Verilog is very similar to that of C. The learning curve for the language is therefore generally not as steep as in the case of VHDL and it takes less time and fewer lines of code to arrive at your first working design. Unlike VHDL, Verilog is case sensitive. Conventionally Verilog source files have file extension '.v' (or '.vh', '.sv', or '.svh'). As in C, Verilog single-line comments are introduced by two slashes (`//`), and multi-line comments start with `/*` and finish with `*/`. White space is not significant in Verilog. Identifiers (variable names) consist of alphanumeric characters plus underscore and the dollar sign (`a` to `z`, `A` to `Z`, _ and `$`), but the latter cannot occur as the first character of a name. The equivalents of braces in C (`{` and `}`) in Verilog are the keywords begin and end. As in the world of software development, there are various recommended coding styles, differing in indentation depth and in the use of `begin` and `end`. For a beginner in the language the simplest approach is to bracket every sequential block and every conditional code branch with `begin` and `end`.

Numerical constants and data types

Numbers are written in Verilog according to the template `<size>'<signed><radix>number`. Here `<size>` specifies the number of bits (not digits) that will be used to represent the value. The parts of the template in angle brackets are optional, and by default numbers are interpreted as unsigned decimal values. The number 42, for example, can therefore be represented in any of the following ways.

```
6'b101010  // binary (MSB after the ' )
6'o52      // octal
6'h2a      // hexadecimal
6'd42      // decimal
6'b10_1010 // the underscore '_' can be used
           // to improve readability
'd42       // with no prefix the value
           // is 32 bits wide
```

Verilog distinguishes between two kinds of data type. The first of these is the 'net', of which the most important example for synthesis is the `wire`. As the name indicates, a wire is used to represent a connection. The other data type is the register, of which the most important example is the `reg`. Integer, real, time and realtime are also examples of register types but are less important for synthesis. A register type implies underlying storage: whenever a value must be (temporarily) stored somewhere, a register type must be used. That is not to say, however, that whenever a `reg` is used a flip-flop or other physical storage element is automatically synthesized: more on this later. Data types in Verilog employ a four-valued logic: see **Table 1**. Arithmetic and logical operations are defined for both kinds of data type, and **Table 2** shows the most important operators, their functions, and some simple examples.
If no type is declared then Verilog defaults to assuming a one-bit `wire`. That means that if you forget to include a type declaration for a signal then the synthesis or simulation tool will not report an error, but rather just a warning that the variable in question has been automatically assumed to be a one-bit wire. Verilog does not require any 'packages' or functions to be linked in or called to provide type conversion operations. This makes code simpler and more compact, but comes with the

Table 1. Four-valued logic in Verilog.

Logical value	Interpretation
0	Logic 0 or false
1	Logic 1 or true
x	Don't care
z	High impedance

disadvantage that a Verilog compiler sometimes lets things pass that a VHDL compiler would not allow. So, for example, an assignment between vectors of differing bit-widths leads only to a warning that truncation has occurred. The consequence of this is that a circuit might become impossible to synthesize or might have a function different from what is wanted. Suppose that we declare a vector of type reg [7:0], that is, an eight-bit wide register, and assign different values to it as follows.

```
reg [7:0] bus1 = 8'b1011; /* bus1 = 00001011 MSBs are
   zero-padded */
reg [7:0] bus2 = 3'b1101; /* bus2 = 00000101 bit 3 is
   truncated */
```

When evaluating a numerical constant the specified size takes priority. When the value is subsequently assigned to the eight-bit bus the MSBs are padded with zeros.

Modules

The basic unit of Verilog code is the module. A Verilog module begins with the module keyword and ends with the endmodule keyword. Following the module keyword is the name of the module, followed by an optional parameter list and then the port list. The port list specifies the names of all the module's inputs and outputs. In older versions of Verilog only the names of the signals were given, but since 2001 a port declaration is also permitted to appear directly in the port list specifying the direction of signal flow (input or output) and the type of the signal. After the port list comes the actual code describing the function of the module using various assignments or statements.

```
module a (input   wire   a);
// empty module
endmodule
```

Verilog supports hierarchical circuit design. A module can thus instantiate submodules. If it is desired to adapt the behavior of a submodule, this can be done through the use of a parameter. A parameter is 'passed' to the submodule through the parameter list, which is distinguished from the port list through the use of the # symbol.
The following example module has a parameter N with a default value of 1. This value can be overridden when the module m is instantiated. Module m comprises a submodule a_1, which is an instance of the module listed above. When a module is instantiated the signals from the instantiating module can be connected explicitly to the ports of the submodule by name, or the connection can be implicit by its position in the port list: see **Listing 1**.

```
module m #(parameter N = 1) ( input wire a);
// submodule a: instance a_1
```

Table 2. Operators in Verilog.

Logical operators	**Description**	**Example** using :reg [1:0] A = 2‘b01;reg [1:0] B = 2‘b10;reg [1:0] C = 2‘b00;
!	Logisches NOT	! A; // logic 0
&&	Logisches AND	A&&B; // “1 && 1“ = logic 1A&&C; // “1 && 0“ = logic 0
||	Logisches OR	A||C; // “1 || 0“ = logic 1
Bitwise operators		
~	Not	C&(~A);
&	And	A&B;
|	Or	A|C;
^	Xor	A^B;
Arithmetic operators		
*	multiplication	C= A*B // = 2‘b10
/	division	C= A/B // =2‘b00
+	addition	C = A+B // = 2‘b11
-	subtraction	C = B-A // = 2‘b01
**	power	
%	modulo	
Comparison operators		
>; <;	greater than; less than	if (A>B)||(A>C)
>=; <=;	greater than or equal to; less than or equal to	if (A>=B)||(A>=C)
==	equal to	if (A == B)
Other operator		
>>	right shift	A<<1 // A=2‘b10
<<	left shift	B>>1 // B=2‘b01
{}	concatenation	{A,B} // 4‘b0110
{{}}	replication	{2{A}} // 4‘b0101

```
a a_1(.a(a)); // instance a_1 of module a is created
// if the instantiated module a also had a parameter
   N, then
// the line would look like this: a #(.N(10))
   a_1(.a(a));
endmoDule
```

Around and around

An important keyword in Verilog is `always`. An always statement introduces a procedural block which, as its name indicates, will be called again and again. For synthesis, this block will describe a combination of combinatorial and sequential logic. A signal of type `reg` can only be assigned to within an always block. An always block can have a 'sensitivity list', which specifies that the statements given within the block should only be executed when a signal in the list has been assigned a new value since the last simulation time step when the block was executed. The always block is thus analogous to the 'process' in VHDL. The sensitivity list in Verilog is introduced by the `@` symbol: the symbol means that the signals named in the following bracketed list will be monitored to see when any of them change. The keyword `posedge` or `negedge` can be used if only one edge polarity on a signal is to be checked for.

When describing a sequential synchronous circuit the statement `always @ (posedge clk)` is used. All blocks in a design, whatever their level in the hierarchy, are connected to a common clock signal `clk`, and therefore all blocks are executed simultaneously. For beginners this is always the most difficult mental obstacle. Inside an always block is a sequence of 'procedural' assignments, which are processed one by one. There are two types of assignment, 'blocking' assignments (A = B;) and 'non-blocking' assignments (A <= B;). In the case of blocking assignments ('=') the assignment is executed completely before the next statement is processed. In the

Listing 1. A cloud of logic.

```
module logic_cloud_verilog
(
 input  wire  a,
 input  wire  b,
 input  wire  c,
 output reg   d,
 output reg   e
);

wire [2:0] abc;

// continuous assignment
// left-hand side operand of continuous assignment must be
// of wire type
assign  abc = {a,b,c};  // concatenation of a, b and c
using {}

// the * in the sensitivity list is a wildcard
// this means every signal which is in the block
// triggers the block
always @ (*) begin
 d = 1'b0; // this default assignment helps to avoid
    inferred latches
 if (abc == 3'b001) begin // begin..end is optional
  d = 1'b1;
 end else if (abc == 3'b101) begin
  d = 1'b1;
 end
end

always @ (*) begin // begin..end is optional
 case (abc)
  3'b000  : e = 1'b1;
  3'b011  : e = 1'b1;
  default : e = 1'b0;
 endcase
end

endmodule
```

Listing 2. D-type flip-flops with asynchronous reset and clock enable.

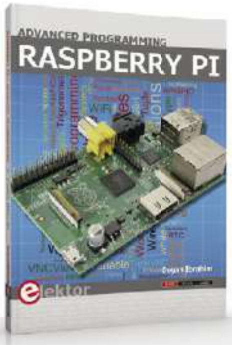
```
module d_ff_verilog
(
 input  wire clock,
 input  wire reset_n,
 input  wire ena,
 input  wire d,
 output reg  r_q,
 output reg  r_qn,
 output wire w_qn,
 output wire w_notOut

);

assign w_qn = ~r_q;      // continuous assignment -> this
                         //  will be a not buffer
not not_1(w_notOut,r_q); // usage of Verilog primitive

// Verilog 2001 permits comma separation in the
// sensitivity list
// always @ (posedge clock, negedge reset_n)
always @ (posedge clock or negedge reset_n) begin
 if (reset_n == 1'b0) begin
  r_q  <= 1'b0;          // reset value of output
  r_qn <= 1'b1;
 end else begin
  if (ena == 1'b1) begin // clock enable
   r_q  <= d;            // this translates to a first FF
   r_qn <= ~r_q;         // this translates to a second FF
                         // with inverted output
  end
 end
end
endmodule
```

case of non-blocking assignments ('<='), all the subsequent statements are processed before the assignment is finally committed. It is not always immediately apparent which type of assignment is wanted in each instance, and without an in-depth knowledge of the structure of the Verilog simulator, which is part of the language specification, cannot always be determined. Critics of Verilog often claim this is one of the weakest points of the language. In practice, however, it is not of great importance as there is a simple rule of thumb: when describing combinatorial logic, use blocking assignments ('='); and when describing sequential logic, use non-blocking assignments ('<='). If you stick to this rule then, apart from in a small number of exotic scenarios, you will avoid all the headaches. If you are still interested in knowing more, there is plenty of information about how the different types of assignments work in [1].

An example

Listing 1 shows a module which describes exclusively combinatorial logic. In accordance with our rule of thumb above all the assignments in this module are blocking ('='). The circuit does not have a particularly useful function, but it does employ a complete range of language constructs and is worthy of close examination. First we declare a three-bit vector of type `wire`. Then the elements of this vector are assigned values from inputs a, b and c using a continuous assignment. Since each of these input is a single-bit wire, before the assignment they are first assembled together using the {} concatenation operator. It is important to note that on the left-hand side of a continuous assignment there must be a wire type. In the first always block that follows we demonstrate the use of the `if-else` construct. Another rule of thumb, here to prevent the possible inferring of a latch, is that any variables that is assigned to inside a block must always be assigned to, regardless of which conditional paths are taken within the block. A simple alternative is to ensure that all variables used within the block are unconditionally assigned a default value at the start of the block: in this case the assignment statement is `d = 1'b0;`. The second always block gives an example using the `case` statement, which works in a similar way to the `switch` statement in C. The default branch at the end of the `case` block is optional, but it is good practice always to include it.

Listing 2 shows a description of a flip-flop with asynchronous reset and clock enable. In accordance with our rule of thumb above all assignments that describe sequential logic are non-blocking ('<='). In this example all the assignments are inside the always block, which describes a non-inverting and an inverting flip-flop. The non-inverting output of the flip-flop is also inverted using a 'primitive' at the beginning of the module description. Primitives are pre-defined Verilog modules, including NOT, AND, OR and XOR functions, that can be instantiated exactly like normal modules. In this case the module `not_1` is instantiated with its output connected to the wire `w_notOut` and its input to the `reg r_q`. The assignment `assign w_qn = ~r_q;` on the line immediately above has an exactly analogous inverting effect to the use of the primitive. **Figure 1** shows how the Quartus synthesis tool interprets



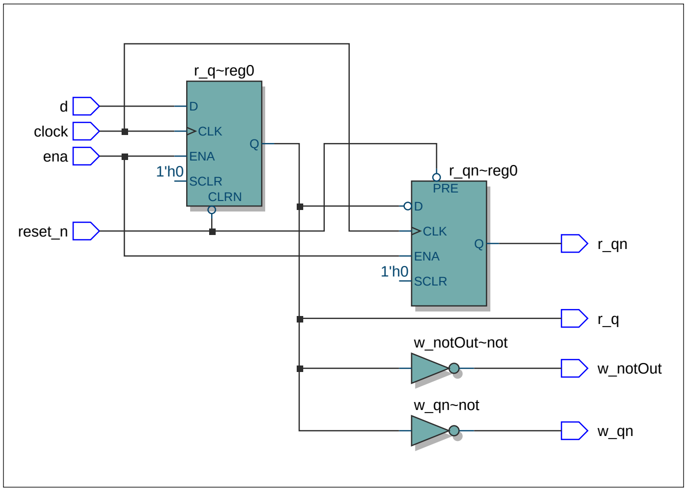


Figure 1. The flip-flop design in the RTL viewer.

the description in Listing 2. As expected, the two outputs w_notOut and w_qn are driven by inverting buffers, whereas the reg outputs are directly connected to the outputs of flip-flops. It can also be seen from the figure that the two signals in the sensitivity list are connected to the two edge-sensitive inputs of each flip-flop.

Simulation

In order to simulate a design in Verilog we use a module that instantiates the circuit to be verified as a ‘device under test’ (DUT) and stimulates it with an appropriate set of signals. This module is called a ‘testbench’. **Listing 3** shows one possible testbench for the flip-flop design of Listing 2. Inside the testbench, which in general will not be synthesizable, an always block is used to generate a clock signal. The symbol # is used in Verilog to introduce a time delay. The initial block used in the testbench has the same basic characteristics as an always block, with the distinction that it is only executed once, at the beginning of simulation. The system function $display can be used to send a text string to the simulator output: Verilog system functions are not synthesizable and start with a $.

Conclusion

Of course, there are many more functions in Verilog, and not all of them can be covered in a brief article. The examples we have looked at here should however give you a good starting point from which you can fearlessly explore the wider world of Verilog. Readers interested in delving deeper are encouraged to experiment with the wide range of Verilog (and VHDL) examples that can be found at EDA Playground [2] or to take a look at one of the many online tutorials available [3].

180562-02

Listing 3. Testbench for the flip-flop design.

```
`timescale 1ns/1ps
// timescale is a compiler directive: this means
// 1 time step equals 1 ns and the resolution is 1 ps
module tb_dff;

reg  clk;
reg  rst_n;
reg  r_d;
wire w_q;
wire w_qn;
wire w_qn_1;
wire w_qn_2;

// create 50 MHz clock
always begin
  #10; // delay for 10 time steps = 10 ns before inverting
          // clk again
  clk = ~clk;
end

d_ff_verilog DUT(
      .clock    (clk),
      .reset_n  (rst_n),
      .ena      (1'b1),
      .d        (r_d),
      .r_q      (w_q),
      .r_qn     (w_qn),
      .w_qn     (w_qn_1),
      .w_notOut (w_qn_2)
      );

initial begin
  $display(„Hello World „);
  clk   = 0;
  rst_n = 0;
  r_d     = 1'b0;
  #100;
  r_d = 1;
  rst_n = 1;
  #200;
  $stop();
end

endmodule
```

Web Links

[1] Sunburst Design: www.sunburst-design.com/papers/CummingsSNUG2000SJ_NBA.pdf

[2] EDA Playground: www.edaplayground.com/

[3] Nandland Verilog tutorials: www.nandland.com/verilog/tutorials/index.html

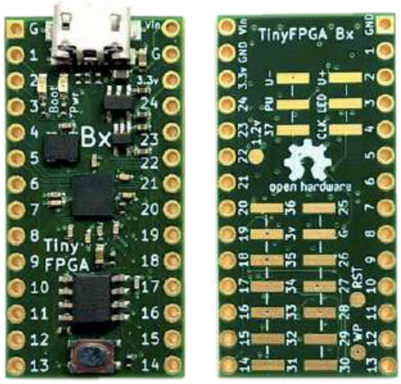
Elektor Store Highlights

What electronicists dream of...

...and how dreams come true in the Elektor Store

Getting started with FPGAs...

For most electronicists, grabbing a microcontroller is the most common thing in the world when it comes to building something more complicated than a blinker LED. In many cases, an FPGA (Field Programmable Gate Array) is an excellent alternative to a microcontroller. However, even 30 years or more on the market, FPGAs have some serious disadvantages including complexity and cost. Designing for FPGAs is difficult, and only works with expensive and complicated development tools. Also, the chips themselves are not exactly cheap — especially when compared to a powerful ARM processor that currently sells for less than one euro. Thanks to TinyFPGA, that can change. This company supplies small, cheap and easy to use FPGA breakout boards. Because TinyFPGA does not manufacture FPGA chips itself, these breakout boards are of course more expensive than an individual chip, but the prices remain within reasonable limits.



For the development of FPGA applications, TinyFPGA promotes free open source software (see below). The FPGA's used by TinyFPGA all originate from the chip baker Lattice Semiconductor. Currently, four breakout boards are available, with different FPGA chips. Here we limit ourselves to the BX model, with the iCE40LP8K on board. It has 7680 logic cells (lookup tables), 128 KB RAM and one PLL on the chip. To store a specific configuration, TinyFPGA employs an external SPI flash memory. A configuration (a 'program' in FPGA speak) is loaded into the chip in the form of a bitstream; if the chip does not have any internal non-volatile memory, that bitstream must be read again each time it is switched on — which entails the risk of hacking, something that companies hate intensely. Consequently these bitstream formats are proprietary. Nevertheless, the bitstream for the Lattice iCE-40 is 'documented' (i.e. hacked...) allowing the chip to be used with open-source development tools, in this specific case, Apio.

Far from claiming that FPGA programming is child's play now, with TinyFPGA and Apio the activity should come within reach of many more (hobby) electronicists.

...and thermal imaging cameras

With the increasing miniaturisation of electronic circuits and their power supply circuits, it is becoming increasingly important to monitor heat development. Sadly, that's easier said than done. You can't just hold a (fever) thermometer against an IC to check how's faring.

Thermal imaging cameras have been used in professional laboratories for ages, but these devices have a big disadvantage: they carry a professional price tag — hobbyists can only dream of them. With the HT-02 thermal imaging camera, that's over. This camera, which is produced in China (where else?), actually offers all the possibilities that similar high-end devices also have (for example, Flir or Fluke), but for a considerably friendlier price. How is that possible? Mainly by accepting a lower resolution. Just like more expensive ones, the HT-02 offers the possibility to superimpose an IR image on a 'normal' image for easier identification of hotspots, but the camera's lower resolution does not actually affect the 2.4-inch display much.

At 300 °C the temperature range of the camera is quite high (the nearest Flir camera in terms of price does not exceed 120 °C) and the accuracy with ±2 °C is large enough for normal use. Weighing no more than 320 g, the camera is easily held in your hand.

The HT-02 comes with a good manual and a sturdy cover. . ◀

180689-3

@ WWW.ELEKTOR.COM

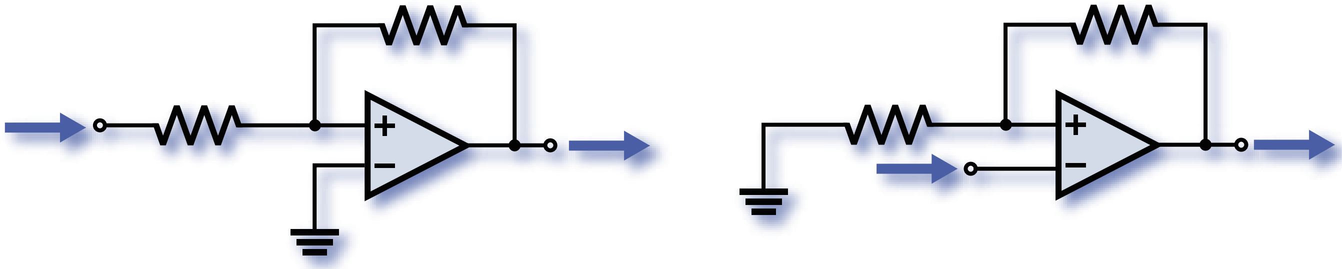
→ TinyFPGA BX
www.elektor.com/tinyfpga-bx

@ WWW.ELEKTOR.COM

→ Thermal Imaging Camera HT-02
www.elektor.com/handheld-thermal-imaging-camera-ht-02

Opamp Schmitt Triggers

Circuits and calculations



Schmitt triggers are standard circuits in electronics, so you might think there's nothing special about them. All you need is an opamp and a couple of resistors, right? That's true if you don't need precisely defined hysteresis and switching thresholds. But if you want to determine these values precisely and/or optimize them for standard resistance values, you need suitable circuits and the right formulas. Here we present both of these.

By **Volker Schmidt** (Germany)

Calculating component values for opamp Schmitt triggers is not all that easy if you do it only occasionally. One way to approach this calculation is to determine the reference voltage from the feedback resistance. However, in most cases the resulting reference voltage is not in the middle of the hysteresis window, and the calculation often yields odd values for which no Zener diodes or other reference sources are available. Then your only option for generating the reference voltage is a voltage divider with resistance values not found in any standard series, or a trimpot. A more practical approach would be to generate the reference voltage with a Zener diode, a reference voltage source or a simple voltage divider using resistance values from the standard E12 series, and then calculate the circuit from the reference voltage. That is not a straightforward task with standard circuits, but with a minor modification you can first specify the reference voltage and the switching thresholds and then calculate the component values. The necessary effort is reasonably small because only three additional components are required. This procedure can be used for both non-inverting and inverting Schmitt triggers.

Non-inverting Schmitt trigger

The switching characteristics of the non-inverting Schmitt trigger shown in **Figure 1** are determined by the voltage on resistor R2 and its output voltage range. The switching thresholds of the Schmitt trigger can be determined by adding or subtracting the hysteresis voltage ΔU_H to or from the voltage U_{R2}. The reference voltage (**Figure 2**) should ideally be in the middle between the upper switching threshold (U_{H+}) and the lower

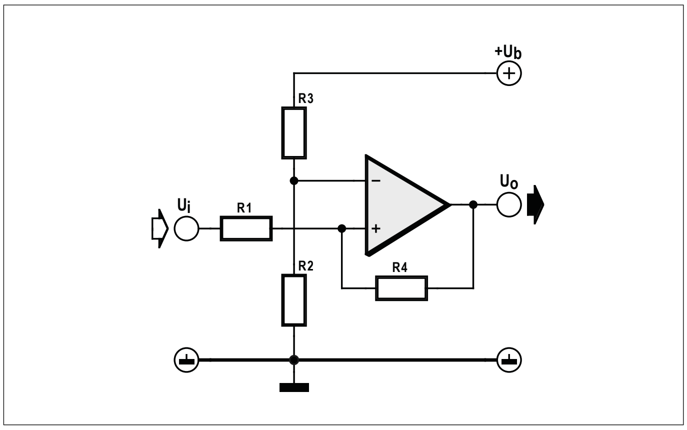


Figure 1. Standard circuit diagram of a non-inverting opamp Schmitt trigger.

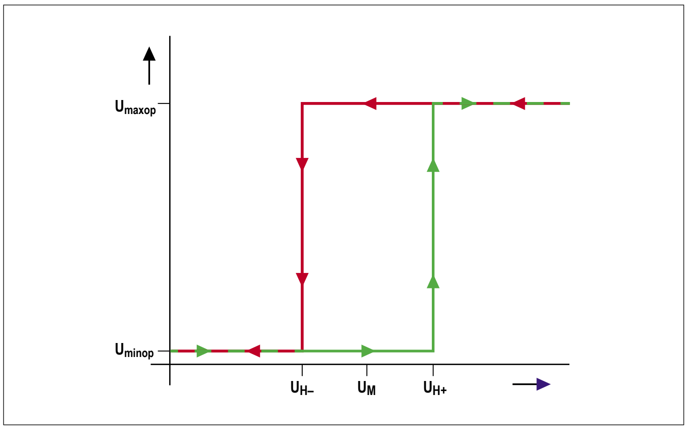


Figure 2. Hysteresis diagram of an ideal Schmitt trigger.

switching threshold (U_{H-}). However, this is not absolutely necessary. You should also ensure that the reference voltage is not too close to either of the switching thresholds. If you calculate the standard circuit in Figure 1 based on the reference voltage U_{R2}, you have the problem that one of the switching thresholds is not freely definable. The values of U_{H+} and U_{H-} are not symmetrical with respect to the reference voltage. In theory, they are only symmetrical when you use balanced supply voltages and tie the inverting input to ground, or you use a single supply voltage and tie the inverting input to exactly half the output voltage range of the opamp.

The reason for this behavior is the current flowing through R4 and R1. It depends on the difference between the output voltage and the voltage on the inverting input. The following relationships are true only if the magnitude of this current is the same in both the on and off states:

$$U_{H+} = U_{R2} + \Delta U_{H+}$$

$$U_{H-} = U_{R2} - \Delta U_{H-}$$

$$|\Delta U_{H+}| = |\Delta U_{H-}|$$

$$I_{R4} = \frac{U_{maxop} - U_{R2}}{R4} = -\frac{U_{minop} - U_{R2}}{R4}$$

$$I_{R1} = -I_{R4}$$

$$\Delta U_H = -I_{R4} \times R1$$

Otherwise it is still possible to specify one of the switching thresholds (for example, U_{H+}), but the other one is outside your direct control. In that case the value of I_{R4} is not the same in both switch states. This can be clearly seen from a calculated example using a type CA3140 opamp in the circuit shown in Figure 1. The initial parameters are:

$U_m = U_{R2} = 1.7$ V	$U_B = 9$ V	$\Delta U_{H+} = 0.2$ V
$\Delta U_{H+} = 1.9$ V	$U_{maxop} = 6.8$ V (measured)	
$U_{H-} = 1.5$ V (desired)	$U_{maxop} = 0.01$ V (measured)	

Now you have to calculate the **voltage divider** R2/R3:

$$\frac{R2}{R3} = \frac{U_{R2}}{U_{R3}}$$

$$R2 = R3 \times \frac{U_{R2}}{U_{R3}}$$

If you choose 10 kΩ for R3, then

$$R2 = 10k\Omega \times \frac{1.7V}{7.3V} = 2.33k\Omega$$

This value can be approximated closely enough with 2.2 kΩ + 100 Ω. Since the voltages at the inputs of the opamp are the same at the switching points, you also have

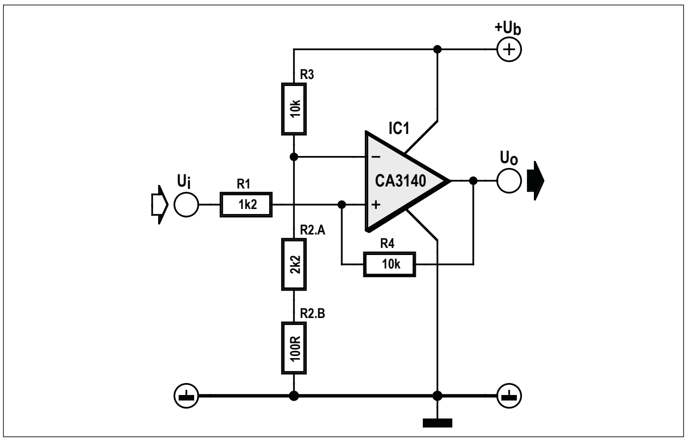


Figure 3. Fully dimensioned circuit of the standard non-inverting Schmitt trigger.

$$U_p = U_n$$

$$U_{R4} = U_{minop} - U_{R2}$$

$$U_{R4} = 0.01V - 1.7V = -1.69V$$

IF R4 is 10 kΩ, then

$$I_{R4} = \frac{U_{R4}}{R4} = \frac{-1.69V}{10\,k\Omega} = -169\mu A$$

and

$$I_{R1} = -I_{R4}$$

The value of R1 is therefore given by the formula

$$R1 = \frac{\Delta U_{H+}}{I_{R1}} = \frac{0.2V}{169\mu A} = 1.183\,k\Omega$$

A value of 1.2 kΩ is a good approximation for R1.

Now you have to determine the **lower switching threshold** U_{H-}:

$$U_{R4} = U_{maxop} - U_{R2}$$

$$U_{R4} = 6.8V - 1.7V = 5.1V$$

$$I_{R4} = \frac{U_{R4}}{R4} = \frac{5.1V}{10\,k\Omega} = 510\,\mu A$$

$$R1 = \frac{\Delta U_{H-}}{I_{R1}}$$

$$\Delta U_{H-} = R1 \times I_{R1}$$

$$I_{R1} = -I_{R4}$$

$$\Delta U_{H-} = 1.2\,k\Omega \times -510\,\mu A = -0.612V$$

$$U_{H-} = U_{R2} - \Delta U_{H-} = 1.088V$$

The actual lower switching threshold differs significantly from the desired value.

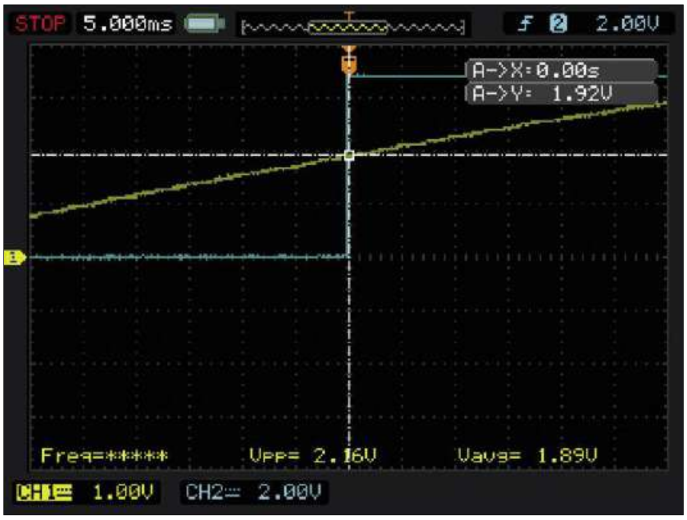


Figure 4. Oscilloscope screenshot of the upper switching threshold of the standard non-inverting circuit.

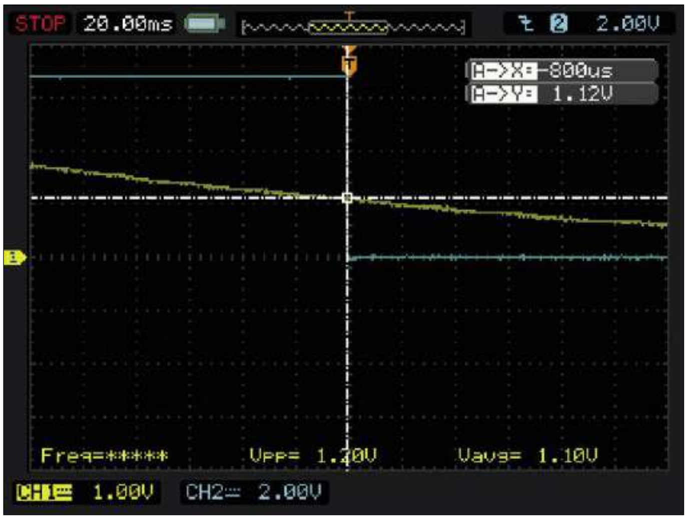


Figure 5. Oscilloscope screenshot of the lower switching threshold of the standard non-inverting circuit.

Figure 3 shows the fully dimensioned standard circuit. The math and the two oscilloscope screenshots (**Figure 4** and **Figure 5**) show that the switching thresholds are theoretically and practically not symmetrical with respect to the reference voltage.

Modified circuit

The situation can be remedied by providing separate feedback paths for the upper and lower switching thresholds. Then both thresholds can be freely defined by choosing suitable resistor values. For this you must add two feedback resistors and two diodes to the basic circuit, resulting in the circuit shown in **Figure 6**. It is a good idea to use Schottky diodes (for example, type 1H5817) due to their low voltage drop. Resistor R4 is for the upper switching threshold, while R5 is for the lower switching threshold. Now the hysteresis can be very flexibly adapted to specific circumstances.

The following parameters apply to the calculation of this circuit (here again using a CA3140 opamp):

U_m = U_{R2} = 1.7 V	U_B = 9 V	U_{FD} = 0.21 V (measured)
ΔU_{H+} = 1.9 V	U_{maxop} = 6.8 V (measured)	R_3 = 10 kΩ
U_{H-} = 1.5 V	U_{minop} = 0.01 V (measured)	ΔU_H = 0.2 V

The **voltage divider** R2/R3 is calculated as

$$R2 = R3 \times \frac{U_{R2}}{U_{R3}} = 10\,k\Omega \times \frac{1.7V}{7.3V} = 2.33\,k\Omega$$

Here again R2 can consist of a 2.2-kΩ resistor in series with a 100-Ω resistor. The **upper switching threshold** is calculated as

$$U_{R4} = U_{minop} + U_{FD} - U_{R2} = 0.01V + 0.21V - 1.7V = -1.48V$$

With a value of 10 kΩ for R4, the current I_{R4} is −148 μA. Since the voltages at the inputs of the opamp are the same at the switching thresholds, you have

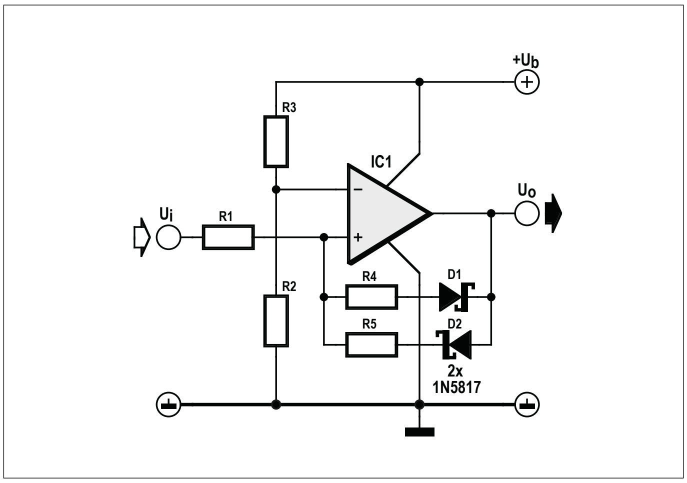


Figure 6. Basic circuit diagram of the modified non-inverting Schmitt trigger.

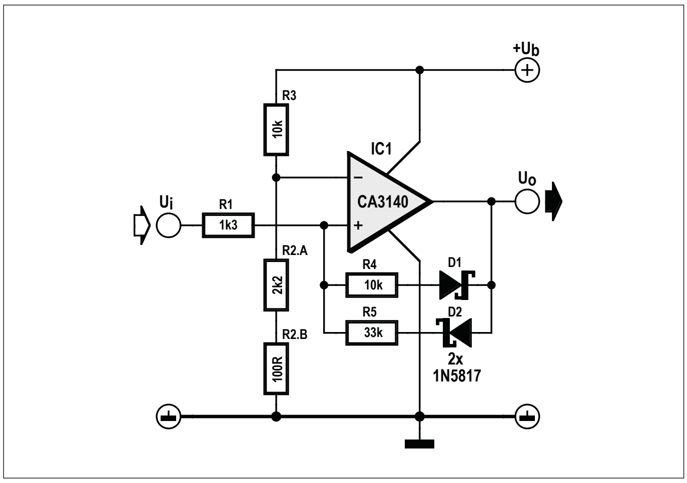


Figure 7. Fully dimensioned circuit of the modified non-inverting Schmitt trigger.

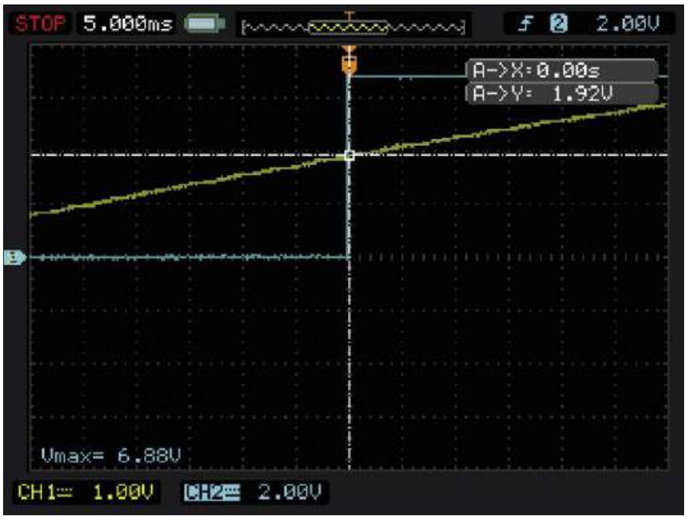


Figure 8. Oscilloscope screenshot of the upper switching threshold of the modified non-inverting circuit.

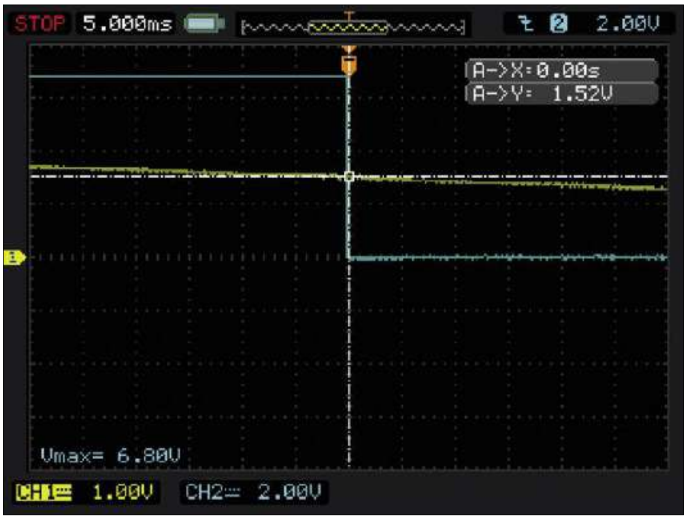


Figure 9. Oscilloscope screenshot of the lower switching threshold of the modified non-inverting circuit.

$$U_p = U_n$$

which means that

$$I_{R1} = -I_{R4}$$

The potential at the non-inverting input must be higher by the amount

$$\Delta U_{H+} = U_{H+} - U_{R2}$$

to cause the Schmitt trigger to switch and change the output to U_{maxop}.

The value of R1 is now given by the formula

$$R1 = \frac{\Delta U_{H+}}{I_{R1}} = \frac{0.2V}{148\,\mu A} = 1.351\,k\Omega$$

You can use a standard E24 value of 1.3 kΩ for R1 with only a slight error.
For the **lower switching threshold** you have

$$I_{R5} = \frac{U_{maxop} - U_{FD} - U_{R2}}{R5} = 148\,\mu A \rightarrow upper\ switching\ threshold!$$

$$R5 = \frac{U_{maxop} - U_{FD} - U_{R2}}{I_{R5}} = \frac{6.8V - 0.21V - 1.7V}{148\,\mu A} = 33.04\,k\Omega$$

The E12 value of 33 kΩ is a perfect fit for R5.

Figure 7 shows the fully dimensioned circuit. The two oscilloscope screenshots (**Figure 8** and **Figure 9**) show that the actual switching thresholds are very close to the theoretical values.

Inverting Schmitt trigger

With the inverting Schmitt trigger it is also possible to calculate the switching thresholds based on the reference voltage. As with the non-inverting version, a small modification to the standard circuit in **Figure 10** is also possible, leading to the version shown in **Figure 11**. Here again, the two thresholds are decoupled by using two resistors and two Schottky diodes. However, in this case these components are locate in the link

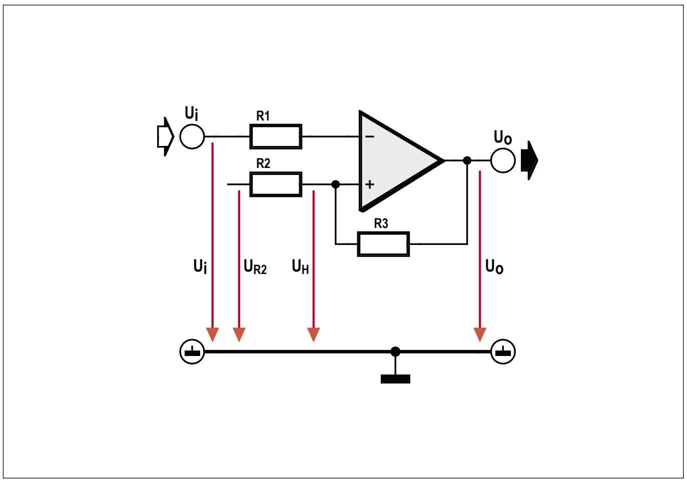


Figure 10. Circuit diagram of the standard inverting Schmitt trigger.

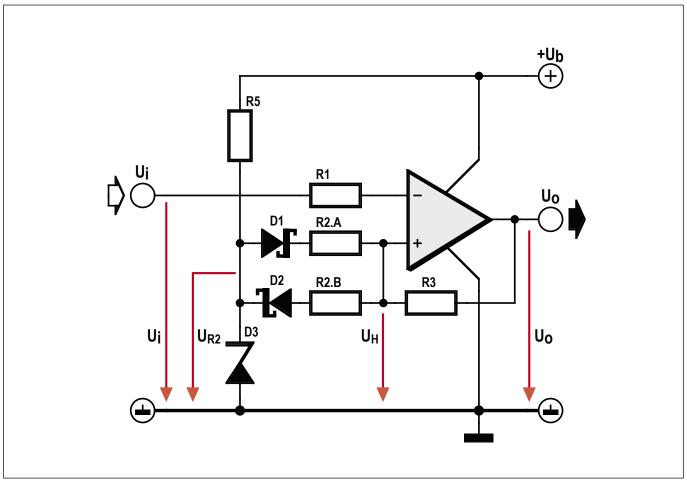


Figure 11. Basic circuit diagram of a modified inverting Schmitt trigger.

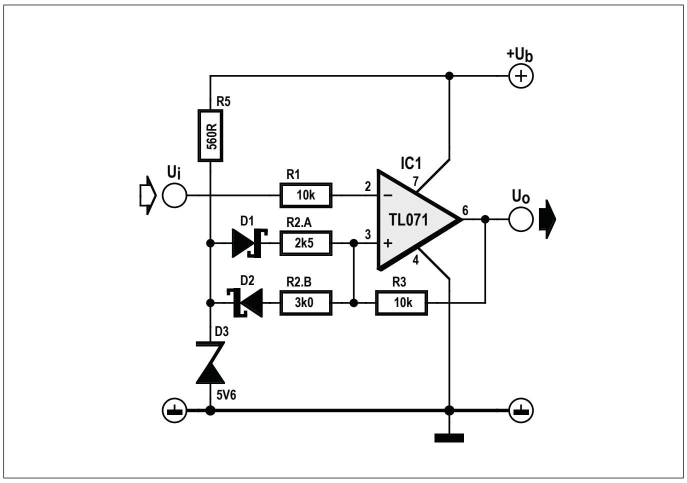


Figure 12. Fully dimensioned circuit of the modified inverting Schmitt trigger.

to the reference voltage instead of the feedback path. Splitting R2 into R2.A and R2.B puts the right comparison voltage on the non-inverting input of the opamp for each state of the Schmitt trigger. First you choose a value for resistor R3, and then you calculate the values of R2.A and R2.B from that value. For the **lower switching threshold** the relevant equations are $U_a = U_{minop}$ and $U_H = U_{H-}$, as well as the following formulas:

$$\frac{U_{H-} - U_{minop}}{R3} = \frac{U_{R2} - U_{FD} - U_{H-}}{R2.A}$$

$$R2.A = R3 \times \frac{U_{R2} - U_{FD} - U_{H-}}{U_{H-} - U_{minop}}$$

For the **upper switching threshold** you can use the equations $U_a = U_{masop}$ and $U_H = U_{H+}$.

$$\frac{U_{maxop} - U_{H+}}{R3} = \frac{U_{H+} - U_{R2} - U_{FD}}{R2.B}$$

$$R2.B = R3 \times \frac{U_{H+} - U_{R2} - U_{FD}}{U_{maxop} - U_{H+}}$$

Now let's dimension the circuit for a specific case and use it as example to check how well it works. The following parameters apply if you use a TL071 opamp:

$U_m = U_{R2} = 5.6$ V	$U_B = 9$ V	$R_3 = 10$ kΩ
$\Delta U_{H+} = 6.6$ V	$U_{maxop} = 8.2$ V (measured)	$U_{FD} = 0.21$ V (measured)
$U_{H-} = 4.6$ V	$U_{maxop} = 1.48$ V (measured)	

The following formula applies to the **lower switching threshold**:

$$R2.A = R3 \times \frac{U_{R2} - U_{FD} - U_{H-}}{U_{H-} - U_{minop}} =$$

$$\frac{5.6\,V - 0.21\,V - 4.6\,V}{4.6\,V - 1.48\,V} \times 10\,k\Omega = 2.532\,k\Omega$$

R2.A can be conveniently composed of 1.5 kΩ and 1 kΩ in series.

For the upper switching threshold you have

$$R2.B = R3 \times \frac{U_{H+} - U_{R2} - U_{FD}}{U_{maxop} - U_{H+}} =$$

$$\frac{6.6\,V - 5.6\,V - 0.21\,V}{8.2\,V - 5.6\,V} \times 10\,k\Omega = 3.038\,k\Omega$$

If you use 1.8 kΩ and 1.2 kΩ in series for R2.B, the resulting error is about 1.3%.

Figure 12 shows the fully dimensioned circuit. The oscilloscope screenshots (**Figure 13** and **Figure 14**) show that the actual thresholds are close to the calculated values.

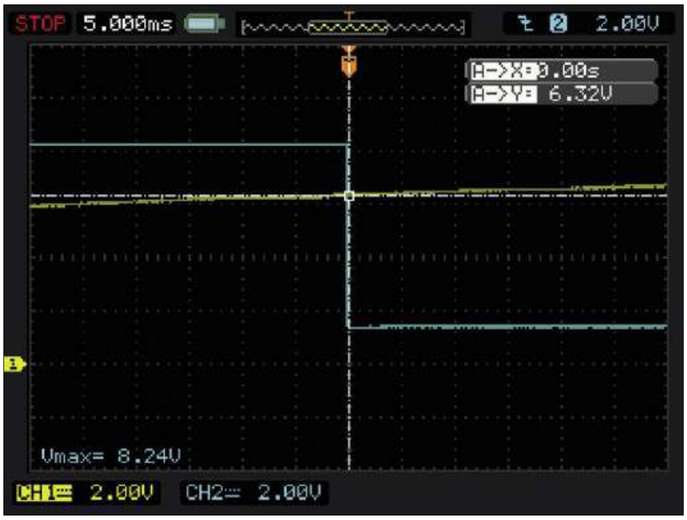


Figure 13. Oscilloscope screenshot of the upper switching threshold of the modified inverting circuit. The output signal curve (blue-green line) is also inverted.

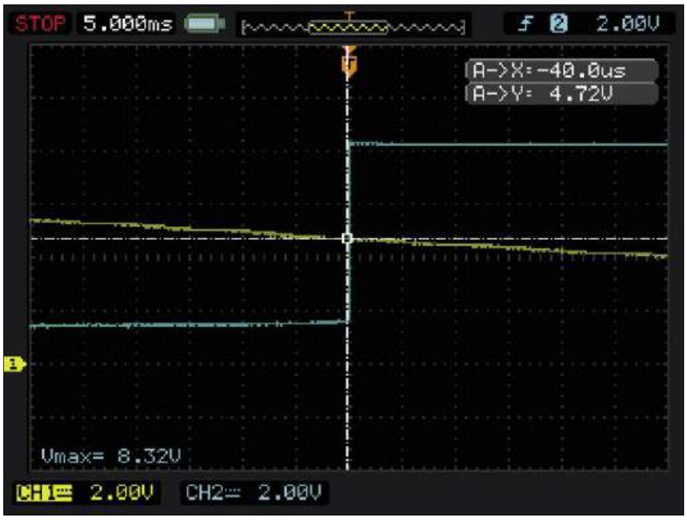


Figure 14. Oscilloscope screenshot of the lower switching threshold of the modified inverting circuit.

Key to parameter symbols

Parameter	Meaning
U_b	Supply voltage
U_{maxop}	Maximum opamp output voltage
U_{minop}	Minimum opamp output voltage
U_H	General switching level
U_{H+}	Upper switching threshold
U_{H-}	Lower switching threshold
U_M	Mean switching threshold
Δ_{UH+}	Positive hysteresis
$\Delta_{UH}-$	Negative hysteresis
U_{R2}	Reference voltage
U_{FD}	Schottky diode forward voltage

The values of U_{minop}, U_{maxop} and U_{FD} should be determined by actual measurement because they are crucial for accuracy. The values given in datasheets for opamps and diodes often have a large tolerance range.

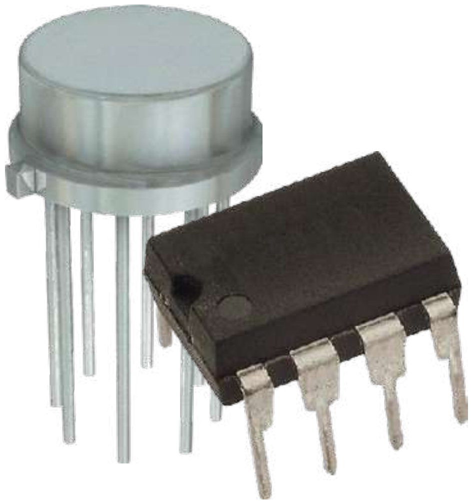
Summary

Even if this all sounds very abstract, the next time you need to calculate component values for a Schmitt trigger you will see that the two circuits shown in Figures 7 and 12 can be dimensioned very quickly using the associated formulas. However, the expense is only justified when precise threshold values are required. Otherwise the standard circuit shown in Figure 3 is sufficient. The formulas for it are also given in this article. Happy designing! ⏮

(160340-I)

About the author

After working for many years as a project engineer for remote control systems and programmable logic controllers, Volker Schmidt is now a self-employed IT specialist. He has been an electronics enthusiast since the age of 12. He mainly works with AVR microcontrollers now, but also with analog circuits.



Old School AC Powerline Conditioning & Regulation

How not to blow a fuse

By **Jan Buiting**, Resident Editor, Elektor Retronics

Just because it's so stable and reliable in this day and age, the AC voltage on the power outlets in our electronics workspaces is often taken for granted. It's not until you get a jolt, a fuse blows, or equipment starts to behave capriciously, that we surmise trouble at the 230 VAC side. That's totally different when working with vintage electronic equipment, especially precious gear if it's powered up for the first time… then "the mains" is of foremost and even deadly concern. In this article we show some options to avoid killing our darlings before they even wake up from years of slumber and neglect. And protect ourselves.

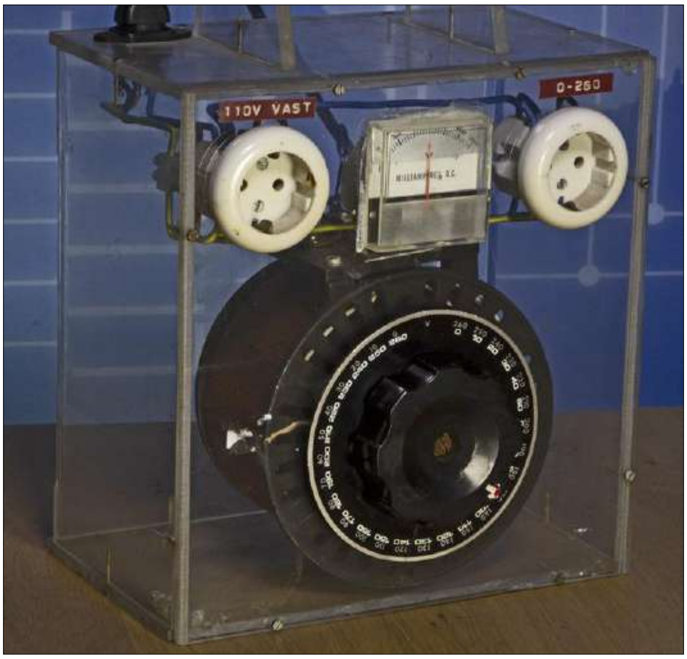


Figure 1. The Elektor Labs 850-VA variac is now entrusted to the *Fingerspitzen* at Retronics.

Welcome to another instalment of Retronics, the longest standing regular feature in the history of Elektor magazine. This time it's all about AC mains power and how we (gingerly) apply it to vintage equipment, or distribute it safely in the e-workspace or lab even when working with modern gear only. What options are available?

Variac

Variac was a U.S. trademark of General Radio for their "variable autotransformer" products. A variac offers a reliable way of raising the AC line voltage from 10 VAC or so up to the "nominal value" and even slightly higher, i.e. from 10 to 250 VAC. The basic variac is an autotransformer operated by hand using a large knob, and with a dial scale to indicate the approximate output voltage. Inside, the knob operates a carbon slider contact that can be moved to any position between the extremes of the autotransformer's winding made from copper wire. The relative position of the slider determines the output voltage, like a wiper in a potentiometer determines the resistance. Some large variac have motor control, these devices were usually intended for remote control with the obvious benefit of not having a bulky, extremely heavy and forever humming thing on or under your desk.

Variacs represent inductances of tens of henries and may produce an odd sound (like short whistle) when switched on, even with no load. Or blow the domestic fuse.

Being an autotransformer, a variac has only one winding which although "tapped" does not isolate the load from the AC grid. This is a common misunderstanding with beginners, believing that 10 or 40 volts AC or so from a variac means 'low voltage = safe voltage'. That is correct on the adjective 'low' but the full Live (L) voltage may exist on a terminal, assuming the power outlet is not polarised (as it is I Holland).

The variac shown in **Figure 1** has been a faithful companion in the Elektor Labs for years but was decommissioned due to lack of use. It's a 4-amp type i.e. for a good 850 VA. I remember it was used frequently on Elektor's high-power audio power amplifier projects of the old days, as a means of carefully ramping up their power supplies and avoid blown fuses and general mayhem in the lab. Today it has found good use again with any piece of Retronics equipment that needs a slow and careful first start — mostly tube based equipment with the inherent danger of exploding high-voltage electrolytics.

Originally a 220-volt type from the 1970s, the variac has a scale that's somewhat off with today's 230-VAC nominally on the power outlets. The voltmeter on it is a poor show, I trust it till about 150 VAC or so but above that a Fluke DVM is used. Otherwise that variac is an invaluable instrument to me, and the first "tool" to bring into play when fresh vintage equipment has arrived. For electrical safety and to impress folks working on lowly DC voltages, the Elektor Retronics variac is encased in a Plexiglas™ (acrylic sheet) housing (**Figures 2** and **3**) and has a carrying handle made from PVC electrical wiring conduit! Younger people find it heavy at about 12 kilograms, and the slider sparks a little when loaded with 800 watts.

In the case of SMPSUs and some 110-240 VAC auto-adjusting supplies, a variac is not recommended. I once damaged an oscilloscope supply of the inverter/oscillator type by using the variac to slowly ramp up the AC supply voltage. Too slowly, as the supply expects the nominal level to appear within a certain period. Way under this level (like 150 of 220 VAC), oscillation does not start and excessive and prolonged DC flows into certain transformer windings with dire results. A variac is not recommended either for heavily inductive or capacitive loads as resonance may occur. Or a blown domestic fuse.

Variacs can be picked up at ridiculously low prices these days as most just gather dust or act as doorstops. I say, grab one when you can.

Mains stabiliser, anyone?

In the old days, in buildings where heavy electrical machinery and "office electronics" was cheerfully operated off the same heavy-duty AC supply cable, voltage surges, dips, outages and sheer interference were a constant source of trouble to … the office workers. Some suppliers of telephone equipment even refused to install exchanges and other "sensitive" equipment if the mains supply was found to be flaky. The same for office lighting, calculators and the odd computer or server. And consider the extra voltage losses in the US where the AC powerline voltage was just 110-117 VAC compared to 220 VAC across most of Europe. In labs and factories, a solution was found by installing mains stabilisers at carefully selected points, both electrically and acoustically! One such device is pictured in **Figure 4,** in operation

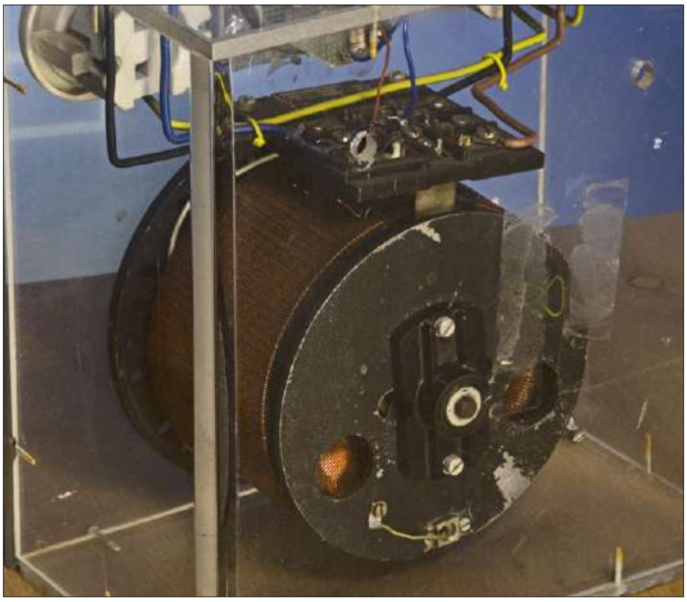


Figure 2. Back side of the 'portable' variac in its transparent Plexiglas case. The slider contact ('wiper') is not visible; it is between the coil and the round cover plate.

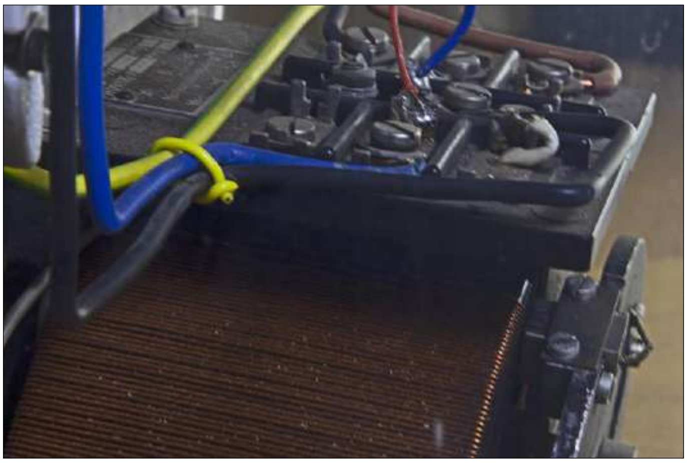


Figure 3. There's a good amount of precious copper wire lurking inside a variac. Here you can see the screw terminal plate and the autotransformer turns with light dust on the copper wire.

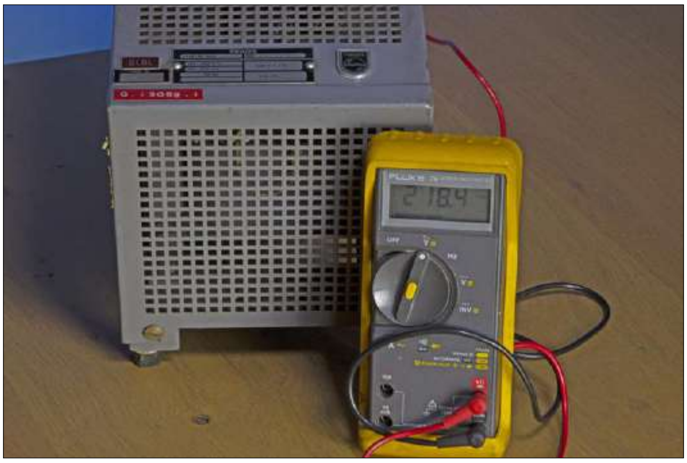


Figure 4. Thirty years or more of neglect have not affected the PE1033's passively maintained stability: the output voltage is still stable within 1% of the specified 220 VAC. Input voltage: 232 VAC.

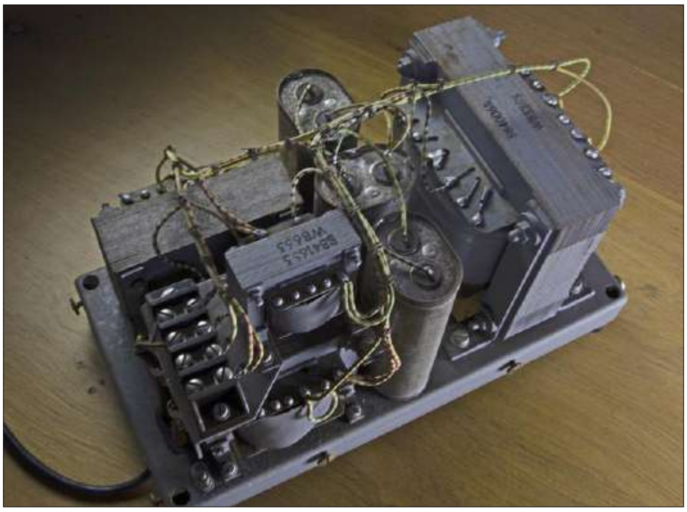


Figure 5. The PE1033 mains stabiliser consists of nothing but whopping transformers and capacitors.

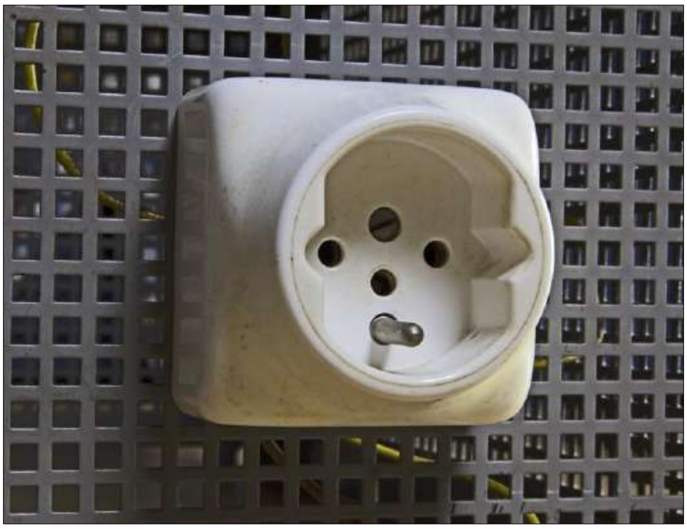


Figure 6. Someone has secured a porcelain power socket to the PE1033 casing. That suggests portability of the instrument, which is false, the thing is for stationary use.

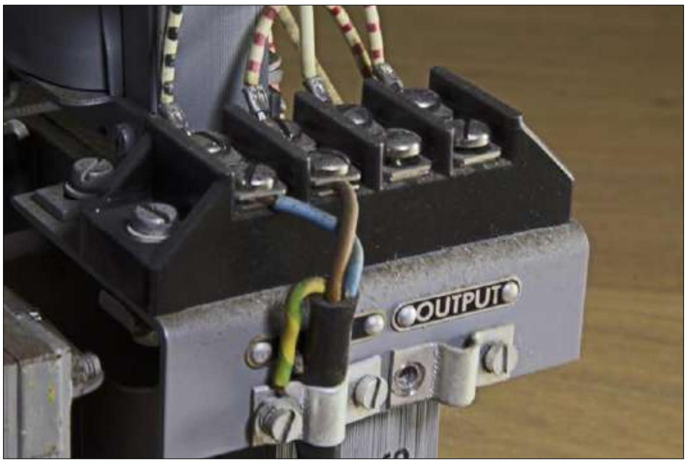


Figure 7. The Philips PE1033 mains stabiliser from about 1960 was designed for permanent wiring into 220 VAC electrical systems in labs, factories and offices, and installation in an out if the way place.

within spec as ascertained by the Fluke DMM reading 218.4 VAC! The grey thing is a Philips PE1033 from the mid-1960s, rated at:

- Output voltage: 220 VAC ±1%
- Output power: 250 VA
- Input voltage: 187–242 VAC
- Input power: 315–335 VA
- Frequency: 50 Hz

Other than what I was able to copy from the equipment identification tag I have nothing on this device in terms of schematics. Inside, there are four transformers, three tall capacitors (5 µF / 500 V) and a wire dress, no more (**Figure 5**). The PE1033 weighs 25 kgs yet is fairly compact at 32 x 18 x 17 cms, which makes it a pain to move around without hurting your fingers. It rests on 4 rubber feet and that's not for electrical safety only, as you discover when the thing is switched on. No thump, but a constant hum that would drive people crazy and not just 1960s office workers or the telephone exchange lady. From what I read on the web the hum is due to the transformers being operated at high iron losses on purpose to maintain resonance at 50 Hz in concert with the capacitors. How that produces a 1% stabilising effect I've yet to discover, but the system works a treat to suppress anything riding on the mains lines that's not 50 Hz. I discovered that by powering the PE1033 from my domestic grid (230-235 VAC mostly) and examining the output waveform. Compared to what went in, the output was as clean as a whistle. At the time of the measurement the grid voltage available on my workbench was heavily infested by bursts and surges from my 7-kWp PV installation outputting just 4.8 kWp on a partly cloudy afternoon.

Unthinkable these days is the power loss introduced by this type of mains stabiliser — at a load of 105 watts (A Tektronix 310A 'scope) I measured about 280 watts of input power. Stability and cleanliness of the AC supply come at a cost… and a lot of hum and heat 24/7!

My PE1033 has a Belgian, possibly French style power socket mounted in an improvised manner at the rear side of the cabinet (**Figure 6**). It is connected to the PE1033's OUTPUT terminals (L and N) and the chassis (GND) by screws. Both the INPUT and OUTPUT terminal pairs are inside the cabinet and intended for connecting permanent wiring (**Figure 7**).

Stabilising, filtering and humming the PE1033 may excel at — it still does not afford the much coveted full electrical isolation from the mains. Besides it has no adjustment range to speak of. For these and other desiderata, we have to look elsewhere.

Adjustable transformer — no jolts from the TV chassis

So on our workbench we want at least one outlet powered from an AC source that's both adjustable across a large range (like 10–260 V from a variac) and double-isolated from the mains, i.e. having a real transformer with a primary and a secondary winding, rated for very high inter-winding isolation voltage (say 2 kV). And fused… and quiet.

A device meeting these two requirements and more exists. Recently one was given to me by an old radio & TV repairman. Clearly, he had been concerned about his health and back in the 1970s convinced his boss to install a Grundig RT 5A for him (**Figure 8**). This is the *nec plus ultra* as far as electrical safety and versatility went in one instrument in the early 1970s. But it's again heavy at 18 kgs.

Germany's Grundig was a major manufacturer of radio and TV sets and not surprisingly the RT series (where RT is: *Regel-Trenn-Transformator*) was often seen in RTV repair shops. Mine is a 3.5-amp (say 800-VA) type with separate volts and amps readouts on moving-coil meters.

The symbols on the RT 5A's front panel (**Figure 9**) are important and reflect German *Gründlichkeit*: double isolation class, fused, fully separated transformer windings, on/off switch.

Then main reason for discussions on the RT 5A to appear at all on the web is that it tends to blow your domestic fuse every time it is switched it on. Today's fuses in 230 VAC domestic installations are mostly high or medium speed semi-automatic, with a faster response than the 16-amp porcelain/wire fuses used in the 1970s. My RT 5A is no exception and even if it's 10-amps slow-fused itself on the front panel, I need to consider installing a slow start add-on that connects a big stopper resistor (like 10 ohms, 25 watts) in series with the transformer primary. The resistor will limit the inrush current the first 500 ms or so after switching on, and then gets shunted by a relay contact. Leaving the RT 5A on 24/7/365 is no option as it hums and wastes a lot of energy.

If you open the RT 5A, the electronics are okay in terms of assembly methods, but not the mechanical construction, which is far too weak for a 15+ kgs transformer. In a bid to keep the weight of the instrument within limits, the designers at Grundig skimped on the sheet steel used — it is way too thin for the purpose.

Conclusions

Each of the three instruments discussed has its own merits and shortcomings, all depending on what you want to do with them, and where.

The variac is great for quick use outside the workspace as it is quasi-portable. Its power rating of about 800 watts I found adequate for most of the unknown equipment I want to start up in a controlled manner over a period of six hours or so. Its fixed 115 VAC output is also handy for US equipment; sadly the variac lacks an ammeter to indicate trouble before the first smoke appears.

The PE1033 is a dinosaur which really excels at filtering of the mains voltage only. Its intended purpose, output voltage stabilisation within 1%, was primordial in its period but has limited use today except as a benchmark or known-good-boatanchor (KGB).

The RT 5A is the preferred instrument to use, offering full electrical isolation, and adjustability from 0 to 250 VAC. Given the ramshackle case it's not as suitable to go 'portable' with you like the variac is. For use today the RT 5A requires a soft start itself!

I am in the fortuitous position to have all three in the Retronics Equipment Collection, for use as governed by the situation, and obtained for scrap iron prices. No blown fuses! ⏮

180574-01

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

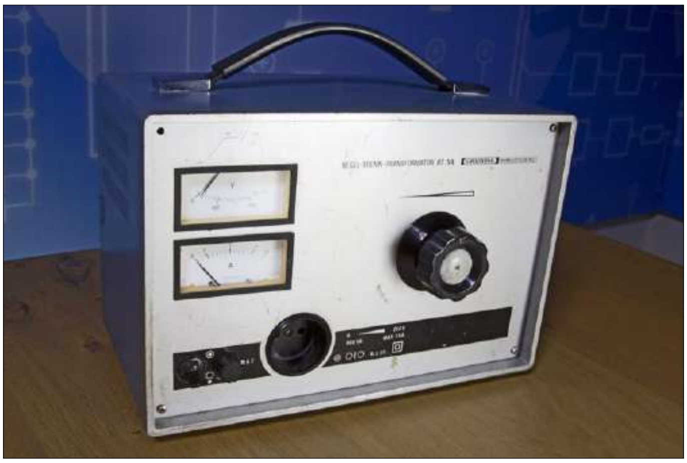


Figure 8. The Grundig Electronic type RT 5A from about 1970 combines a variac with an adjustable transformer, meaning its output is isolated from the mains potential. It's great for repairs on old TV sets and radios where the chassis can be at Live (L) potential. Note the power socket on the front panel is not earthed and that's both wise and desirable.

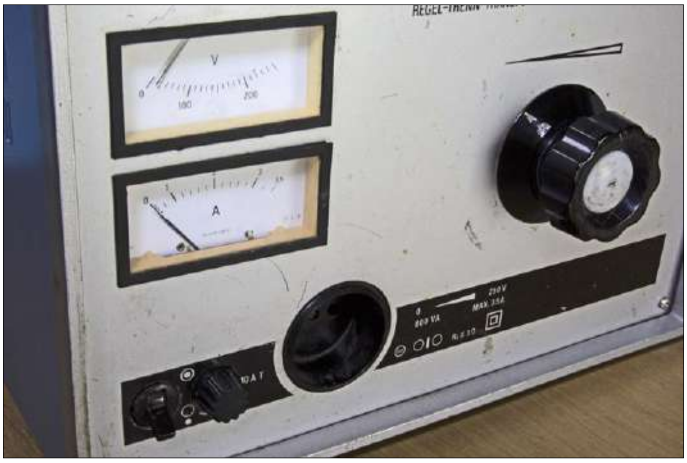


Figure 9. Who needs a user manual or circuit diagram when everything you have to know on the Grundig RT 5A variac/isolator is printed as symbols on the front panel?

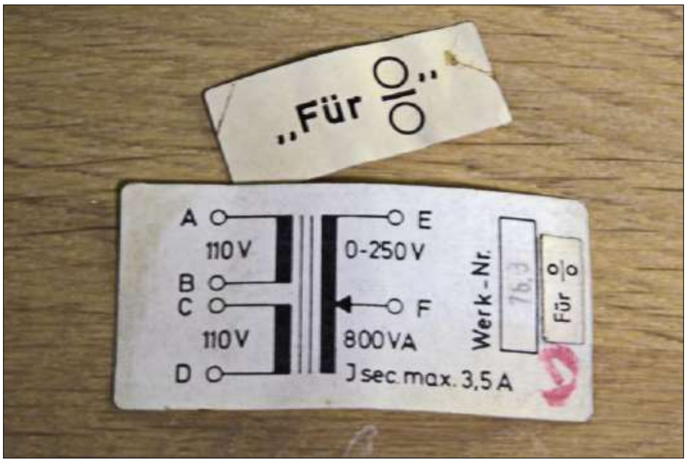


Figure 10. Two small adhesives that dropped out of the RT 5A when I opened the case adequately summarise the electrical composition of the instrument.

The Interplanetary File System

Behind the scenes work is underway on a new worldwide web

By Tessel Renzenbrink (The Netherlands)

An Earthly explorer is currently crawling around on Mars – NASA's Curiosity is part of the robot vanguard that investigates the red planet. The goal is that one day these robots will be joined by people. And as soon as the first interplanetary pioneers establish themselves on Mars, they too will want to be able to browse Wikipedia.

Source: NASA

However… with the current structure of the global web, they will inevitably have to deal with the problem of latency. A Martian web browser takes at least 4 minutes to connect to the Wikipedia server on earth. And depending on the position of the planets, the latency can be up to 24 minutes.

The Interplanetary File System (IPFS) wants to solve this by basing the web on a new protocol. IPFS does not only solve a hypothetical future problem. The latency problem already applies today to people living far away from data centres or with slow connections such as in rural areas and developing countries.

IPFS is not only faster than the current web. It also solves other issues that currently plague the web, such as censorship, privacy violations and Distributed Denial of Service (DDoS) attacks. That sounds almost too good to be true. To understand what this claim is based on, it makes sense to look at what is wrong with the current web.

The ailments of the global web

At present, the web is a highly centralised system (**Figure 1a**). In a centralised computer network, there is one central node. All communication between computers in the network operates via that central node. The weakness of this configuration is immediately visible. The network is vulnerable because it can no longer function if the central node fails.

Privacy violations

In addition to the vulnerability of the system, centralisation also leads to an unequal balance of power. The central node is master of the information streams. It can refuse to pass on data, read data itself or charge tolls on the transit of data. This "power" can lead to breaches of privacy. For example, we are en masse on the servers of Facebook and Google. The tech giants position themselves between the content on the web and the people who want to access that content. Think of Youtube, for example. Google does not make videos itself; millions of people worldwide do so.

That was also the original idea behind the global web: people who make content and information available to others via the web. Google and Facebook have manoeuvred themselves as the central point between the content and the people. The toll visitors pay consists of their personal data.

Censorship

The centralisation of the web is a consequence of the client-server model on which it is based (**Figure 2**). For example, if you want to visit Wikipedia, you must contact a server that hosts the website. If access to that server is blocked, the information is no longer available to the clients. This happened, for example, in 2017 in Turkey, when the government banned Wikipedia nationwide.

DDoS attacks

The client-server model is also the reason why DDoS attacks work. Simply put, a DDoS attack is a colossal number of information requests to one server. That server is then flooded with requests. This means that people with legitimate information requests can no longer reach the server. Again, the web is vulnerable because information can only be retrieved from one central location.

The promise of the IPFS

As said, IPFS comes with big promises. It is faster, censorship resistant and gives us control over our data. This is because IPFS is not a centralised but a distributed network. A distributed network (Figure 1c) is by far the most resilient network configuration. Every computer in the network acts as a node over which information can flow. If a number of computers fail, the rest of the network can continue to function. In a distributed network there is no hierarchy: all nodes in the network are the same.

The distributed IPFS network is not based on the client-server model for exchanging data. Instead, files are shared peer-to-peer. Consequently, every computer in the network can host a file. If you want to download that file, ask for a copy from the nearest hosts. Then the downloader itself can also become a host and help spread the file further (**Figure 3**).

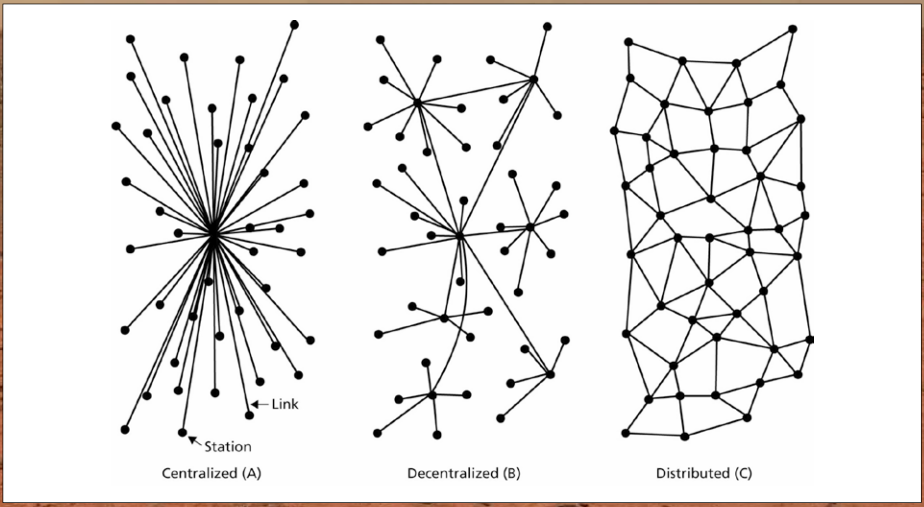


Figure 1. This scheme was devised by electrical engineer Paul Baran for his research into resilient communication networks. It was published in 1962 by the Rand Corporation under the title 'On Distributed Communications Networks' (https://bit.ly/2AFAqdp).

The invisible takeover of the web

Because of these and other technical properties of the IPFS, it is not susceptible to the ailments that the current web is struggling with. DDoS does not work because there is no central server to attack. That is why IPFS is censorship-resistant. During the Turkish Wikipedia ban, a copy of Wikipedia was made available via an IPFS network. Because the information is hosted on multiple computers, it is much more difficult to suppress its distribution.

IPFS also gives people control over their data. If you want to share a video, you can make it available on the network, for example by hosting it on your own computer, after which it can be further distributed in the network. The content is immediately available and is not locked in "central" platforms such as Youtube and Facebook.

Finally, IPFS also works offline. The system does not need to be constantly connected to the entire physical Internet to function. It also works locally. That's why the future Martians are better off with IPFS. The first Martian to want to see Wikipedia will have to pick it from the Earth network and take the long wait in his/her stride. But after that, none of his/her companions have to travel the slow route to Earth anymore. Wikipedia is now available locally on Mars.

Perhaps the most impressive aspect of IPFS is that the creators want to implement it without anyone realising it. Invisible to end users, the current HTTP protocol is replaced by IPFS. Juan Benet, who was at the cradle of IPFS, said about this in a presentation for students at Stanford University: "If nobody knows that a switch has taken place, we've won. The goal is to keep the same interface. If you design a system that forces users to change their habits, they simply won't use it. That is the way to improve the Internet: nobody should know. *It should just get better*." [1]

180572-02

IPFS is an open source project that's open for anyone to use and contribute to. For more information: https://ipfs.io.

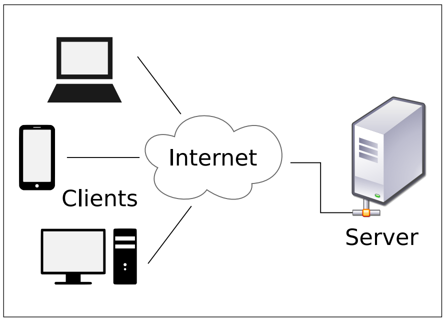


Figure 2. The client-server model
(David Vignoni: https://bit.ly/2t7aPmM).

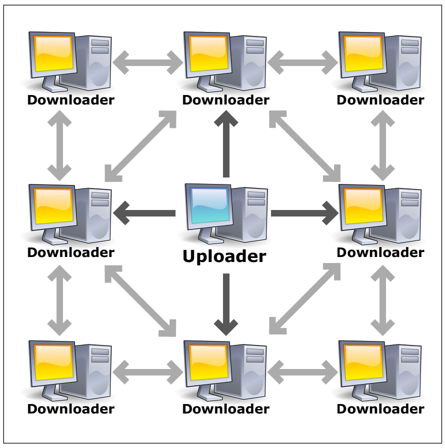


Figure 3. Bittorrent network
(Scott Martin: https://bit.ly/2PCnuNu).

Web Link

[1] Stanford Seminar – IPFS and the Permanent Web, by Juan Benet.: https://bit.ly/2qncrtx

welcome in your ONLINE STORE

EDITOR'S CHOICE

A lot of electronics designers and electronics repairers might be dreaming of a thermal imaging camera. However, most of these are in a price category that makes it seriously difficult to justify such a purchase. In our lab we are keen to use the handheld thermal imaging camera HT-02. Working with the device itself is quite straightforward. The configuration options are very complete. For the target groups — service personnel and developers — I cannot quickly think of any missing options that would be indispensable in this price category. It actually does very well what it needs to do and I really can't think of any negative points.

Thijs Beckers (Elektor Editor)

www.elektor.com/ht-02

Elektor Bestsellers

1. Raspberry Pi 3 (model A+)
   www.elektor.com/rpi3a-plus

2. Android App Development
   www.elektor.com/b4a-book
3. VFD-Tube Clock
   www.elektor.com/vfd-tube-clock
4. Programming with STM32 Nucleo Boards
   www.elektor.com/stm32-nucleo-book
5. Raspberry Pi 3 (model B+)
   www.elektor.com/rpi3b-plus
6. PiJuice HAT
   www.elektor.com/pi-juice-hat
7. The Official ESP32 Book
   www.elektor.com/esp32-book

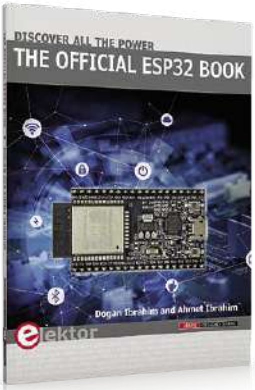
My Journey into the Cloud

Our German Editor-in-Chief has made its way into the IoT. As part of a series of articles, he developed several demo projects - from the lamp control in the home network up to a autarkic sensor board that sends datas to the cloud. With the slogan „Learning by Doing", topics such as TCP/IP, MQTT, control via smartphone, WiFi access, connection of a cloud service, object-oriented programming and much more are treated. In this book, the first 24 episodes of this IoT series compactly summarized.

member price: £11.95 • €13.46 • US $16

www.elektor.com/journey-into-cloud

The Official ESP32 Book



This book is an introduction to the ESP32 processor and describes the main hardware and software features of this chip. The book teaches the reader how to use the ESP32 hardware and software in practical projects. Many basic, simple, and intermediate level projects are given in the book based on the ESP32 DevKitC development board, using the highly popular Arduino IDE and also the MicroPython programming language.

member price: £27.95 • €31.46 • US $36

www.elektor.com/esp32-book

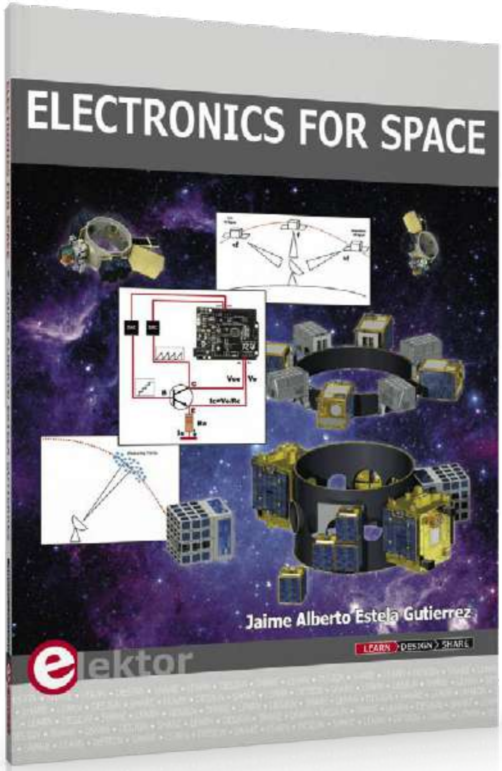
Raspberry Pi TV HAT

The Raspberry Pi TV HAT allows you to receive digital terrestrial TV broadcast systems, including DVB-T and DVB-T2, on a Raspberry Pi. With the board you can receive and view TV on a Raspberry Pi, or create a TV server that allows you to stream received TV over a network to other devices. The TV HAT can be used with any 40-pin Raspberry Pi board as a server for other devices on the network. The board comes with a set of mechanical spacers, a 40-way header, and an aerial adaptor.

member price: £19.95 • €22.46 • US $26

www.elektor.com/raspberry-pi-tv-hat

NEW



Electronics for Space

Space, the final frontier, will become more and more popular. The space industry is continually growing and new products and services will be required. Innovation is needed for the development of this industry. Today it is no longer possible to follow all the events in field of space. The space market is growing and activities are increasing, especially the market for small-satellites. One of the main difficulties is finding people with knowledge of space electronics design. This book wants to help close the gap and encourage electronic engineers to enter into the fascinating field of space electronics.

MEMBER PRICE: £19.95 • €22.46 • US $26

www.elektor.com/electronics-for-space

PORSCHE Carrera-Race Engine

The Carrera 547 racing engine was the first Porsche engine specially developed for use in races. The Carrera Race Engine kit brings the 547 vertical shaft engine to your desk as a transparent functional model on a scale of approx. 1:3. With this package, containing more than 300 parts, you can build a transparent functional model of the four-cylinder boxer engine from 1953 in around six hours. No glue needed.

member price: £183.95 • €206.10 • US $234

www.elektor.com/porsche-carrera

Programming with STM32 Nucleo Boards

FREE L476RG BOARD

The book covers many projects using most features of the STM32 Nucleo development boards where the full software listings for Mbed and System Workbench are given for every project. The projects range from simple flashing LEDs to more complex projects using modules and devices such as GPIO, ADC, DAC, I²C, LCD, analog inputs and others. Comes with a FREE STM32 Nucleo L476RG Board for a limited time only!

member price: £27.95 • €31.46 • US $36

www.elektor.com/stm32-nucleo-book

Android App Development

You can develop apps for Android mobile devices using the Basic For Android (B4A) programming language and Integrated Development Environment (B4A IDE). This book includes simple projects to introduce B4A's syntax and programming features. Electronics designers will enjoy this book because it describes how an Android mobile device can communicate with a variety of hardware platforms over a Wi-Fi link or by using SMS text messages.

member price: £23.95 • €26.95 • US $31

www.elektor.com/b4a-book

Hexadoku The Original Elektorized Sudoku

Traditionally, the last page of Elektor Magazine is reserved for our puzzle with an electronics slant: welcome to Hexadoku! Find the solution in the gray boxes, submit it to us by email, and you automatically enter the prize draw for one of five Elektor book vouchers.

The Hexadoku puzzle employs numbers in the hexadecimal range 0 through F. In the diagram composed of 16 × 16 boxes, enter numbers such that **all** hexadecimal numbers 0 through F (that's 0-9 and A-F) occur once only in each row, once in each column and in each of the 4×4 boxes (marked by the thicker black lines). A number of clues are given in the puzzle and these determine the start situation.

Correct entries received enter a prize draw. All you need to do is send us **the numbers in the gray boxes**.

Solve Hexadoku and win!

Correct solutions received from the entire Elektor readership automatically enter a prize draw for five Elektor Book Vouchers worth **$70.00 / £40.00 / €50.00 each**, which should encourage all Elektor readers to participate.

Participate!

Ultimately January 21, 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 6/2018 (November & December) is: **386BF**.

The €50 / £40 / $70 book vouchers have been awarded to: Frederik Andriof (Germany), Freddy Ghys (Belgium), Biette Francis (France), Alexandr Papazyan (Russia) and Henk Brand (The Netherlands).

Congratulations everyone!

3	8		A	C		E			1		0	D		B	4
6	7	F	1		D	B			3	2		E	5	0	C
		E	B				7	D				F	6		
	C					6			F					9	
	A			6		D			4		2			5	
2			E	4	3		9	1		C	6	A			F
	9				C	7	0	3	E	5				8	
1	6		4		2					D		9		C	B
	E			0			5	2			F			A	
A	F		0	D		3			6		E	5		7	9
	D	7	C	9							3	B	0	F	
5		8			E					0			C		1
			F	5	7					E	4	0			
		6			A					3			F		
D		A		E							1		9		5
	2			B	F					9	5			1	

0	E	3	7	F	9	4	8	1	2	6	A	5	D	C	B
F	2	1	A	D	0	C	3	7	5	8	B	E	6	9	4
4	6	5	8	1	2	E	B	C	9	3	D	F	0	7	A
9	B	C	D	A	5	6	7	4	E	F	0	1	2	8	3
E	3	8	6	B	F	D	C	5	1	A	4	9	7	0	2
A	C	9	B	4	6	0	E	3	7	D	2	8	5	1	F
D	F	0	5	3	7	1	2	8	6	B	9	A	4	E	C
7	1	2	4	5	8	9	A	E	F	0	C	B	3	6	D
B	4	E	9	C	D	F	5	A	0	1	3	6	8	2	7
1	5	D	2	6	3	8	9	F	4	C	7	0	A	B	E
8	0	6	F	7	E	A	4	9	B	2	5	3	C	D	1
C	7	A	3	0	B	2	1	D	8	E	6	4	9	F	5
5	9	F	C	8	1	7	D	6	A	4	E	2	B	3	0
2	A	B	1	9	C	3	F	0	D	5	8	7	E	4	6
3	8	4	0	E	A	B	6	2	C	7	1	D	F	5	9
6	D	7	E	2	4	5	0	B	3	9	F	C	1	A	8

The competition is not open to employees of Elektor International Media, its subsidiaries, licensees and/or associated publishing houses.

THERMOSTAT
SECURITY PANEL
SMART REFRIGERATOR
SMART | CONNECTED | SECURE

reichelt elektronik

Technology connects.

✓More than 100,000 products
✓Top price/performance ratio
✓High availability and reliable delivery
✓Strong brands and choice quality

THE MOST IMPORTANT TOOLS FOR ELECTRONICS SPECIALISTS

Weller®

The Weller WE1010 Education Bundle

70 watts of power for professional requirements!

The most powerful soldering station in its class! This economy bundle will equip you for any task. In addition to the WE1010 soldering station this set includes 170 mm Weller Xcelite side cutters, a roll of Weller WSW SCN N1 solder, and an additional ETB 2.4 mm soldering tip.

- ESD-safe soldering station, soldering iron and heat-resistant silicone cable
- Temperature stability and lock for protection of soldering tip and component
- Password protection saves all settings

Weller Educational Bundle:
+ Side cutters
+ Solder
+ Soldering tip

Order no.: WELLER WE1010EB

NEW £ 182.48 (£ 152.06)

AC power cable, 1.8 m

- UK plug > IEC socket

NKSK UK 180 **£ 3.39** (£ 2.83) **ORDER NOW**

TRMS digital multimeter

XLABs

With integrated LED work light.

- Illuminated display with 6,000 counts
- Voltage up to 1000 V DC / 750 V AC
- Current up to 10 A AC/DC
- Frequency, capacitance and resistance measurement
- Duty-cycle measurement: 5 – 95 %

THE UNIVERSAL INSTRUMENT FOR ALL MEASUREMENT TASKS

EN 61010-1 CAT III 1000 V | EN 61010-1 CAT IV 600 V | EN 61010-1

Order no.:
VAL MS0205

BEST SELLER £ 36.63 (£ 30.53)

6-piece screwdriver set

RND lab

Ergonomic design for outstanding grip and comfort.

- Blades: chrome vanadium molybdenum steel
- Blade profiles: 2x Philips: PH0 / PH00 4x plain slot: 1.6 / 2.0 / 2.4 / 3.0 mm

Order no.:
RND 550-00197

NEW £ 13.75 (£ 11.45)

LED illuminated magnifier with real glass lens

Shadow-free illumination and detailed magnification.

- Lens: Ø 100 mm, 3 dioptres
- Ring-shaped diffuser
- Light: cold white (6,400 K)

A+
(A++ - E)

Order no.:
LL ZD-2012B-2H

instead of 22.88

SAVE 20 % £ 18.29 (£ 15.24)

Daily prices! Price as of: 4. 12. 2018

Payment Methods: VISA PayPal

The statutory cancellation provisions apply. All prices are stated in £ including statutory value added tax, plus shipping charges for the entire shopping basket. Our Terms and Conditions (available at https://rch.lt/TERMS or on request) apply exclusively. Actual appearance of products may differ from photos. We accept no liability for misprints or errors; prices subject to change.
reichelt elektronik GmbH & Co. KG, Elektronikring 1, 26452 Sande (Germany), Phone: +44 203 808 95 25

www.reichelt.co.uk

ORDER HOTLINE: +44 203 808 95 25