

Driving Motors with H-Bridges



p. 6

Versatile Servo
Tester

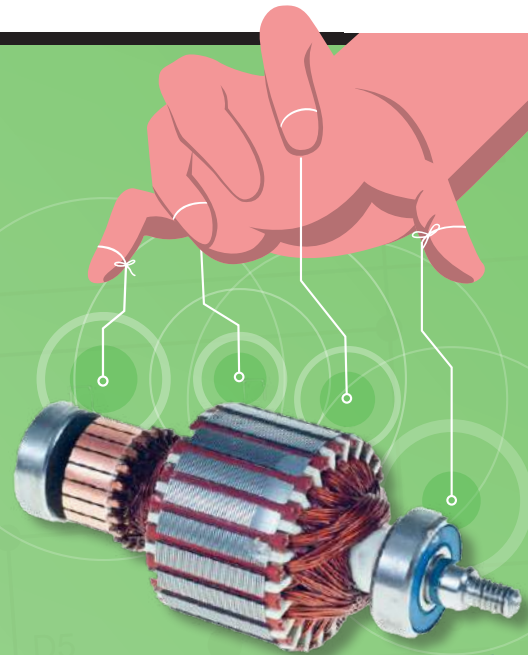


p. 73



p. 52

Large
Electric Motors



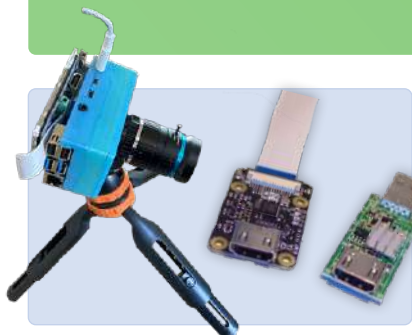
In this edition

- > DIY Master Power Switch for the Lab Bench
- > The Raspberry Pi Zero 2 W Goes Quad-Core
- > The Elektor Lab Team
- > Sound Card for the Raspberry Pi Family
- > Practical Neurons
- > Inside an Open-Source Processor

ELEKTOR INDUSTRY

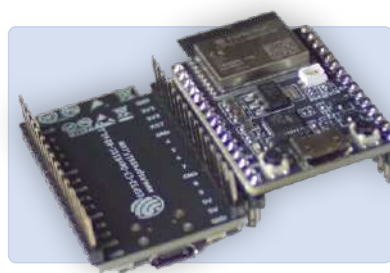
- > Motor Control Complexity Simplified
- > Impressions from the World Ethical Electronics Forum

and much more!



Raspberry Pi as a KVM
Remote Control
Software in Practice

p. 22



ESP32 with RISC-V Core
Getting Started with the
ESP32-C3

p. 59



Autonomous Vehicle with
2D Lidar Remote Control Over
Bluetooth

p. 36

elektor e-zine

Your dose of electronics



Every week that you don't subscribe to Elektor's e-zine is a week with great electronics-related articles and projects that you miss!

So, why wait any longer? Subscribe today at www.elektor.com/ezine and also receive free Raspberry Pi project book!



What can you expect?

Editorial

Every Friday, you'll receive the best articles and projects of the week. We cover MCU-based projects, IoT, programming, AI, and more!

Promotional

Don't miss our shop promotions, every Tuesday and Thursday we have a special promotion for you.

Partner mailing

You want to stay informed about the ongoing activities within the industry? Then this e-mail will give you the best insights. Non-regular but always Wednesdays.

Elektor Magazine,

English edition

Edition 1/2022

Volume 48, No. 511

January & February 2022

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

www.elektor.com

www.elektormagazine.com

Elektor Magazine, English edition

is published 8 times a year by

Elektor International Media

78 York Street

London W1H 1DP

United Kingdom

Phone: (+44) (0)20 7692 8344

Head Office:

Elektor International Media b.v.

PO Box 11

NL-6114-ZG Susteren

The Netherlands

Phone: (+31) 46 4389444

Memberships:

Please use London address

E-mail: service@elektor.com

www.elektor.com/memberships

Advertising & Sponsoring:

Raoul Morreau

Phone: +31 (0)6 4403 9907

E-mail: raoul.morreau@elektor.com

www.elektor.com/advertising

Advertising rates and terms available on request.

Copyright Notice

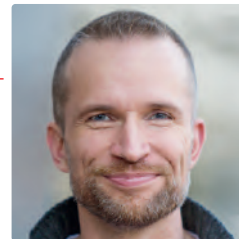
The circuits described in this magazine are for domestic and educational use only. All drawings, photographs, printed circuit board layouts, programmed integrated circuits, disks, CD-ROMs, DVDs, software carriers, and article texts published in our books and magazines (other than third-party advertisements)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

© Elektor International Media b.v. 2022

Printed in the Netherlands

Jens Nickel

International Editor-in-Chief, Elektor Magazine



Motor Control and More

We know from surveys that most *Elektor* readers are also professional electronics engineers. They work in development, purchasing, or sales of components and devices. We have therefore always been a magazine that has defied the usual division into hobby and professional electronics. After all, many a professional developer has drawn valuable inspiration from our playfully presented projects. And in times when makers are founding their own companies or are being courted by the development departments of large semiconductor companies, the old black-and-white thinking is probably finally obsolete.

A few years ago, we launched our Industry Special (first under the name “Elektor Business”). Here you could find out first-hand what the development departments of the manufacturers had just brought to market. There were also interesting overview articles on the latest trends in the industry (e.g., by Stuart Cording). The engineer was on the road for us at trade shows and had an excellent connection to many experts who provided him with background information.

In 2022, starting with this issue, we will return to a uniform concept and integrate the interesting articles from the industry into *Elektor* magazine. With a stronger focus on one main topic — without sacrificing coverage of the entire spectrum of electronics in our issues — we can cover different areas more comprehensively and in depth.

In this issue, we begin with a focus on “Motor Control.” On pages 6, 52, and 73, you’ll find two basic articles by Mathias Claussen and Thomas Scherer, as well as (of course!) a circuit project on the subject. And on the Industry pages Stuart Cording shows in another article how motor controls can be simplified.

Stay curious!

The Team



International Editor-in-Chief: **Jens Nickel**

Content Director: **C. J. Abate**

International Editorial Staff: **Eric Bogers, Jan Buiting, Stuart Cording, Rolf Gerstendorf,**

Dr Thomas Scherer, Clemens Valens

Laboratory Staff:

Mathias Claussen, Ton Giesberts, Luc Lemmens, Clemens Valens

Graphic Design & Prepress:

Giel Dols, Harmen Heida

Publisher:

Erik Jansen



Elektor is a member of FIPP, an organization that has “grown over almost 100 years to include media owners and content creators from across the world.”



Elektor is a member of VDZ (Association of German Magazine Publishers), which “represents the common interests of 500 German Consumer and B2B publishers.”

Driving Motors with H-Bridges

An Introduction to DC, Stepper, and Brushless Motors



Regulars

- 3 Colophon**
- 72 productronica Fast Forward 2021 Winners**
Exciting Technologies and Creative Solutions
- 92 Starting Out in Electronics**
We Are Not Yet Done with the Coil
- 96 Err-lectronics**
Corrections, Updates and Readers' Letters
- 110 Ethics**
Poverty and Electronics
- 114 Hexadoku**
The Original Elektorized Sudoku

- 52 Large Electric Motors**
Basic Principles and Useful Information
- 68 Create GUIs with Python (Part 2)**
Spy Name Chooser
- 82 Understanding the Neurons in Neural Networks (Part 3)**
Practical Neurons
- 87 Inside an Open-Source Processor**
Sample Chapter: Lattice and Xilinx FPGA Results
- 102 BattLab-One**
Measure and Optimize the Battery Life of IoT Devices

Features

- 6 Driving Motors with H-Bridges**
An Introduction to DC, Stepper, and Brushless Motors
- 16 The Elektor Lab Team**
Our Approach, Preferred Tools, and More
- 28 IQaudio Codec Zero**
A Sound Card for the Raspberry Pi Family
- 31 The PiKVM Project and Lessons Learned**
Interview with Maxim Devaev (Developer, PiKVM)
- 40 The Raspberry Pi Zero 2 W Goes Quad-Core**

Industry

- 44 Notes From the 2021 World Ethical Electronics Forum**
- 46 Motor Control**
How the Complexity of Motor Control Is Simplified

The  lektor LAB Team

Our Approach, Preferred Tools, and More



16

The Raspberry Pi Zero 2 W Goes Quad-Core



40

Protect Yourself and Others!

DIY Master Power Switch for the Lab Bench

64



Projects

- 22 Raspberry Pi as a KVM Remote Control**
Pi-KVM Software Test
- 36 Autonomous Vehicle with 2D Lidar**
ESP32 Pico Interprets Data from the Lidar Module
- 59 Getting Started with the ESP32-C3 RISC-V MCU**
- 64 Protect Yourself and Others!**
DIY Master Power Switch for the Lab Bench
- 73 Versatile Servo Tester**
Check Behavior When There's No Datasheet
- 78 Modbus Over WLAN (Part 2)**
Software for the Modbus TCP WLAN Module
- 98 Color to Sound**
How to Read Out a Color Sensor via I²C
- 106 Simple Earth-Leakage Tracer**
Testing Isolation of Mains Supply



FOCUS

73

78

98

106



Simple Earth-Leakage Tracer
Testing Isolation of Mains Supply

106

Next Edition

Elektor Magazine Edition 3-4/2022 (March & April 2022)

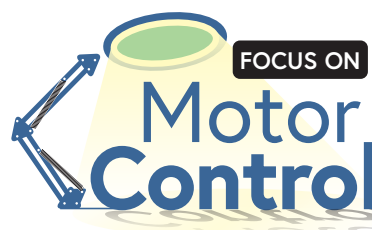
As usual, we'll have an exciting mix of projects, circuits, fundamentals and tips and tricks for electronic engineers and makers. We will focus on Embedded Development.

From the contents:

- > Workshop: RISC-V Core on FPGA
- > Buffer Board for the Raspberry Pi 400
- > 4-Bit and Other Low-Cost Controllers
- > Spring Collection of Raspberry Pi RP2040 Boards
- > Portable Temperature- and Humidity-Measuring Device
- > Audio Measurements with USB Audio Interface
- > WinUI 3: A New Graphics Framework for Windows Apps
- > Over-the-Air Updates for Arduino and ESP
- > NB-IoT in Practice

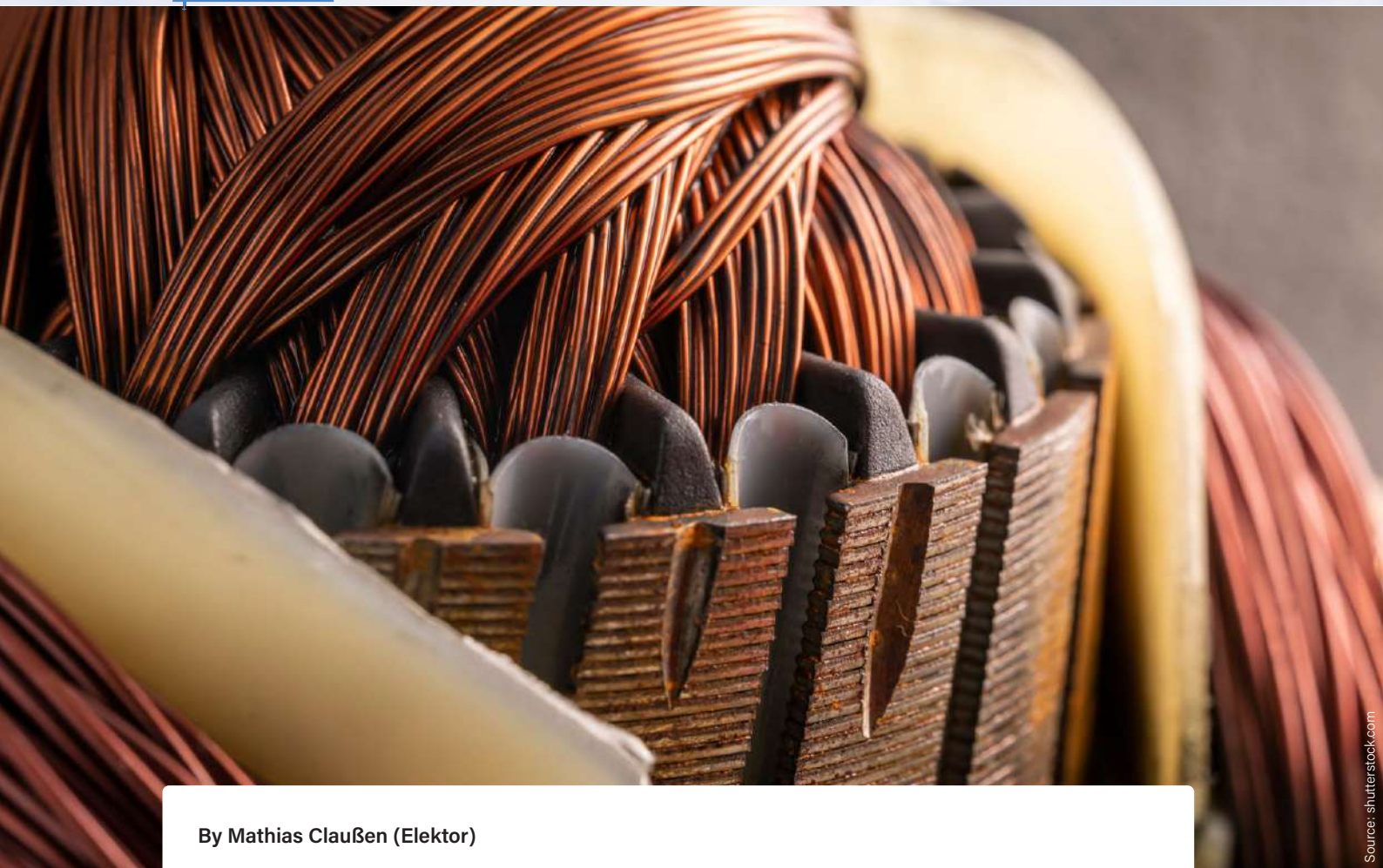
And much more!

Elektor Magazine edition 3-4/2022 covering March & April 2022 will be published around March 10th, 2022. Arrival of printed copies with Elektor Gold Members is subject to transport. Contents and article titles subject to change.



Driving Motors with H-Bridges

An Introduction to DC, Stepper, and Brushless Motors



By Mathias Claußen (Elektor)

Motor control is required in many applications — not only for fans and ventilation systems. Conveyors, pumps, and 3D printers are also equipped with motors that cannot be simply operated in on-off mode. Motor power and rotation direction also need to be controlled. To make make this an easy introduction, here we only consider conventional DC motors, stepper motors, and low-power brushless motors.

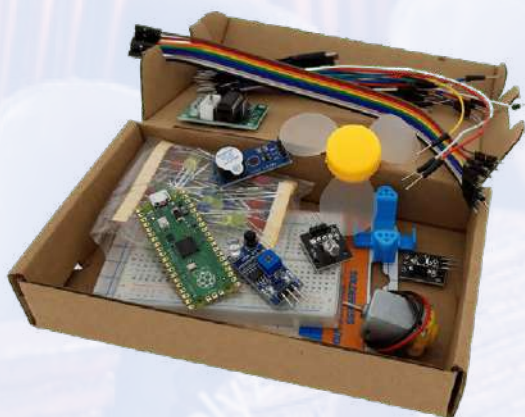


Figure 1: The Basic Kit for Raspberry Pi Pico with a DC motor. (Source: Kuongshun)

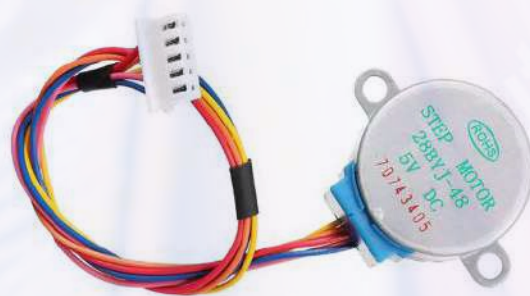


Figure 2: Type 28BYJ-48 stepper motor for 5 V DC. (Source: Geekcreit)

The circuits shown in this article are intended to illustrate the individual sections and serve as starting points for your own designs, not as finished designs that you can copy directly. First, let's look at a simple DC motor. In this case, it is a motor with a rated operating voltage of 3 V to 6 V and a maximum operating current of 1 A. Motors of this sort are included in many starter kits, such as the Basic Kit for Raspberry Pi Pico [1] (**Figure 1**). In the latter case, the supply voltage is 5 V, as provided by a standard USB port. Some starter kits also include a stepper motor or offer it as a low-cost additional purchase. The 28BYJ-48 [2] (**Figure 2**), which can be powered from 5 V, is a common type.

Forwards and Backwards

Let's look at the circuit shown in **Figure 3**. Relay K1 acts as a switch to power the motor on or off. The relay is controlled by momentary switch S1, so that the current required by the motor only flows through the relay, not through the switch. Diode D1 is a snubber diode and is there to prevent voltage spikes over the relay coil when the switch opens.

When momentary switch S1 is pressed, the relay K1 engages and the motor M1 can start running. When momentary switch S1 is released, the relay drops out and the motor is no longer supplied with power. The windings of DC motors can also generate voltage spikes, the same as relay coils. Here diode D2 acts as a snubber diode for the motor to prevent the motor from generating an undesirable voltage spike when it is switched off. With this circuit it is only possible to run the motor in the forward direction. But what if you also want to run the motor backwards (e.g., to drive a winch)? Then you have to modify the circuit so the motor can run backwards.

An H-bridge is a good option for this. An example is shown in **Figure 4**. The name is based on the structure in the circuit diagram. The motor is connected in the middle of the H. If you press momentary switch S1 (motor forwards), then relays K1 and K4 will engage to allow current to flow into the positive terminal of the motor and out of the negative terminal. If you want the motor to run backwards, you

can press momentary switch S2 to engage relays K3 and K2. Then current will flow into the negative terminal of the motor and out of the positive terminal, and the motor will run backwards. Note that you should never press both momentary switches (S1 and S2) at the same time. If relays K1 and K2 or relays K3 and K4 were engaged at the same time, a large short-circuit current would flow through the relays and could damage them severely.

From Slow to Fast

Using the H-bridge, we can now control the rotation direction of the motor. However, we are not yet able to control the motor speed. Switching relays under load, especially with DC, considerably shortens their life expectancy. In addition to the drawback of shortened lifetime with a large number of switching cycles, relays are neither compact nor energy-efficient.

The first logical improvement is to use semiconductor devices. Let's look at a layout with NPN and PNP transistors. **Figure 5** shows the modified circuit diagram. Here the relays have been replaced by a pair of PNP transistors (T1 and T3) and a pair of NPN transistors (T2 and T4). The drive circuitry has also changed. Now switch S1 pulls the bases of T1 and T2 to ground through the current-limiting resistors R1 and R2 when the switch is closed, or ties them to the supply voltage through R5 when the switch is open. Switch S2 performs the same actions for transistors T3 and T4.

In the quiescent state of the circuit with switches S1 and S2 open, the supply voltage is applied to resistors R1 to R4 through resistors R5 and R6. This means that transistors T1 and T3 are cut off and therefore have very high impedance (in the megohm range). By contrast, transistors T2 and T4 are driven into conduction by their base currents and therefore have low impedance. As a result, both ends of the motor are connected to ground and no current flows through the motor.

What happens now if switch S1 is closed and switch S2 is open? Then resistors R1 and R2 will be tied to ground, so transistor T2 will

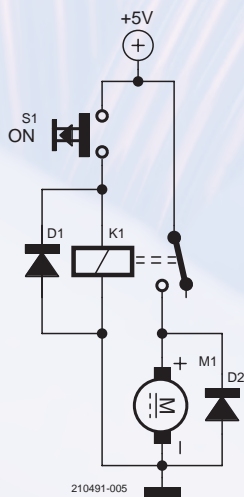


Figure 3: Motor control with relays.

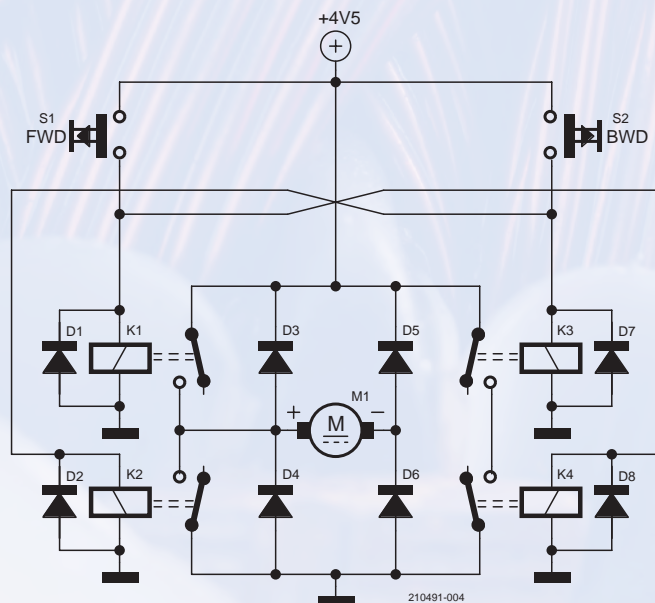


Figure 4: H-bridge with relays.

no longer have any base current because the base and emitter are at the same potential. Transistor T1, by contrast, will be conducting because current can flow from the emitter to the base. As a result, the positive supply voltage is applied to one terminal of the motor while the other terminal is tied to ground, so the motor can start running. If switch S1 is open and switch S2 is closed, then transistor T3 will be conducting and transistor T4 will be cut off. In this case the motor also starts running, but in the other direction.

But what if you want to be able to control the motor speed, for example using a microcontroller? Instead of the switches, you can use two pins of a microcontroller. These are marked *PWM_A* and *PWM_B* in **Figure 6**. If you want the motor to run forwards or backwards, you simply have to set these pins High or Low as appropriate. If you also want to control the speed, you can use pulse width modulation (PWM). This way the speed is determined by the duty cycle (the On time relative to the period) of the PWM signal, and the direction depends on whether a pulse waveform is output on *PWM_A* or *PWM_B*. This layout is designed for a 5 V supply voltage, so the outputs of the MCU must be able to handle 5 V signal levels. If the

motor needs to be operated with a higher voltage or the MCU can only tolerate 3.3 V on its I/O pins (as with the Raspberry Pi), the circuit has to be modified a bit.

The result is shown in **Figure 7** with two additional transistors (T5 and T6) driven by the PWM signals through base resistors. Thanks to these transistors, it is possible to use MCUs that operate with a supply voltage lower than the voltage for the DC motor. Here it should be noted that the PWM signals are now inverted. If a voltage is applied to transistor T5 or T6, this transistor is driven into conduction and pulls the inputs of transistors T2 and T1 or transistors T3 and T4 to ground. This must be taken into account in the drive signals from the MCU, since forwards and backwards are swapped now.

There are drawbacks to using PNP and NPN transistors in this application. They need base currents, which means they consume energy, and the frequency of the PWM signal is limited. Switching between the high-impedance and low-impedance states is not instantaneous, so there are switching losses. Furthermore, at the moment when T1 and T2 or T3 and T4 change states, there is a brief interval in which

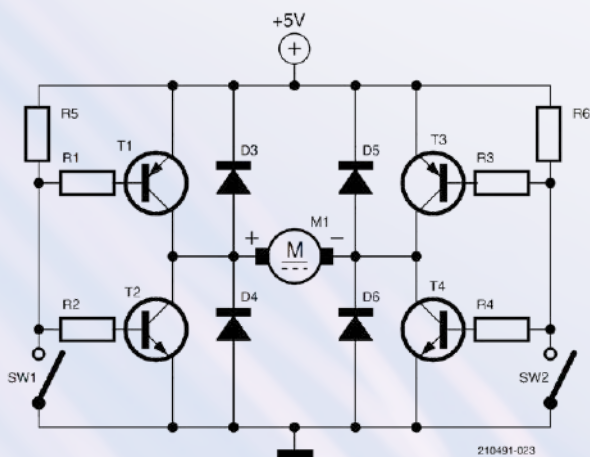


Figure 5: H-bridge with NPN and PNP transistors.

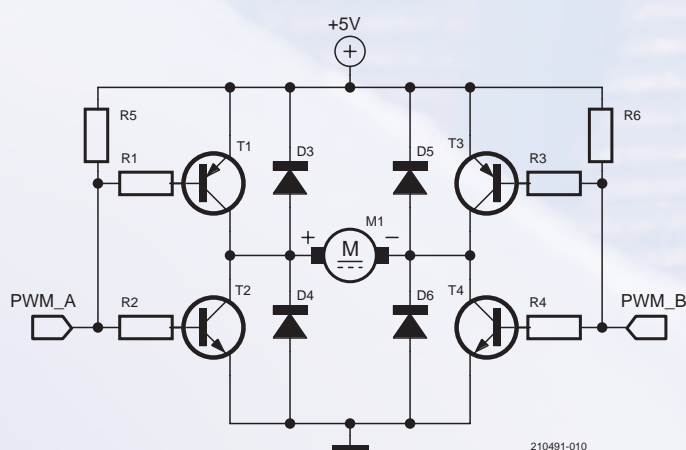


Figure 6: Driving with PWM signals from an MCU.

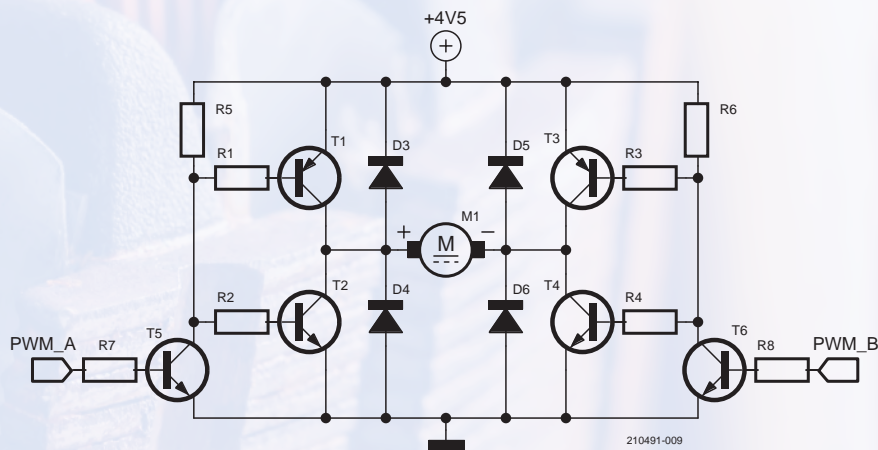


Figure 7: Driving with 3.3 V signals.

both transistors of the pair are conducting. This allows a higher current to flow directly from the supply voltage to ground. This also increases the power dissipation and can even lead to early failure.

H-Bridges with FETs

To avoid some of the drawbacks of bipolar transistors, a good choice is to use field effect transistors (FETs). They can achieve higher switching frequencies and have lower power dissipation. All of the H-bridges described above have the problem that the transistors can generate transient short-circuit currents, and this problem needs to be eliminated. The H-bridge shown in **Figure 8** is therefore built with FETs that can be driven individually.

Here T1 and T3 are PMOS FETs, while T2 and T4 are NMOS FETs. With a PMOS FET, the FET is cut off (high impedance) when the voltage difference between the gate and the source is close to zero. In order to drive a PMOS FET into conduction (low impedance), the voltage on the gate must be more negative than the voltage on the source. For T1 and T3, this means that when the gate is pulled to ground, the FET conducts. The maximum permissible voltage

between the gate and the source must also be taken into account. If this voltage is exceeded, the FET will be damaged. With an NMOS FET, the voltage on the gate must be more positive than the voltage on the source to drive the FET into conduction. For T2 and T4, this means that they are in the high-impedance state when the gate is pulled to ground, and they start to conduct when a positive voltage (relative to ground in this case) is applied to the gate.

Diodes D4 to D6 are still present in Figure 8, but they are not needed when FETs are used because the body diodes of the FETs serve the same purpose. The FET symbol in **Figure 9** for T1-T4 clearly shows the body diode as a component of the FET structure. This means that diodes D4 to D6 can be omitted, resulting in the circuit diagram in **Figure 10**.

If you only have a 5 V supply, couldn't you simply connect PWM_A to !PWM_A and PWM_B to !PWM_B? In theory, this is possible, but it would lead to even worse transient short-circuit currents than with bipolar transistors. As a mental experiment, let's suppose that PWM_A and !PWM_A are connected. With 5 V applied to T1 and

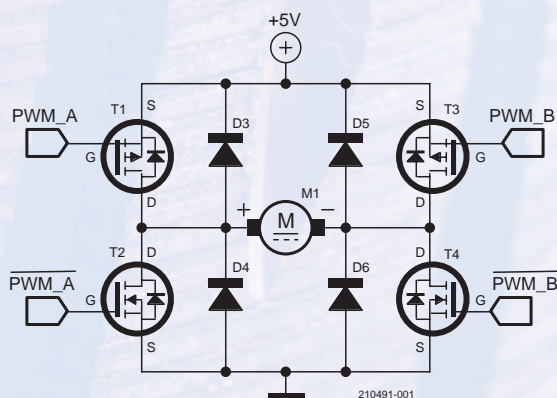


Figure 8: H-bridge with FETs.

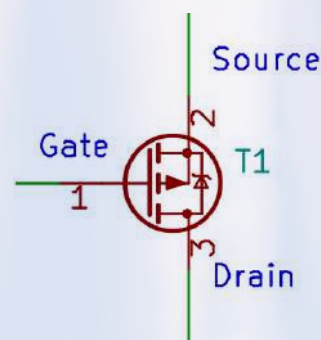


Figure 9: FET symbol with body diode.

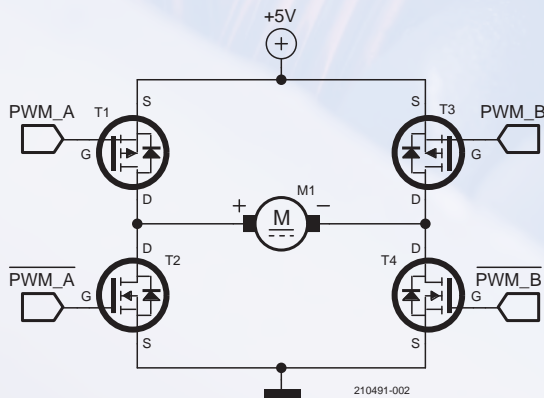


Figure 10: Circuit diagram without snubbing diodes.

T2 as the initial condition, T1 is conducting and T2 is cut off. Now we pull PWM_A and !PWM_A to ground. The 5 V signal drops to 0 V, but not straight away. Instead it declines gradually, over a time interval in the nanosecond range. In this process there is a point where both FETs are conducting: T2 is not yet completely cut off and T1 is already conducting. In this situation both FETs have a resistance on only a few ohms, resulting in a short-circuit current directly through T1 and T2. This is called 'shot-through', and it is undesirable in H-bridges because the components can be severely damaged. You are therefore strongly advised to not try to connect the two signals. Some PWM units of advanced MCUs offer the option of generating signals suitable for driving H-bridges and are able to insert suitable dead times to avoid shot-through conditions.

H-Bridge Driver ICs

You can have the MCU generate dead times to ensure that T1 and T2 are never conducting at the same time, or you can use a suitable H-bridge driver IC. The above-mentioned H-bridge with two NMOS FETs and two PMOS FETs has the disadvantage that the conductivity of the PMOS FETs is at least a factor of 10 worse than their NMOS

counterparts. This means higher power dissipation, especially with relatively high currents. **Figure 11** shows a circuit with two half-bridge driver ICs (ON Semiconductor NCP5901B) and a layout with four NMOS FETs. There are no blocking capacitors in the circuit diagram; capacitors C1 and C2 are part of the charge pump of the NCP5901B. But why does the half-bridge driver IC need a charge pump? Here T1 and T3 are NMOS FETs, which as previously described need a gate voltage higher than the source voltage in order to start conducting. In the case of T1 and T3, this means that the voltage on the gate must be higher than the supply voltage of the H-bridge. A charge pump is used to achieve this. **Figure 12** shows the internal functional block diagram of the half-bridge driver IC. There you can see the dead time generation function to prevent shot-through, which is labelled 'Anti-Cross Conduction'.

This allows an H-bridge to be built from discrete components using FETs suitable for the use case concerned. Currents greater than 25 A can also be handled this way with H-bridges. If you are looking for a more compact solution for lower currents, you can also opt for a fully integrated solution.

The STM L298N, which is fitted on many low-cost motor driver boards (**Figure 13**), provides two complete H-bridges. It does not have integrated FETs, but does have integrated bipolar transistors. It can drive up to 4 A at 46 V. A glance at the block diagram (**Figure 14**) reveals a very familiar structure. MCUs with 3.3 V signal levels are certainly suitable for driving the L298N. The module is available from mail-order dealers for around €3 to €6 (£2.60 to £5.20 or \$3.50 to \$7.00). The Infineon IFX9201SG is distinctly more compact. It can drive up to 6 A and has integrated protection circuitry and diagnostic functions, as can be seen in **Figure 15**. It costs approximately €4 (£3 or \$4.20) per device. It would be so easy to design a DC motor driver with this device, but as of 1 September 2021 it is not expected to be available until mid-2022, like so many other semiconductor devices. If you are looking for a ready-made H-bridge for motor control, you should have a look at the Cytron motor driver boards [3] in the Elektor Store.

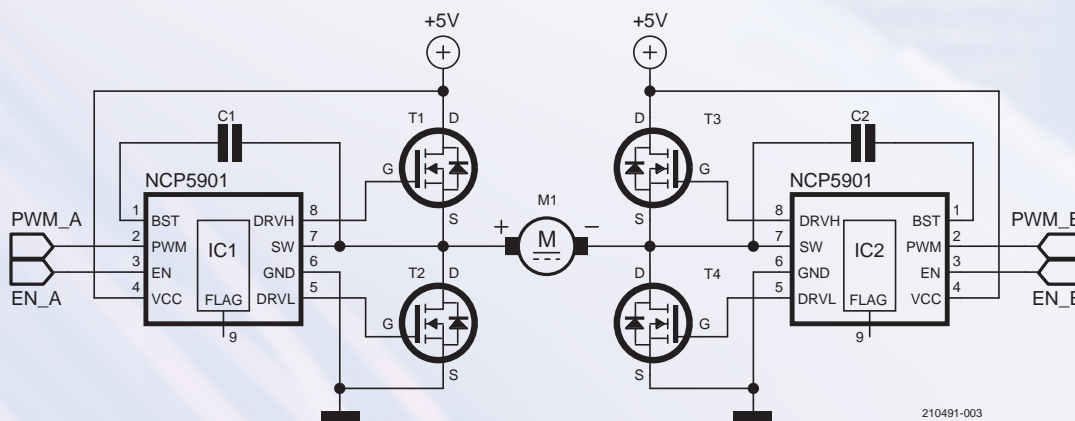


Figure 11: Four NMOS FETs with driver ICs.

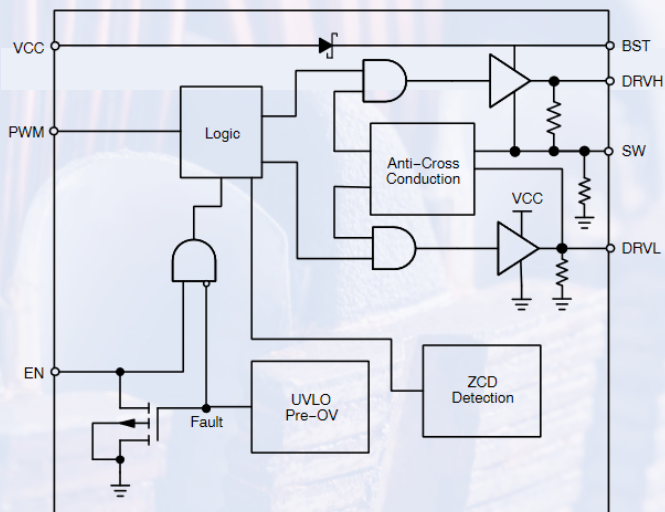


Figure 12: Internal block diagram of the ON Semiconductor NCP5901B (source: ON Semiconductor).



Figure 13: L298N module as a dual H-bridge.

Stepper Motors, Step by Step

As the name says, stepper motors do not simply run, but instead move in incremental parts of a full rotation. They are not intended for high speeds in continuous operation, but they can perform incremental motion with high torque. In most households, stepper motors are most likely to be found in ink-jet printers. There they are used to move the print head or transport the paper. A stepper motor operates with a defined step size (increment of a full rotation), so it is very easy to calculate the exact position of the print head. Some of us may also remember the sound of stepper motors in diskette drives, where they were used to move the read/write head.

But how do you drive a stepper motor, and why does it move in steps? Stepper motors are available in various forms: reluctance stepper motors, permanent magnet stepper motors, and hybrid stepper motors. Here we consider hybrid stepper motors, which have become established in many applications because of their good torque characteristics and large number of steps per rotation.

Figure 16 shows the structure of this kind of stepper motor. On the

inside there is a rotor, consisting of permanent magnets and toothed rings made of soft iron. The rotor consists of a set of windings arranged so that the rotor always move one step at a time.

In terms of drive, the most common stepper motors have two poles. Other pole counts are also possible, but are not considered here. In terms of structure, there are two types of this sort of stepper motor: unipolar and bipolar. We will look at these below.

Unipolar Stepper Motors

As an example of a stepper motor, here we use the 28BYJ-48 shown in Figure 2, which can be found in many starter kits. This small unipolar stepper motor additionally has a gear unit with a 63.68395:1 reduction ratio. The motor has a step size of 11.25°, so the number of steps for a full rotation of the gear unit output shaft is 2037.8864, or approximately 2038 steps per rotation (rounded up). Five wires are needed to connect the motor: A, B, C, D, and the supply voltage.

Driving a unipolar stepper motor is very easy, and with a motor of this size this can be done with a ULN2003, available from a variety

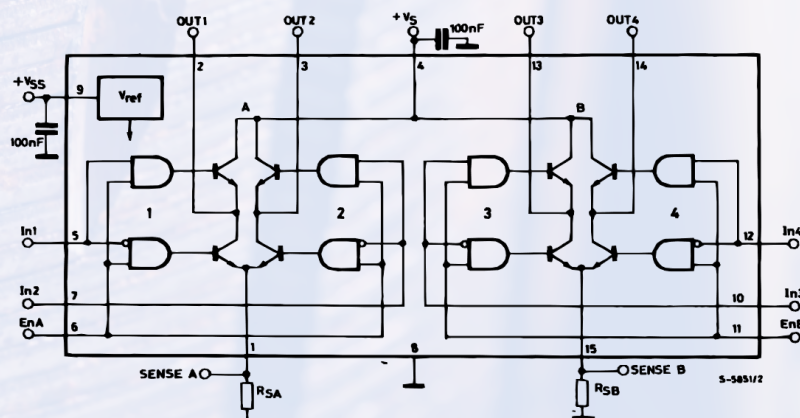


Figure 14: Internal structure of the L294N (source: STMicroelectronics).

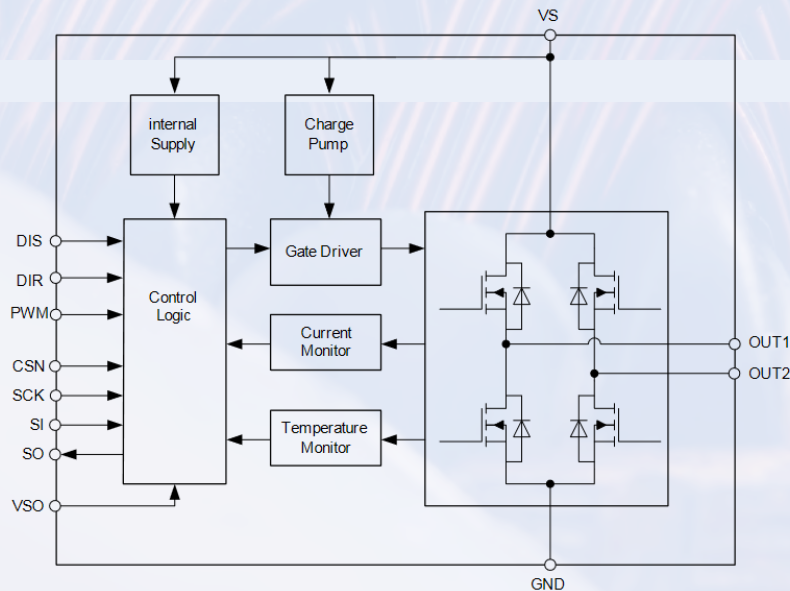


Figure 15: Internal block diagram of the Infineon IFX9201SG (source: Infineon).

of manufacturers. The ULN2003 contains seven Darlington transistors, as shown in the block diagram in **Figure 17**. Snubber diodes are also integrated in the IC, making it quite suitable for driving inductive loads. **Figure 18** shows a circuit with a ULN2003, which can be used to drive a 28BYJ-48 stepper motor. In addition to full step mode, the 28BYJ-48 can be operated in half-step mode. According to the data sheet, this is the recommended drive method. The sequence of the drive signals for clockwise rotation is listed in **Table 1**. To make a step in the backwards direction, you have to run through the sequence in reverse. Unipolar stepper motors have become rare now because they can deliver only about half as much torque as a bipolar stepper motor of the same size. However, this is offset by the very simple driver stage with a ULN2003. If you want, you can modify a 28BYJ-48 stepper motor for bipolar operation. To learn how, read jangeox's blog post [4]. Then the stepper motor will be able to generate more torque, but it will need a different driver.

Bipolar Stepper Motors

Bipolar stepper motors, which are installed in most 3D printers, have only four leads. For these stepper motors there are standardised

sizes defined by the National Electrical Manufacturers Association (NEMA). NEMA 17-03, for instance, designates the size of a type of stepper motor that is produced and/or sold by a variety of companies. The NEMA17-03 stepper motor from Joy-IT [5] is one example (**Figure 19**). A ULN2003 is not sufficient to drive this type of stepper motor, which requires two H bridges in the drive circuit. An L298N, as previously mentioned for DC motors, provides two H-bridges in a single package and is used as a stepper motor driver in the circuit in **Figure 20**. As these H-bridges are made with bipolar transistors, snubber diodes are required at the outputs. The sequence in **Table 2** must be performed to generate one step in the clockwise direction.

Unlike unipolar stepper motors, in this case the complete winding is used, instead of only half with a centre tap. This enables higher torque than with a unipolar stepper motor of the same size. Driving with H-bridges is very easy, and highly integrated drivers such as the MP6500 from MPS or the TMC2300-LA from Trinamic make this even easier. They contain part of the control logic that performs the step sequences. The MCU only specifies the desired direction (clockwise or anti-clockwise) and supplies a pulse for each desired step.

Table 1: Sequence for one step.

Step/ colour	Orange	Yellow	Pink	Blue
1.	1	0	0	0
2.	1	1	0	0
3.	0	1	0	0
4.	0	1	1	0
5.	0	0	1	0
6.	0	0	1	1
7.	0	0	0	1
8.	1	0	0	1

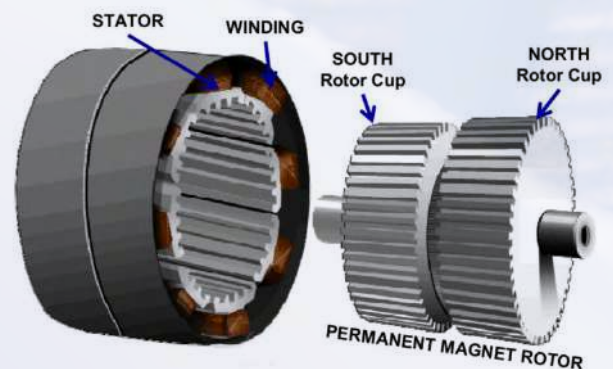


Figure 16: Structure of a stepper motor (source: Microchip)

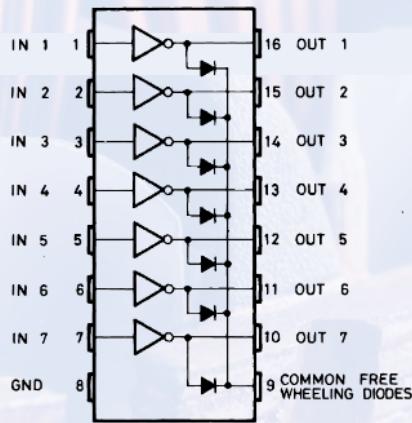


Figure 17: Internal structure of the ULN2003 Darlington array (source: STMicroelectronics).

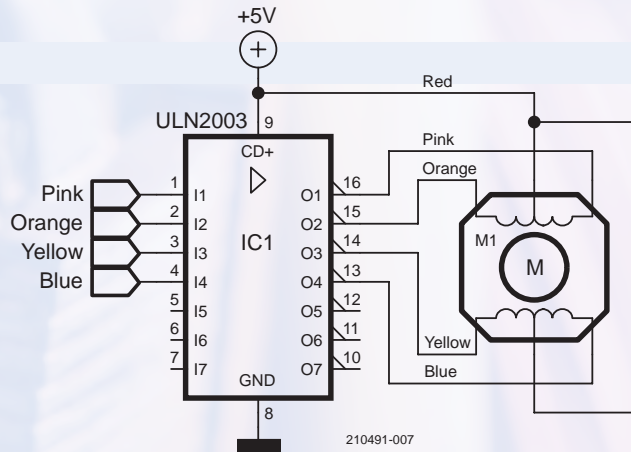


Figure 18: Circuit diagram with ULN2003 and stepper motor.

The nice thing about stepper motors is that they have high torque and do not need any brushes or commutators to carry the current to the rotor, as with DC motors.

Brushless DC Motors

If you combine the properties of a DC motor with the torque and low maintenance of a stepper motor, you get a brushless DC motor. Brushless DC motors are ideal for drive tasks in a wide variety of applications. In conventional DC motors, brushes provide a path for current to flow into the rotor windings. With every rotation, these brushes wear and distribute conductive dust inside the motor, so the brushes of DC motors must be replaced at regular intervals and the motors may need to be cleaned. Brushless DC motors do not need this sort of maintenance, and they deliver high torque in a small package. However, driving brushless DC motors is complicated, and faulty drive can lead to damage. For this reason, here we only give a very brief introduction to the theory of driving brushless DC motors.

The structure of a brushless DC motor is very similar to that of a stepper motor. It has a stator (the non-moving part) and a rotor. Two types of

rotor are possible: internal and external. Some brushless motors have integrated Hall sensors to determine the position of the rotor; motors with only three leads do not have this form of feedback. They require a bit more complicated circuitry and more computation. For sensorless types of brushless DC motors, the AVR444 application note [6] is a good starting point for learning more about the subject.

Figure 21 shows the structure of a brushless DC motor with an external rotor. In this case the external rotor is a permanent magnet. Inside the rotor there are six windings arranged in three pairs. Each pair is wound such that one winding forms the north pole and the other the south pole when current flows through them. The poles of the rotor tend to align with a pair of windings in which current is flowing. **Figure 22** shows how the rotor is pulled by the magnetic field of the windings, causing it to rotate. However, only one of the three pairs of windings is used at a time, so there is room for improvement. When winding pair A is attracting the permanent magnet in the first step, winding pair C can be used to repel the permanent magnet. In this way two windings contribute to the rotary motion, increasing the torque and therefore the performance of the motor.

Table 2: Sequence for a bipolar stepper motor.

	A	B	C	D
1.	1	1	0	0
2.	0	1	1	0
3.	0	0	1	1
4.	1	0	0	1



Figure 19: Joy-IT NEMA17-03 stepper motor (source: Joy-IT).

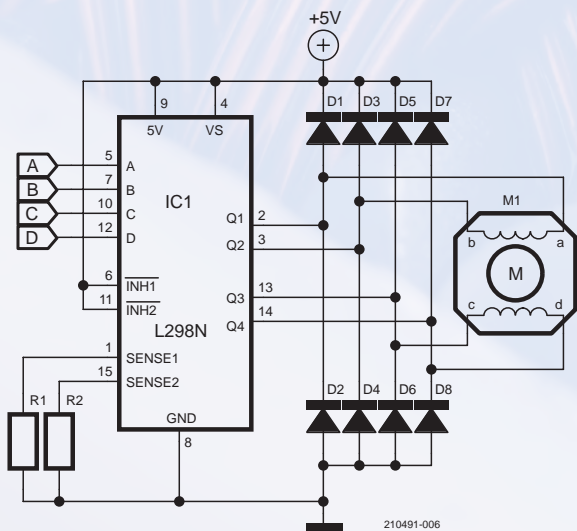


Figure 20: L294N as a stepper motor driver.

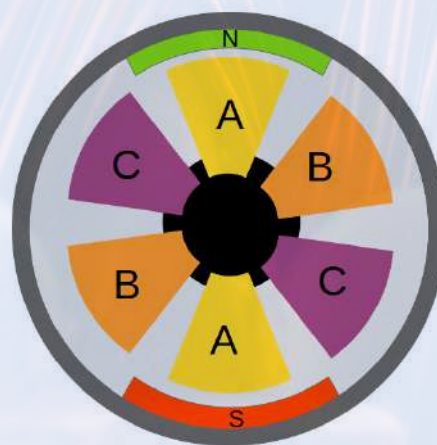


Figure 21: Structure of a brushless DC motor.

Figure 23 shows the rotation with two driven windings. The drive signals must have the right timing.

The windings are driven by three half-bridges. You have already learned about these in the section on DC motors. The circuit in **Figure 24** shows the theoretical layout of these three half-bridges, which are also available as ICs from various manufacturers. In order to drive the motor, suitable signals must be supplied to the inputs A, B and C. The NCP5901B has a voltage divider at each input, which biases the input pin to 2.0 V when it is not being driven by the MCU. In the range between 1.3 V and 2.7 V, the NCP5901B switches to diode emulation, which means free-wheeling mode, so the winding is not connected to ground or to the supply voltage of the motor. This allows the back electromotive force (BEMF) to be measured later in order to determine the rotor position.

Assuming that the rotor is rotating and no Hall sensors are installed in the motor, the BEMF of the non-driven winding can be utilised. This allows the position of the rotor to be determined, or more precisely the point in time when the rotor has travelled far enough that it is positioned between two winding pairs. The three winding pairs A, B and C are

marked in Figure 23. If a voltage is applied between A and B, the voltage between the two windings can be sensed over winding C. At the point in time when a voltage is applied between A and B, a timer is started and the voltage on winding C is monitored. When the voltage reaches half the supply voltage of the motor, the rotor has travelled half of the rotation distance between two winding pairs. This time can then be used to calculate when the next windings should be driven and with which polarity. As a result, the permanent magnet is kept in motion by the rotating magnetic field.

The speed of the motor can be controlled by PWM signals, which must be applied accordingly to inputs A, B and C of the driver IC. The hardware used in application note AVR444 is the ATAVRMC100 evaluation kit, whose description and schematics can be found on the Microchip web page [7]. More information is also available in application note AN1946 from ST [8] and application note AN12435 from NXP [9]. Driving brushless DC motors requires a bit more software than with stepper motors or simple DC motors. Rotor position detection, internal signal filtering and signal conditioning by the software make driving less intuitive than with conventional DC motors.



Figure 22: The rotor is pulled by a magnetic field.



Figure 23: The rotor is driven by two of the three winding pairs.

An H-Bridge for All Seasons

H-bridges are an important basis for DC drives, regardless of whether they are for stepper motors, motors with brushes or brushless motors. You do not necessarily need to build them with discrete components; for relatively small motors and low currents, you can use ready-made ICs if available. Driving brushless motors is more difficult, and you should consider whether you want to acquire experience in building controllers for brushless motors or would instead be satisfied with ready-made solutions. However, knowledge of the underlying theory certainly doesn't hurt. ◀

210491-01

Contributors

Concept and text: **Mathias Claußen**

Editor: **Jens Nickel**

Translation: **Kenneth Cox**

Layout: **Harmen Heida and Giel Dols**

Questions or Comments?

Do you have technical questions or comments about this article? Email the author at mathias.claussen@elektor.com or contact Elektor at editor@elektor.com.

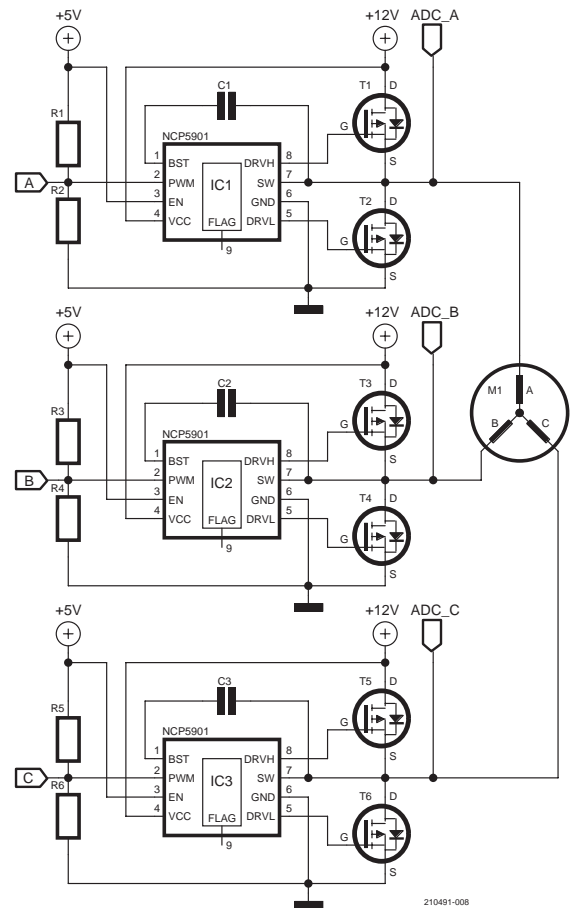


Figure 24: Brushless DC motor with three half-bridges.



RELATED PRODUCTS

- **Geekcreit 5pcs 5V Stepper Motor with ULN2003 Driver Board Dupont Cable (SKU 19783)**
www.elektor.com/19783
- **Cytron 3Amp 4-16 V DC Motor Driver (2 Channels) (SKU 18999)**
www.elektor.com/18999
- **Cytron 25Amp 7-58 V High Voltage DC Motor Driver (SKU 19062)**
www.elektor.com/19062
- **Cytron 10Amp 7-30 V DC Motor Driver Shield for Arduino (SKU 19063)**
www.elektor.com/19063
- **E-book: Power Electronics in Motor Drives (SKU 18517)**
www.elektor.com/18517
- **Book: Motors for Makers (SKU 18194)**
www.elektor.com/18194

WEB LINKS

- [1] Basic Kit for the Raspberry Pi Pico: <https://www.elektor.com/basic-kit-for-raspberry-pi-pico-pico-included>
- [2] 28BYJ-48 stepper motor: <https://www.elektor.com/geekcreit-5pcs-5v-stepper-motor-with-uln2003-driver-board-dupont-cable>
- [3] Cytron H-bridge motor driver boards: <https://www.elektor.com/catalogsearch/result/?q=cytron%20driver>
- [4] Jangeox blog: <http://www.jangeox.be/2013/10/change-unipolar-28byj-48-to-bipolar.html>
- [5] Joy-IT NEMA 17-03: <https://joy-it.net/en/products/NEMA17-03>
- [6] Microchip AN AVR444: <https://www.microchip.com/en-us/application-notes/an8012>
- [7] ATAVRMC100 Hardware User Guide: <https://ww1.microchip.com/downloads/en/DeviceDoc/doc7551.pdf>
- [8] STM AN1946: https://www.st.com/resource/en/application_note/cd00020086-sensorless-bldc-motor-control-and-bemf-sampling-methods-with--st7mc-stmicroelectronics.pdf
- [9] NXP Semiconductors AN12435: <https://www.nxp.com/docs/en/application-note/AN12435.pdf>



The **e**lektor **LAB** Team

Our Approach, Preferred Tools, and More

By Jens Nickel (Editor in Chief)

There are many electronics magazines around the world, but most of them just report on new developments and applications. Elektor is different. With our in-house Elektor Lab team of engineers, we are constantly developing, refining, and testing electronic projects and products.

For six decades, *Elektor* magazine has featured a healthy mixture of theory and practice. It is in our DNA. It all started more than 60 years ago with electronics enthusiast Bob van der Horst who was bored by dry datasheets and theoretical articles full of formulas and diagrams. Instead, he wanted to publish a magazine with circuits that readers could build and modify [1]. From those early days, most of the circuits and projects were built and tested before they were published in *Elektor* magazine. And as a result, the Elektor Lab quickly became famous. The work of the Elektor Lab continues to this day. Curious about the status of the Elektor Lab in 2022? Read on and join us!

What Is the Elektor Lab?

The Elektor Lab is both a location and a team. Each Elektor Lab member has a home

electronics workbench for soldering, reverse engineering, programming, and tinkering. Elektor also maintains a useable electronics workspace in its HQ that is stocked with handy lab equipment, components, and engineering tools. Lab team members can pop into the workspace at any time to test designs or collaborate on projects.

The actual team comprises engineers of various ages, backgrounds, and personalities. From the very beginning, the Lab has included analog guys interested fiddling around for months on big circuits as well as programming enthusiasts interested contributing fresh ideas. To put it simply, we have a diverse crew.

Some magazines only report about kits or new products. Not *Elektor*! We do much more. We have always endeavored to

uncover all the details of a project, including the source code, whether it was a new project from one of our engineers or it was from an enthusiastic reader's home lab. Why has this been Elektor's approach? Our reasoning is clear. We want our readers to be able to build the same projects, and we invite them to modify, improve, and expand everything we publish. Whether you are a professional engineer or a weekend maker, we want you to understand how everything works. So, it is not an overstatement to say that *Elektor* has been one of the long-time leaders in the open-source movement!

Most Elektor readers and community members will also recognize that there is no strict line between our Lab engineers and editorial team. All the engineers in the Lab (Clemens Valens, Luc Lemmens, Mathias Claussen, and Ton Giesberts) are writing articles, and all the editors including me and Content Director C. J. Abate also have fun contributing ideas and input to Lab projects.

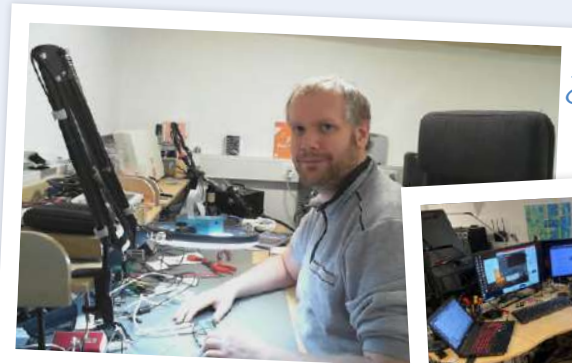
As we kick off 2022, we encourage everyone to get to know the Elektor Lab team. On these pages, you will find a short presentation about the Elektor Lab team members and some of the platforms and tools that we use. We invite you to contribute your ideas, feedback, projects, and articles!

The Elektor Lab Team



Based in France

Clemens Valens is an engineer who manages the Elektor Labs online platform. He holds a BSc in Electronics and an MSc in Electronics and Information Technology. Clemens started working for Elektor in 2008 as Editor in Chief of Elektor France, and he has also worked as an editor for Elektor UK/US and ElektorMagazine.com. Later, Clemens was head of Elektor's design labs in the Netherlands, Germany, and India. Today, he is Elektor's creative technologist responsible for the Elektor Labs community website where electronics enthusiasts can publish their work and interact with peers from all over the world. Besides contributing own projects and other articles to the magazine, he is also producing regular videos for Elektor TV and moderating webinars. His main interests are sound generation and signal processing.



Based in Germany

Mathias Clausen started as a trained IT systems electronics technician and earned a BSc in Electrical and Information Engineering and a MSc in Microelectronic Systems. After this, he worked as a project engineer for embedded development and focused on ARM Cortex-M-based architectures and real-time operating systems. Mathias joined Elektor in 2018 as part of the Elektor Lab team focusing mainly on software. Today, it is more than just software development in the background. Mathias writes articles about hardware and software development, creates videos for Elektor TV, and more. Want to learn about his home workspace? You can read about it online at Elektor [2]. You can see some of the instruments and tools Mathias is using for many daily tasks there.



Based in the Netherlands

Luc Lemmens started working for Elektor in March 1990 after his studies at the Technical University Eindhoven. At the time, he also published *Elex*, an electronics magazine aimed primarily at the novice electronics hobbyist. Luc designed and edited projects for both magazines. He has many interests, which means that he knows a little about a lot of topics in electronics. Of course, he has also written or edited software in a wide range

of programming languages — and, especially in his early days at Elektor, in assembly language. Nowadays, he usually limits himself to the Arduino IDE, which is perfect for most simple projects. Over the years, Luc has also written and translated technical text and even Elektor books. In his spare time, Luc likes to play with pinball machines, especially repairing and restoring both modern electronic machines and electromechanical ones (with relays and stepper units).

Ton Giesberts started working at *Elektuur* (now called *Elektor*) after his studies when they looked for someone with an affinity for audio. Over the years, he has worked mainly on audio projects. Analog design has always been his preference. Of course, projects in other fields of electronics are part of the job. One of Ton's mottos is: "If you want to have it done better, do it yourself." For example, a PCB design: for an audio project with distortion figures on the order of 0.001%, a good layout is crucial!



Based in the Netherlands



Recent Elektor Lab Projects and Products

The Elektor Lab team has prepared many electronics projects for Elektor. Sometimes a design is developed entirely by the Elektor Lab. Other times, our Lab team members adapt projects from external designers. And sometimes ideas are outsourced

for collaboration with our partners. The end result might be an article (or article series) or product that is sold in the Elektor Store. Below is a list of just a few successful Elektor Lab projects from recent years.

Clocks

Electronic clocks have always been popular with the readers of *Elektor* magazine. Also because they are the devices par excellence to give old displays a new application, as in the 6-digit Nixie Clock [3] and the Pinball Clock [4].

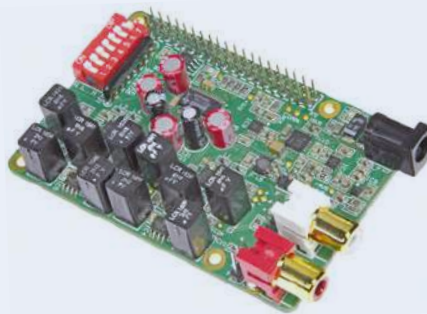


Weather Station with ESP32



Buy an affordable, complete set of weather station sensors and add state-of-the-art electronics and software to build your own weather station [5].

Audio DAC for Raspberry Pi



Raspberry Pi boards have audio outputs, but if you want real high-quality sound, an external high-end Digital-Analog-Converter is needed, like this board [6] designed by our audio wizard Ton Giesberts.

DIY LiPo Supercharger Kit



Need a rechargeable LiPo power supply kit for 5 V and 12 V output? Check out the kit the Elektor Lab developed in cooperation with GreatScott! [7]

New LCR Meter 50 Hz - 2 MHz



The LCR Meter 50 Hz - 2 MHz is a high-precision device for testing and measuring passive components. It can compete with — and even beat! — commercially available equipment [8].

Book

Mastering Microcontrollers Helped by Arduino



Clemens Valens's book, *Mastering Microcontrollers Helped by Arduino*, will familiarize you with the world of Arduino and you will learn how to program microcontrollers in general. Some of the proposed projects include a car GPS scrambler, a weather station, an infrared remote manipulator, and PID controller.

210545-01





Elektor Needs You!

In contrast to a lot of other magazines, articles and other interesting content does not flow one-way from *Elektor* to you. It also goes the other way around. We encourage our growing worldwide community of engineers and makers to contribute ideas, tutorials, tips and tricks, circuits and projects.

Can you teach others about a certain subdomain of electronics or software? Have you developed a nice electronics project? Do you have special area of technical expertise? Our readers may be very interested in that.

There are 2 ways to share tips, tutorials and projects:

1) You post something on Elektor Labs (see *Elektor Labs* textbox). Here you can immediately get in touch with other readers, maybe on the other side of the globe, to get positive

and critical feedback! You also have the chance to get backers for your project (please see the *Elektor Jumpstarter* textbox). On the Elektor Labs platform, ideas and questions are welcome!

2) You can send your article/project to the editorial team by mail: editor@elektor.com. You don't have to have a ready-made article text, but a circuit diagram and a short description is a must to let us know what you have in the works. Of course, we will not uncover any details of your project without your ok!

Elektor editors and Lab team members review all the contributions during our weekly internal meetings. If we select your article to be published in print or online, we will let you know, and we will discuss the next steps with you.

Elektor Labs Platform

The Elektor Labs online platform was developed for engineers, makers, and students who are passionate about electronics. It is a place where you can share your projects, participate in those posted by others, and discuss project development and electronics. Elektor Labs is open to everyone; the only requirement is a free Elektor ID.



Now in its tenth year, Elektor Labs has gathered more than 10,000 active users who have posted more than 2,000 projects and countless comments. The online platform is much more than just a free space where you can show off your skills. Not only can our team of electronics specialists assist you in successfully completing your project, but we can also help if you wish to turn it into a product to be sold in the Elektor Store. And, as every project posted at Elektor Labs is evaluated by our editorial team, Elektor Labs is also the place for posting an article proposition for Elektor Magazine.

www.elektor-labs.com

Elektor Jumpstarter

Elektor Jumpstarter is Elektor's take on crowdfunding. The online platform is intended to help innovators like you turn your electronics projects into products.

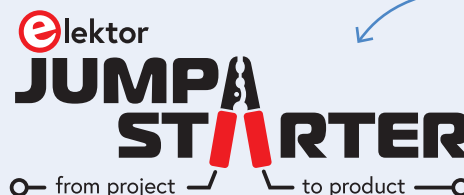


To apply, simply post a project at Elektor Labs and then click the Jumpstarter button. It will then be evaluated by Elektor. When a project is accepted, the community can start backing it.

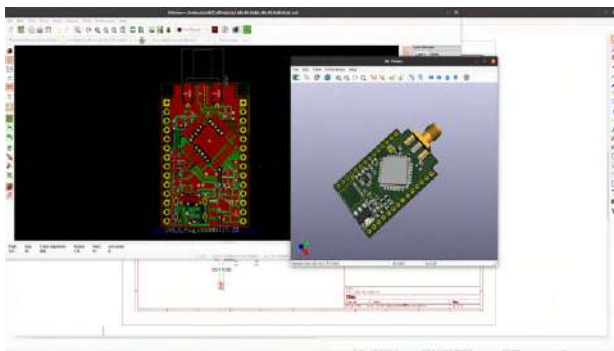
Backing a project does not involve any money. Instead, when someone "backs" a project, it indicates their interest in having the chance to purchase the final product if the Jumpstarter campaign reaches its goal (i.e., a certain production quantity at a predefined price). If the campaign succeeds, Elektor will produce the product and put it in the Store for anyone to buy.

In short, at Elektor Labs, our motto *Design, Share, Sell* really comes to life.

www.elektor-labs.com
www.elektormagazine.com/jumpstarter-info



Open-Source Design Tool: KiCad



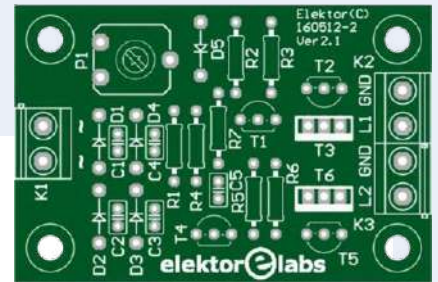
Elektor loves open-source solutions, so the Elektor Lab team members tend to use open-source tools. One example is KiCad, which we use (besides Altium and Eagle) to design schematics and PCBs. KiCad is free to use, open-source and runs on all three major operating system. This enables you to share your work with others so that they can also adapt or modify your creation as needed.

You can find online articles and videos about KiCad at the Elektor site: www.elektormagazine.com/tags/kicad

Elektor PCBs

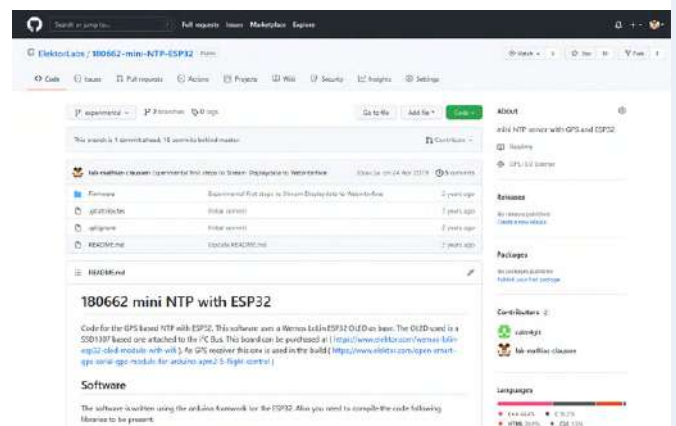
Elektor readers are used to finding green drawings of (mostly small) PCBs in the magazine. As a service for many years, almost all these PCBs were available in our Store for people who wanted to find and solder the components themselves. From most of these naked PCBs, we only sold a few pieces, and these figures have even decreased in the last years. We have to face the fact that the electronics world is changing. It makes no sense to develop a microcontroller or a small audio amplifier board (just to name two examples) from scratch and produce it in small quantities for (too) high prices. Today, makers and professionals alike are used (and are even expecting) to getting inexpensive modules with at least the most important components — as SMD blocking capacitors or pull-up resistors — already integrated. Time is money, so most electronic developers want out-of-the-box prototyping solutions. They expect to open the box, connect everything, and start working.

However, this doesn't mean that we do not sell any PCBs or kits anymore! We focus on the most promising and interesting projects, where we really invented something original which can't be ordered for a few bucks everywhere. For these projects, we have set up collaborations with partners who are experienced at producing small and also larger quantities when needed. Two of these partners are Eurocircuits and SIMAC (Joy-IT).



If a small board is needed but is not available via an Elektor-partner collaboration, we will offer the CAD files for free. Each reader can adapt the PCB or use it as it is, and order it via a PCB production service. We recommend www.elektorpcb4makers.com and www.elektorpcbservice.com, which we offer in cooperation with our partner Eurocircuits.

Elektor @ GitHub



The Elektor Lab uses the Elektor Labs platform (www.elektormagazine.com/labs) to host its own projects and projects from external authors. You will find the files (software and hardware) required for the projects there.

For larger projects that are continuously developed, managing the different versions can become confusing. This is where GitHub comes in, which brings a version management system so you can easily revert to a specific state or track changes made. GitHub also offers several other advantages, such as experimental alternate versions, issue trackers for suggestions and comments, or community-supplied patches in the form of pull requests.

Therefore, we will offer more and more software and also CAD data for download on GitHub (<https://github.com/ElektorLabs>). The corresponding web link can be found at the end of each project article.

For GitHub beginners, Clemens Valens from the Elektor Lab has created a video tutorial at on Elektor TV: www.youtube.com/watch?v=X5e3xQBef8.



Follow Us on Social Media

Elektor Lab members and editors are easily accessible on social media. Feel free to follow us and reach out on Twitter, Facebook, and YouTube.

Twitter

Mathias Claussen: <https://twitter.com/elektormathias>
 Clemens Valens: https://twitter.com/clemens_elektor
 Elektor: <https://twitter.com/Elektor>
 C. J. Abate: https://twitter.com/elektor_us

YouTube

ElektorTV: <https://www.youtube.com/user/ElektorIM>

Facebook & Instagram

Elektor Labs (FB): <https://www.facebook.com/ElektorLabs/>
 Elektor Labs (IG): <https://www.instagram.com/elektorlabs>



WEB LINKS

- [1] "Elektor @ 60", ElektorMag May/June 2021 : <http://www.elektormagazine.de/210025-01>
- [2] "An Elektor Engineer's Workspace for Embedded Software Development", Elektormagazine.com: <https://www.elektormagazine.com/articles/elektor-workspace-embedded-software-development>
- [3] 6-digit Nixie Clock: <https://www.elektormagazine.com/magazine/elektor-146/58526>
- [4] Pinball Clock: <https://www.elektormagazine.com/magazine/elektor-88/42431>
- [5] Weather Station with ESP32: <https://www.elektormagazine.com/magazine/elektor-70/42351>
- [6] Audio DAC for Raspberry Pi: <https://www.elektormagazine.com/magazine/elektor-201707/40496>
- [7] DIY LiPo Supercharger Kit: <https://www.elektormagazine.nl/labs/diy-lipo-supercharger-kit-by-greatscott>
- [8] New LCR Meter 50 Hz - 2 MHz: <https://www.elektormagazine.com/magazine/elektor-159/59096>

Advertisement



Delivering more

The widest selection of semiconductors and electronic components in stock and ready to ship™

[mouser.com](https://www.mouser.com)





Raspberry Pi as a KVM Remote Control

Pi-KVM Software Test

By **Mathias Claussen** (Elektor)

KVM stands for keyboard, video, mouse, and whoever has access to it can remotely control a computer. Using the clever Pi-KVM software and a Raspberry Pi 4, you can inexpensively control a PC and other devices via the Internet without having to install software on the remotely controlled computer. Pi-KVM also enables the provision of virtual disks, so that a computer can not only be controlled and maintained remotely, but a complete reinstallation is possible.

During the pandemic, I received information that my parents still had computers running Windows 7 (some were even Windows XP) at their house. Because of the security gaps, which can no longer be fixed, these computers urgently needed to be updated. Unfortunately, an update was not enough for these machines. They required a complete reinstallation, but driving there was not an option, because of the pandemic and more than eight hours of driving time.

Anyone who has ever tried to guide someone

through dozens of installation steps with a mixture of phone call and video chat can easily imagine that such a thing is not a good idea for an operating system installation and data backup. Remote access is what's needed. And that's exactly what is made possible with a Raspberry Pi as a KVM remote control.

At this point in the text, you can replace *parents* with *customers*, *computers* with *machine controls*, and *new installation* with *error analysis*. A pair of eyes with which one can look at a system remotely, independent

of hardware, makes it possible to avoid long trips to a site. It is also possible to remotely operate devices and measuring instruments that were never intended for this purpose by the manufacturer. **Figure 1** illustrates the use of Pi-KVM for remote control of a PC. A machine could be remotely maintained as shown in **Figure 2**.

The Pi-KVM Alternative

There is remote maintenance software, isn't there? If you are using a Raspberry Pi or a PC, you will surely have used a tool like VNC, AnyDesk or TeamViewer. Besides these, there are quite a few other solutions for remote maintenance and control. All these solutions are inexpensive or even free of charge for the private user. To help someone set up new software or create social media posts, something like this is a quick and easy solution. However, these helpers all require a running operating system with an Internet connection.

But what if the computer no longer boots or can no longer establish a network connection? Or if the operating system has to be

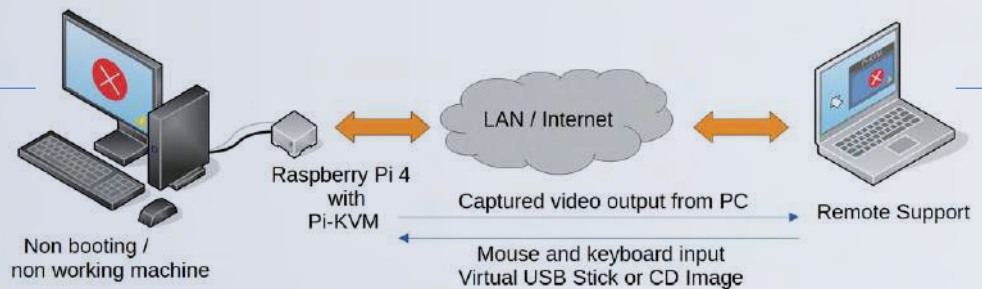
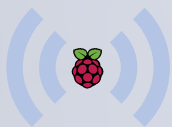


Figure 1: Remote support with Pi-KVM on remote PC.

reinstalled? That's the point at which someone with appropriate expertise needs to be on site to get the operating system to boot again and establish a network or Internet connection. This also applies to reinstalling an operating system. **Figure 3** shows the BIOS message of a PC that could not find bootable media.

Pi-KVM works independently of the computer to be controlled and transmits the video image that the graphics card outputs via the network to a second system that only needs to run an HTML5-capable web browser. There, not only the image output of the remotely controlled computer can then be displayed; the mouse movement and keyboard inputs are also returned to the computer to be controlled. This allows you to remotely control the computer as if you were sitting in front of the computer's screen, mouse and keyboard. **Figure 4** shows the BIOS message of the PC now comfortably in the browser of a remote computer. Access to the BIOS (as shown in **Figure 5**) can also be conveniently controlled via a remote browser, so it is not absolutely necessary to be near the PC.

Virtual Disks

Something that gets complicated over longer distances (usually 2 m is enough) is the insertion of USB bootsticks, or rotating data media such as DVD or CD. You could send a bootstick to the PC to be controlled remotely and hope that someone onsite will insert it correctly. Pi-KVM offers the option of virtual disks (i.e., it can emulate a USB stick or a USB-CD-ROM drive). To do this, you can simply upload and mount the appropriate disk image via a browser. **Figure 6** shows the virtual disk menu. Currently, CD emulation is limited to ISO files, so they must not be larger than 2 GB. For ISO files larger than 2 GB (**Figure 7**), hybrid ISOs must be available (i.e., those that could also be written to a USB stick). These can then be used with the *Flash* emulation type and appear as USB mass storage.

Measurement equipment

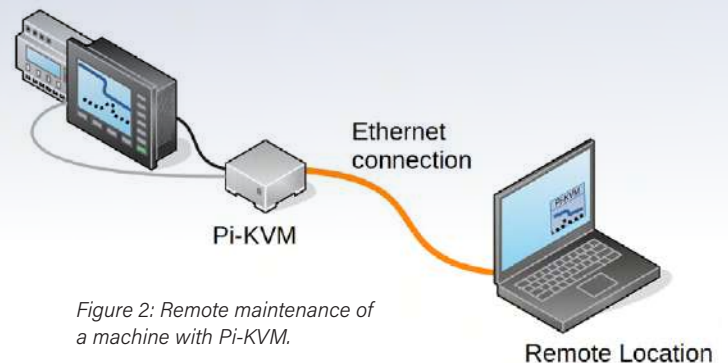


Figure 2: Remote maintenance of a machine with Pi-KVM.



Figure 3: Error message at boot.

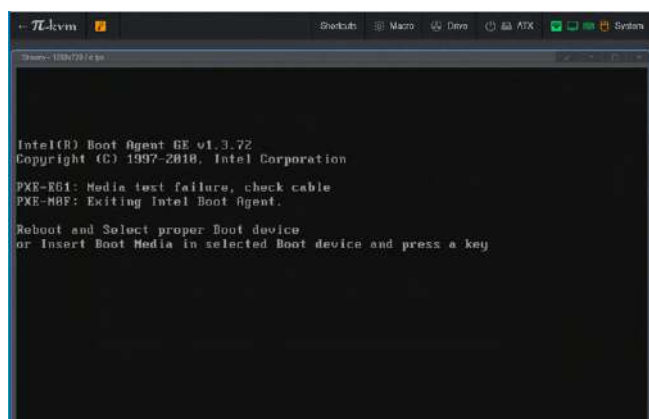


Figure 4: Error message in the browser with Pi-KVM.

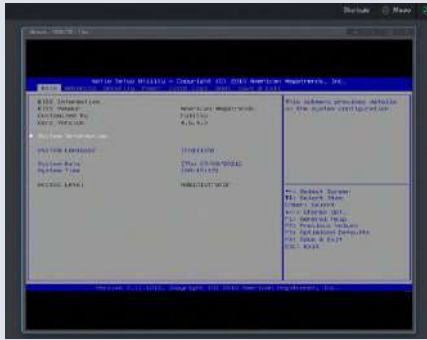


Figure 5: BIOS settings remotely.

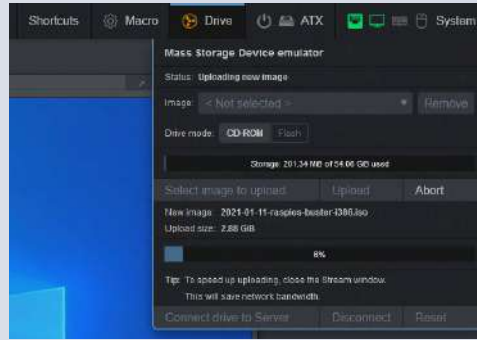


Figure 6: Virtual disks.

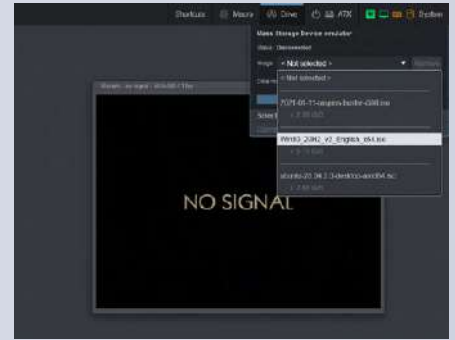


Figure 7: ISOs over 2 GB as mass storage.

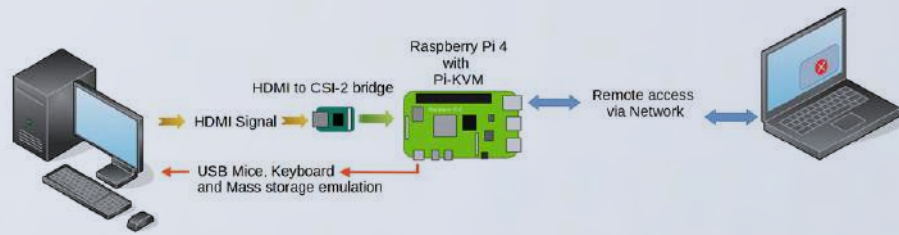


Figure 8: Data flow with HDMI-CSI bridge.

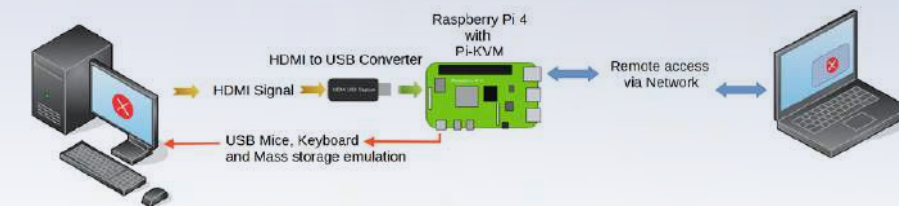


Figure 9: Data flow with HDMI-USB dongle.



Figure 10: Raspberry Pi 4 2 GB version.

An Affordable, Open-Source Solution

Doesn't something like Pi-KVM already exist? The functions provided by a Pi-KVM are also offered by other devices from other manufacturers, but usually much more expensive than the Pi-KVM solution and with closed-source firmware. With these devices some comfort functions are then also subject to an extra charge. The Pi-KVM comes as an open-source project and can be assembled with common hardware. In the end, you have to budget about €100 for the DIY solution.

There are two ways to build a Pi-KVM, one with an HDMI-CSI bridge and one with a USB-HDMI dongle. **Figure 8** and **Figure 9** show the data flow through the Raspberry Pi 4 with the respective solution. At this point, we would like to present the required components and give a few hints about how to build them.

The Necessary Hardware

The hardware needed for the Pi-KVM is quite simple:

- A Raspberry Pi Model 4 (2 GB RAM or more) (**Figure 10**)
- One Micro SD card (16 GB recommended)
- One USB power supply (5 V/3 A)
- Optional housing
- HDMI to CSI Bridge or USB HDMI Capture Dongle (**Figure 11**)
- A USB Y-cable (**Figure 12** and **Figure 13**)

While the first three components should be available or easy to get for most readers, the last three are a bit more difficult to get. For the HDMI-to-CSI bridge (**Figure 14** shows two available variants), on the other hand, you have to search a bit for a European source or resort to one from the Far East. A USB

HDMI capture dongle can be ordered for a few Euros from relevant mail-order companies. Even though the price of these dongles may be tempting, they have some limitations in terms of stability or supported resolutions. The HDMI-to-CSI bridge is the more stable and compatible choice.

For the connection between the Raspberry Pi 4 and the computer to be controlled, the USB-C port of the Raspberry Pi 4 is used in USB OTG mode. This allows the Raspberry Pi 4 to behave like a keyboard, mouse or mass storage device towards a PC. For this, the Raspberry Pi 4 is connected to the computer with a USB-A to USB-C cable and would also be supplied with power via this. Since 3 A @ 5 V is required here, stable operation cannot be guaranteed, so an additional power must be provided by a power supply unit.



Figure 11: HDMI-to-CSI bridge and USB-HDMI capture dongle.



Figure 12: USB A-Y cable.



Figure 13: USB-A to USB-C cable.



Figure 14: Two variants of the HDMI-to-CSI bridge.

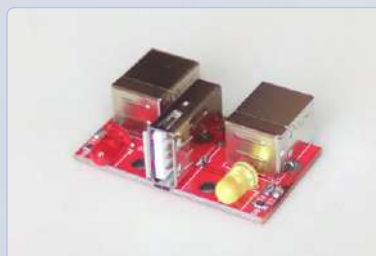


Figure 15: Assembled adapter board.



Figure 16: Two robust USB-B sockets.



Figure 17: Micro SD card with 16 GB memory.



Figure 18: Finished setup with HDMI-to-CSI bridge.



Figure 19: Finished setup with HDMI USB dongle.

Since the USB-C port is used in OTG mode, you could feed 5 V into the Raspberry Pi 4 via the 40-pin header. However, these 5 V would then also supply power to the computer connected to the USB-C port. This is something that quite a few computers do not like and acknowledge with malfunctions. One solution is to use a Y-cable that splits the data and the power supply.

So that two USB cables do not have to be soldered together to fit, the Elektor Lab has routed a small circuit board (**Figure 15**). Two USB-B sockets (**Figure 16**) allow one connection to a power supply and one connection to a computer without feeding 5 V backwards. Both sockets have in common that a LED signals the presence of the 5 V bus voltage. The board is available as a KiCAD project and can be downloaded from the Elektor GitHub repository [1]. However, the board has not yet been tested extensively, so rebuild at your

own risk. You can find a component list and the circuit diagram at the end of the article.

Software Installation

You can download the Pi-KVM software from the developer's homepage [2]. An SD card with 16 GB memory (**Figure 17**) is recommended for the installation. If an HDMI-to-CSI bridge is used, the *Raspberry Pi 4 v2 platform image* can be used for the HDMI-to-CSI bridge; if a USB-HDMI dongle is used, use the HDMI-USB dongle image for the Raspberry Pi 4 v2 platform.

The downloaded file must be unzipped with a program like 7-Zip and then written to an SD card with a tool like the Raspberry Pi Imager. This completes the basic installation and you can start assembling the hardware. If you want to take a look at the source code, you can do so in the GitHub repository [3] of the Pi-KVM.

Hardware Assembly

If the CSI-to-HDMI bridge is used, it is unfortunately a bit more difficult to find a suitable case. Either someone has already built such a case or you have to help yourself with a little FreeCAD and a 3D printer. For this article, the parts are therefore installed using flying wires.

For the setup with HDMI-to-CSI bridge, you need a corresponding board and a Raspberry Pi camera cable. A suitably prepared SD card with the latest image for Pi-KVM must be inserted into the Raspberry Pi 4 and the HDMI-to-CSI bridge connected to the camera port. The USB-Y cable must be connected to the USB-C port of the Raspberry Pi 4. With this, the Pi-KVM is ready for use (**Figure 18**).

For the variant with HDMI USB dongle, this must be connected to one of the USB 2.0 ports of the Raspberry Pi 4B. If you use one of the



blue USB 3.0 ports, the shown USB dongle is not recognized (**Figure 19**).

First Test Run

Once all parts have been assembled and, if possible, stowed away in a housing, nothing stands in the way of the first test run. I used a second Raspberry Pi 4 as a test object, which has been doing its job as a camera on my desk for some time (**Figure 20**). (The article about the Raspberry Pi 4 camera can be found in *Elektor* Sep/Oct 2021 [4].) The Raspberry Pi 4 running Pi-KVM is connected to a wired network.

After the first start of Pi-KVM, it takes a moment because the partition is adjusted to the size of the SD card. If this process is finished, Pi-KVM tries to get an IP address via DHCP. It might be necessary to check in the router or DHCP server which IP the Pi-KVM got. With a web browser the interface of the Pi-KVM can then be called.

After booting, you can log in to the interface with the user name *admin* and the password *admin*. Under the item *KVM* the screen of the second Raspberry Pi 4 appears and both


mouse and keyboard can now be operated remotely.

Figure 21 shows the web interface with the desktop of the remote Raspberry Pi, taken with the HDMI-to-CSI bridge. **Figure 22** shows the same scene, although an HDMI-to-USB dongle was used here. The image quality has decreased and green edges are visible as artifacts.

For additional configuration and options, it is worth taking a look at the Pi-KVM homepage or the GitHub repository. Besides mouse and keyboard, a mass storage is available as already mentioned. The Raspberry Pi 4 with Pi-KVM can emulate either an optical drive or a USB stick. To do this, the appropriate image can be copied to the Raspberry Pi 4 via the web interface (**Figure 23**).

Outlook

With the Pi-KVM software and a few inexpensive components, you can turn a Raspberry Pi 4 into a remote control for computers and other equipment that can be operated with a mouse and a keyboard. If you don't want to assemble the components yourself,

take a look at the Pi-KVM homepage. There they are working on a finished HAT for the Raspberry Pi 4 and a matching case. Pi-KVM and its developers are currently funded by donations through Patreon or Paypal. If you like the project, the developer would surely appreciate some recognition. 

200523-01

Contributors

Design and text: **Mathias Claußen**

Editor: **Jens Nickel**

Layout: **Harmen Heida**

Questions or Comments?

Have technical questions or comments about this article? Email the author at mathias.claussen@elektor.com or contact Elektor at editor@elektor.com.



Figure 20: Raspberry Pi camera (*Elektor* Sep/Oct 2021 [4]).



RELATED PRODUCTS

- > **Raspberry Pi 4 B (4 GB RAM) (SKU 18964)**
www.elektor.com/18964
- > **Raspberry Pi HDMI to CSI-2 Adapter Board (supports up to 1080p25fps) (19707)**
www.elektor.com/19707

WEB LINKS

- [1] KiCad files on GitHub: https://github.com/ElektorLabs/200523-Raspberry_Pi_4_with_PiKVM
- [2] PiKVM Homepage: <https://pikvm.org/>
- [3] PiKVM Github Repository: <https://github.com/pikvm>
- [4] Mathias Claußen, "Imaging and Video-Streaming with a Raspberry Pi 4," *ElektorMag* 9-10/2021: www.elektormagazine.com/200582-01



COMPONENT LIST

Resistors (0.25 W)

R1,R2,R3 = 100 k

R4,R5 = 1 k

Capacitors

C1,C2,C3 = 100 n

Semiconductors

D1,D2 = LED RED 5 mm

Others

K1,K2 = Connector USB B

K3 = Connector USB A

PCB = 200523-1 V1.0

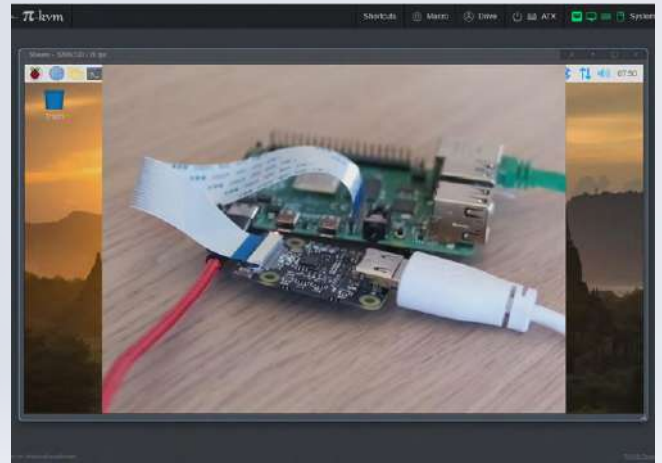
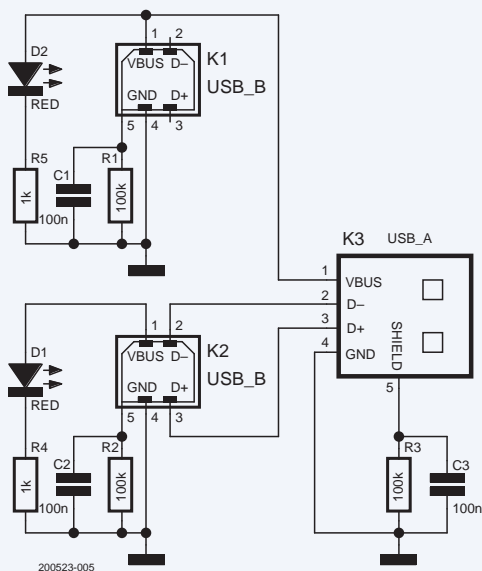
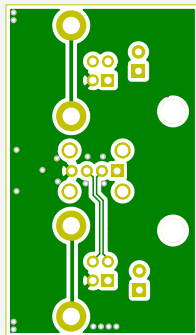
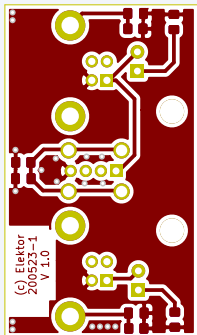


Figure 21: Desktop of the Raspberry Pi camera captured with HDMI-to-CSI bridge.



Figure 22: Desktop of the Raspberry Pi camera recorded with HDMI USB dongle.

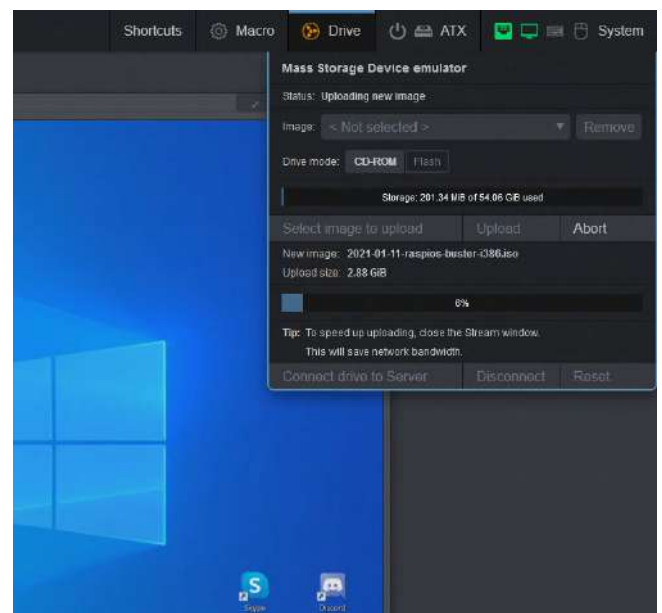


Figure 23: Mass storage emulation on the web interface.

IQaudio Codec Zero

A Sound Card for the Raspberry Pi Family

By Mathias Claußen (Elektor Lab)

Looking for a small and compact sound card for a Raspberry Pi? Check out the IQaudio Codec Zero, which has a Zero form factor.

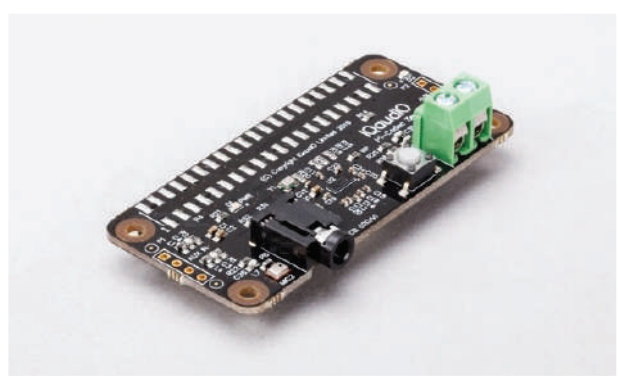


Figure 1: IQaudio Codec Zero (Source: Raspberry Pi Foundation).

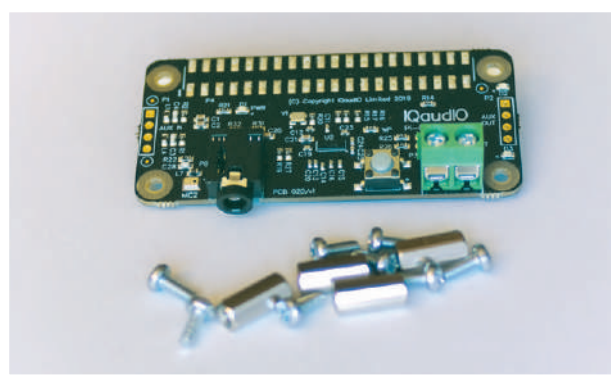


Figure 2: Contents of the IQaudio Codec Zero package.

If you only need a digital sound output via HDMI, you don't need a sound card. A sound card becomes interesting if you want to output analog audio in good quality or if you also want to process analog signals with the Raspberry Pi. The A and B models of the Raspberry Pi still have a 3.5-mm jack connector that can output analog audio; however, the quality is not the best and not really suitable for listening to music.

The IQaudio Codec Zero (**Figure 1**) offers an integrated 1.2 W amplifier for an 8 Ω speaker, an integrated MEMS microphone and an analog AUX input and output with line level. This makes projects such as a smart speaker, a VoIP phone, or a custom jukebox quite easy to implement. **Figure 2** shows the hardware and the included mounting hardware. The heart of the IQaudio Codec Zero is the DA7212 from Dialog Semiconductor. **Figure 3** shows the blocks and signal routing within the DA7212. The IC is a 24-bit codec with up to 96 kHz sample rate. Integrated is a 5-band equalizer as well as an ALC and a noise gate. The documentation for the IQaudio Codec

Zero can be found on the page [1] of the Raspberry Pi Foundation.

IQaudio Codec Zero: Almost Plug & Play

The IQaudio Codec Zero has an EEPROM according to the Raspberry Pi HAT specification, so a Raspberry Pi should automatically recognize the board and set it up appropriately. On a Raspberry Pi 3B+, this works fine; the current Raspberry Pi OS (32 bit) recognizes the board and sets it up appropriately. **Figure 4** shows the IQaudio codec in the sound settings. However, there are still a few steps necessary that are not directly described in the IQaudio Codec Zero manual.

First, the output volume should be reduced to a minimum. After the internal audio codec of the Raspberry Pi has been deactivated according to the instructions, the IQaudio now remains in the system as the default sound card. If a terminal is opened now, a speaker test can be started with `speaker-test -t wav -c 1`. However, the speaker remains mute. It is necessary to adjust a

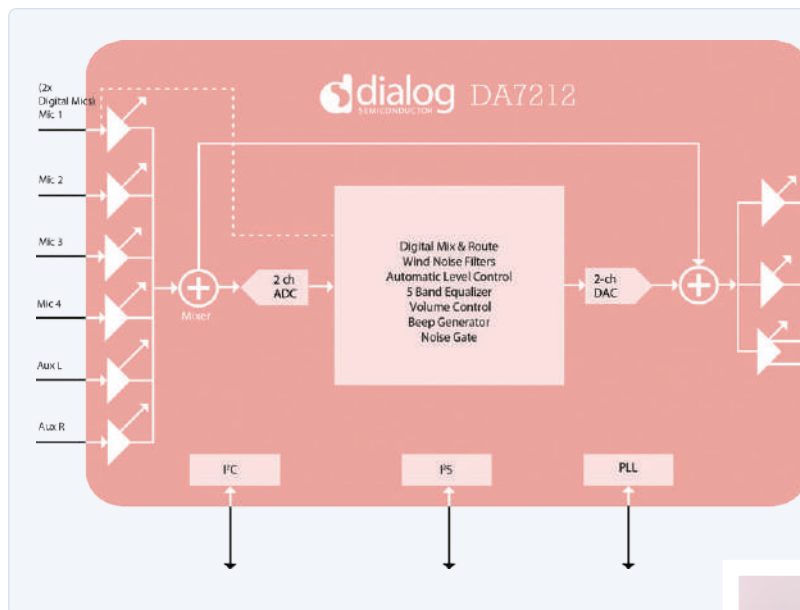


Figure 3: Block diagram of the DA7212 (Source: Dialog Semiconductor).

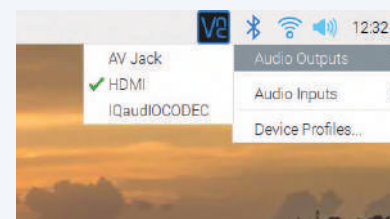


Figure 4: IQaudio Codec Zero settings.

few parameters in the *Advanced Linux Sound Architecture* (ALSA) settings. The fastest way is to reload the provided settings from IQaudio. To do this, a Git repository is cloned via terminal with `git clone https://github.com/iqaudio/Pi-Codec.git`. In the terminal change to the created folder *Pi-Codec* and execute the command `sudo alsactl restore -f IQaudIO_Codec_Playback_Only.state`. After that, `speaker-test -t wav -c 1` should make a *front left* sound from the speaker. From now on all sounds will be output via IQaudio Codec Zero. A quick and easy installation!

Microphone, AUX-In and AUX-Out

With the built-in microphone, the Raspberry Pi can record audio. This way, a smart speaker can be realized — e.g., with *voice2json* [2]. Or you can use the Raspberry Pi in conjunction with *Edge Impulse* [3] to record data. The MEMS microphone on the board won't win any awards in the "recording quality" category, but it does its job. If you are unhappy with the sound, you can connect an external microphone. The predefined *.state* files in the cloned Pi codec folder help to get the appropriate settings for recording via microphone. Using `sudo alsactl restore -f IQaudIO_Codec_OnboardMIC_record_and_SPK_playback.state` selects the microphone as the recording source. For a test recording, `arecord --device=hw:1,0 --format S16_LE --rate 44100 -c2 test.wav` can be used in the terminal to start a test recording with the microphone and end it with `CTRL+C`. This can then be played back with `aplay test.wav`. If you want to connect an external audio source, you can solder suitable jacks to the IQaudio Codec Zero. **Figure 5** shows AUX In with RCA jacks. To select AUX In as recording source, you have to type the following in a terminal:

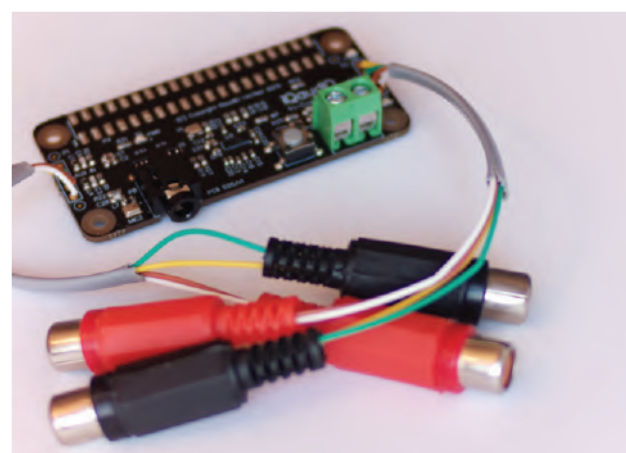


Figure 5: AUX In and Aux Out connected.

```
sudo alsactl restore -f IQaudIO_Codec_AUXIN_record_and_HP_playback.state
```

You can also see the AUX Out on the picture, which allows the IQaudio Codec Zero to act as a feed to an amplifier or active speakers.

New Life for an Old Radio

Something that has been on my to-do list for too long is an old Philips radio (**Figure 6** and **7**). All that's left of the interior is the speaker, and there's room for new hardware. Inside, a low-power Raspberry Pi would do well, but an amplifier is needed for the speaker. It's a good thing that the IQaudio Codec Zero has an integrated 1.2-W amplifier, so that a small office can be sounded wonderfully.



Figure 6: Old housing of a radio.



Figure 7: There is only one speaker left.

For music playback, anything can be used that outputs sound in the direction of ALSA under Linux. But if the MPD or a special distribution like Volumio will be used, is not decided yet. I don't have a completely finished concept for the radio yet, but one of the hurdles, a codec with a small amplifier, is already out of the way. So, with time, the inner workings will fill up piece by piece with hardware. There is still a lot to do and plan for this project.

A Solid Base

The IQaudio Codec Zero offers Aux IN, Aux Out as well as an integrated microphone and can drive a small speaker with 1.2 W. For the price of about €20, this is a compact solution that can be used on a Raspberry Pi Zero as well as on a Raspberry Pi 3B or Raspberry Pi 4B. For streaming players, VoIP phones or smart speakers, the IQaudio Codec Zero provides a solid base. If you want to experiment with audio and streaming on the Raspberry Pi and Linux, you should consider the IQaudio Codec Zero. ◀

210535-01

Questions or Comments?

Do you have technical questions or comments about this article? Email the author at mathias.claussen@elektor.com or contact Elektor at editor@elektor.com.

Contributors

Text and pictures: Mathias Claußen
Editors: Jens Nickel and C. J. Abate
Layout: Giel Dols

WEB LINKS

- [1] IQaudio Codec Zero:
www.raspberrypi.org/products/iqaudio-codec-zero/
- [2] Voice2Json: <http://voice2json.org>
- [3] M. Claussen: Image Processing with the Nvidia Jetson Nano (Part 2) - Image Recognition Using Edge Impulse, Elektor 11-12/2021 : www.elektormagazine.com/210318-B-01



RELATED PRODUCTS

- IQaudIO Codec Zero - Sound Card for Raspberry Pi Zero (SKU 19541)
www.elektor.com/19541
- Raspberry Pi Zero WH (with pre-soldered header) (SKU 18567)
www.elektor.com/18567
- Raspberry Pi 3 B+ (SKU 18452)
www.elektor.com/18452
- Raspberry Pi 400 - Raspberry Pi 4-based PC (EU) + FREE GPIO header (SKU 19431)
www.elektor.com/19431

The PiKVM Project

and Lessons Learned

Interview with Maxim Devaev (Developer, PiKVM)

By Mathias Claußen (Elektor Lab)

PiKVM software is a Raspberry Pi 4-powered tool for remotely accessing a PC, server or other machine over a network connection. We were able to ask the developer of the PiKVM a few questions about his project.

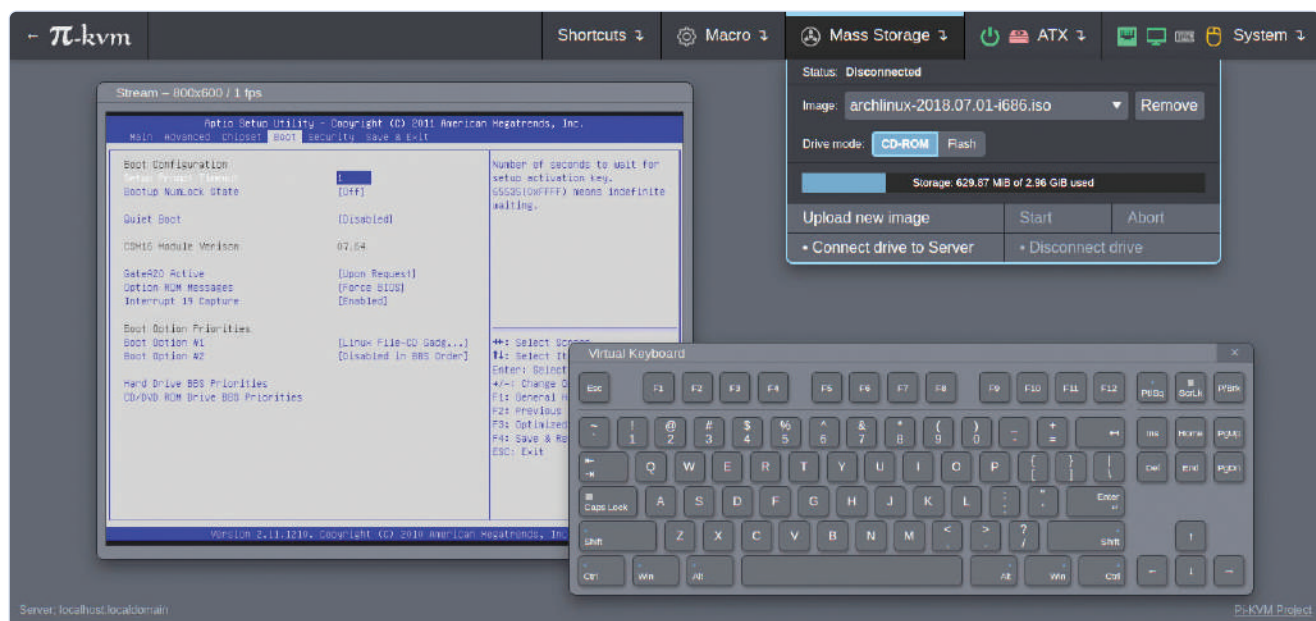


Figure 1: UI of PiKVM.

The PiKVM software [1] makes it possible to turn a Raspberry Pi 4 into an inexpensive remote control for other computers. Locally, the Raspberry Pi is connected to the computer to be controlled. It grabs the monitor image and emulates a USB mouse, keyboard and flash drive. The signals are then made accessible via the Internet by the PiKVM software [2] running on the Raspberry Pi

— all that is needed on the side of the controlling computer is a web browser. The PiKVM project (UI shown in **Figure 1**) is open-source and the cost of the required hardware is less than €100 for the DIY version, a fraction of commercial products. This raises a few questions that we put to the developer of the PiKVM, Maxim Devaev.

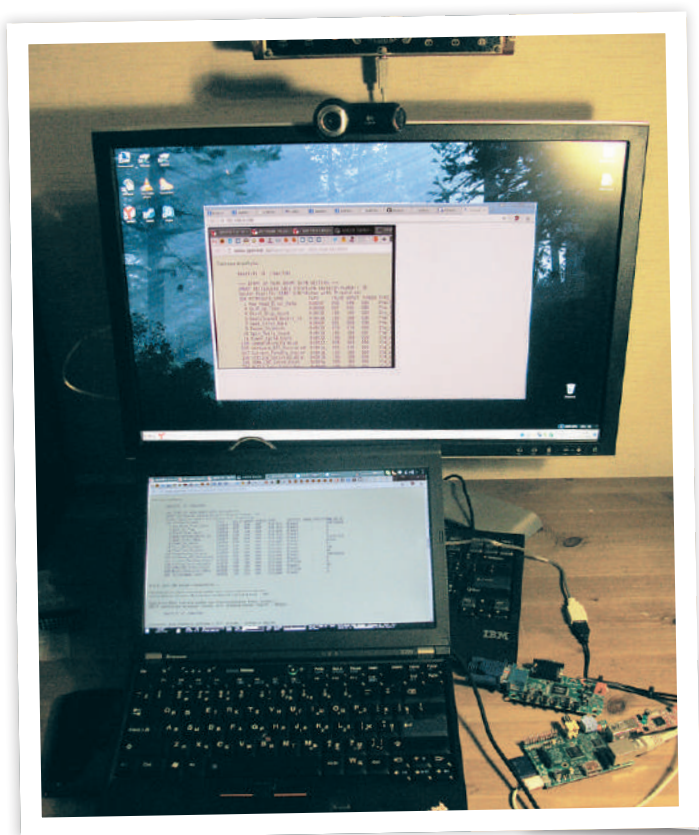


Figure 2: Prototype of PiKVM using a Raspberry Pi 1
(Source: Maxim Devaev).

Claußen: Let's begin with your background. Tell us about how you became interested in electronics. Did you study at university?

Devaev: Sure, but it would be wrong to call me an electronic specialist. I know how to hold a soldering iron, of course, and I can design a simple wiring diagram and assemble it without burning the whole thing. But I'm mainly a software engineer. Electronics is my secondary and far less developed skill.

I became interested in technics when I was a child and I have been reading a lot of different science and tech literature since. Getting my first PC affected the contents of my bookshelves a lot, of course, as they became much heavier with all the computer science books.

A couple years later, when it was time for me to choose university, I went for information security, but abandoned shortly after, as the curriculum disappointed me. Russian higher education is very formal: it's often more important that you don't fail the dozen accompanying humanitarian subjects.

So, I took the self-education route and got a job as a software developer. I worked, I worked, I worked, I improved, and some time later I landed in Yandex (Russian Google of sorts). There I settled down for nine whole years. I was developing distributed systems, solving big infrastructure monitoring and management problems and all that jazz. It might not be the most interesting thing, but it is what it is.

Claußen: How did you come up with the idea to start the PiKVM project?

Devaev: It's a funny story, actually. One time I went to visit my parents 1400 kilometers from home. My home server, which I always keep on, died as I was eating my mom's cake. I was annoyed and had to ask my friend to go to my place and reboot it, and decided to buy a KVM over IP after that. I went online, and saw ... \$500? Seriously? What's in that thing to cost \$500? And then I read some reviews and realized that those \$500 get you an overpriced piece of hardware paired with awful buggy proprietary software with licensing restrictions. And there were no good devices on the market: they were all equally bad. I don't know what got me, but I decided I was not going to pay that amount of money as a matter of principle. If you want something good, do it yourself. And the rest is history.

Claußen: Why did you use a Raspberry Pi for the project?

Devaev: Raspberry Pi just came in handy. It was fresh to the market and was a perfect fit for a small embedded system. Basically it could have been any other ARM computer, but the price, prevalence and great manufacturer support played a crucial role. I didn't want to spend any time debugging noname devices, creating custom kernels, etc. — so, Raspberry Pi allowed me to focus on my task.

Besides Raspberry Pi, I needed a video capture device and a way to emulate a keyboard (no mouse in mind at that point). At that time, there was no cheap VGA capture devices on the market, so I went for a little hack: I used an RCA-USB dongle that was usually used to digitise old VHS tapes, and fed analog video from a cheap VGA-RCA converter (Editor's note: see **Figure 2**). Video quality was horrible but good enough to set up BIOS and use a console. As for the keyboard, I bought a PS/2 to USB converter and emulated PS/2 signals using GPIO. It was cheap, primitive, but it worked. Eventually, I threw all this hardware away and went for a popular HDMI bridge for video and switched my PS/2 converter for an Arduino with custom software.

Claußen: How did the appearance of the Raspberry Pi 4 influence the project?

Devaev: Besides better performance and a way to improve video quality, Pi 4 had two USB controllers. I could use one to emulate a keyboard and a mouse on kernel level which allowed me to ditch

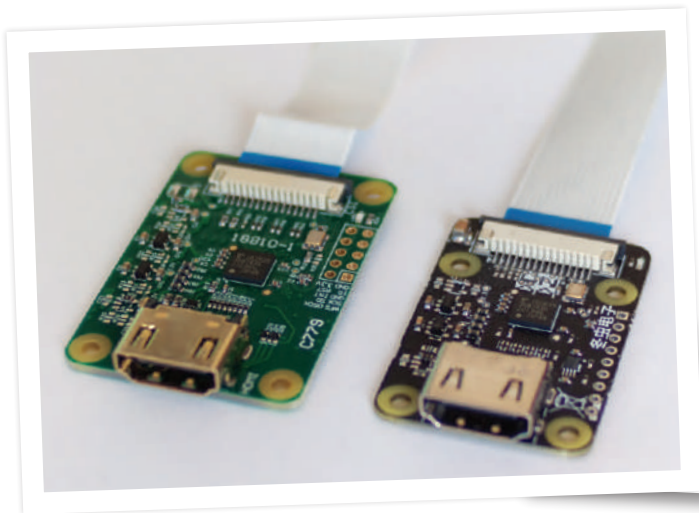


Figure 3: HDMI to CSI Bridge in two variants.



Figure 4: HDMI to USB Dongle.

the Arduino. I also managed to build a virtual CD drive so I could remotely reinstall an OS on the server. It was a moment when PiKVM came before most of the existing commercial devices in price and functionality.

Claußen: Did you start the PiKVM as a part-time project or hobby?

Devaev: It was a pet project of sorts. I enjoyed doing it, humouring my perfectionism. I never thought anyone besides me would need it until I stumbled upon a similar half-abandoned project on GitHub (DIY-IPMI). Its author had followed the same route, but they dropped it on the proof-of-concept stage. I became excited. I began advertising my project and committing like crazy, solving one issue after another, adding more and more features and spending thousands of hours on the project. It was great to see how many people began using my software. I had been developing only small utility scripts or heavy enterprise software before.

Claußen: Are there lessons learned from this project that you would like to share with other developers?

Devaev: Okay, here's a story. Web interface is a main way to use PiKVM, but there's a self-developed VNC server as an alternative. That's cool, right — use a VNC client to set up BIOS! I used MJPEG to transfer video both for web and VNC. It's a simple but very bandwidth-ineffective format. For the browser, I could use H.264 and WebRTC, but for VNC there was no way to use full-fledged differential encoding.

I manned up and wrote to the RFBProto (de facto VNC standard description) and TigerVNC repository maintainer with a proposal: let's create new encoding, it would be great for users with low bandwidth. We discussed details for a long time and finally came up with a format that suited us all. Then I went to IANA and asked them to register this new VNC extension. I was worried they would say no, as they already had a registered format with no description, but they went for it and explained that it was legacy from a proprietary client. Lo and behold, we're developing an official TigerVNC patch for a first ever open source H.264 over VNC extension. The beta version is running smoothly! It might be very new, but some VNC clients and servers are already interested in implementing it.

The moral of the story is: Don't be afraid to communicate with other projects, even if they seem monumental. And one other thing: even one person can push forward development of the whole protocol; you just have to dream big.

Claußen: Did the appearance of the 10\$-USB-HDMI dongle help to push the PiKVM project?

Devaev: It depends. Before the USB dongle I used an HDMI-CSI-bridge (Figure 3) that had many advantages. Surely, the USB dongle (Figure 4) played a big role in popularising PiKVM as it was really cheap, but I always advised against using it due to hardware problems: it's not very stable.

Many people reported buying it and regretting afterwards, as they often saw a blank screen and couldn't do anything about it.

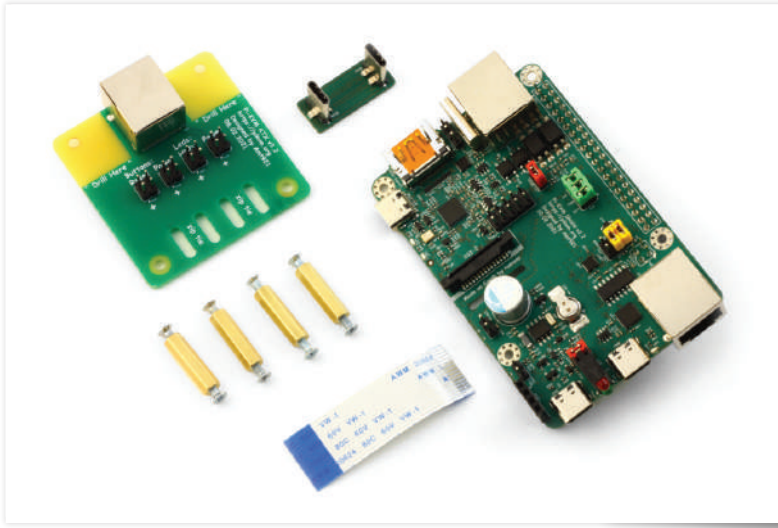


Figure 5: PiKVM v3 HAT (Source: Maxim Devaev) [5].

A USB dongle is a black box: you have no idea how it works, and if something goes wrong, you can't fix it. The bridge, on the other hand, is much more open and almost all video processing is done by the kernel and my software.

Claußen: The PiKVM is an open-source project that can (currently) be used free of charge. How is its development financed? You now seem to work full time on the project.

Devaev: I would lie if I said I had a long-term vision, a plan and understanding of the market. It all started last year, when I quit my job to take a break and to think about my career. I continued working on PiKVM as a hobby. The community was growing and I thought it would be great if PiKVM became my main job, and I've seen some open-source projects whose authors made a living at it. It was a win-win situation: I would be doing what was really interesting to me, and users would get high quality software for a small donation. As a test, I set up Patreon and announced I was accepting donations now, as I was now without a job and ready to improve PiKVM upon request (in case you needed a certain feature quickly). To my surprise, several months later, I began making some profit. It was way less than my previous salary, but considering I stopped spending money on trips to the office, started getting enough sleep, and even lost some weight, it was a fair trade. Users were also happy, as it could now be half an hour between finding a critical bug and a fix, and I was adding new features in a matter of days.

Claußen: How is the community supporting your project besides helping you to afford full-time development?

Devaev: I'm not very good at writing documentation, so many people help me with that. Some write recipes and scripts, come up with and describe new ways to use some of the PiKVM features. Some create bug reports and give me remote access to their hardware for me to debug problems. And the most important thing of all: when the number of users became really big, it became impossible to reply to every single person on Discord [3], so active users helped me onboard new users and now only reach out to me when they can't solve a problem themselves. I'm very thankful for those people, as I can now dedicate all my time to writing software.

Sometimes there are people who bring in patches with new features or bug fixes, but there are not many of those people yet, and I'm virtually the only PiKVM developer at the moment. I'm planning to hire some people to help me with software full-time when we start selling devices. So if you buy our hardware, you will sponsor the development of open source software.

Claußen: If someone likes to support your work, what would be the best way in doing that?

Devaev: Besides donations, help is appreciated with things we can't handle on our own yet. Some of those problems are outlined in our GitHub issues [4].

Claußen: Are there any plans to commercialize your work?

Devaev: Right now my main source of income are donations. I'm often told I could sell a commercial license to the PiKVM software with advanced features, but I don't want to do that for two reasons. Firstly, PiKVM has direct access to the hardware and I wouldn't trust such a thing if it wasn't open source. Secondly, a paywall would inevitably lead to someone forking your open software and developing pro features for free, getting part of your community to themselves.

The other way to monetize, which will soon become primary, is selling my own hardware (PiKVM v3 HAT) (Figure 5)), you can use instead of that DIY version to get more stability and additional hardware features. But I don't plan to abandon PiKVM DIY and force users to buy the v3. I don't pursue wealth at any cost, just enough money to feed me and my cat.

Claußen: Can you tell us a bit about your hardware, the PiKVM v3 HAT?

Devaev: To DIY a PiKVM, you have to use your hands and optionally a soldering iron. I decided to make it a bit easier for the user, as

well as to improve hardware stability of this version and add some features that are hard to DIY. Although I came up with the original concept of the device, I couldn't develop a completely custom board (this is a separate magic that I don't own), so I invited my friend with hardware skills to work with me on this project. Our PiKVM v3 HAT has a built-in server power controller, a video capture device, a USB switch (to emulate plugging a cable out, it's sometimes useful), and a serial port (to connect to a server or console in a rack), clock for precise logging — all on one board. Basically, it's a solution that's better and cheaper than all the \$500 ones you have on the market. You don't have to take my word on it: you can look up v3 reviews on YouTube. And soon everyone will be able to buy it.

Claußen: Also, you have an enclosure for the Raspberry Pi 4 and your add-on — that is made from metal. Was there a reason not to use plastic? We read on Discord that it is easier to build in Russia something out of metal than rubber or plastic.

Devaev: That's true. The metal enclosure came out more heavy, sturdy and much cheaper than a plastic one. Those who ordered v3 HAT on Kickstarter could get this case for free. Then it will be sold in online stores. And we also have a free drawing of the case for 3D printing, if you like to work with your hands.

Claußen: Will your kit be available for customers to be ordered from stores outside of Russia?


Devaev: The first batch should be available in the USA shortly. The second batch will be available worldwide in autumn 2021.

Claußen: What plans for the PiKVM do you have for the future?

Devaev: We will continue developing both software and hardware. We plan to implement 1080p 60Hz capture (you can't do that now due to Pi 4 restrictions) and create a secure solution for cloud access for users without an open external IP address.

Claußen: Do you have any other projects in development? Can you share some details about what you are working on?

Devaev: uStreamer, I guess. It was created for PiKVM, but it's a totally separate component many people use as a server for webcam video. (It's used as a capture device in PiKVM.) I wrote it from scratch as an alternative to MJPG-Streamer to fix the performance problem. Now uStreamer is the fastest specialised MJPEG/H.264 video service, and I'm really proud of that.

Claußen: Maxim Devaev, thank you for the interview. Those who are now heading to the Kickstarter campaign [5], we have some good news. The campaign was more than successful. This means you will be able to order the PiKVM v3 HAT in an online store soon. Also have a look at the Elektor Store [6] as our purchase team is currently working to be able to offer you also some PiKVM v3 HATs in the near future. 

210405-01

Questions or Comments?

Do you have technical questions or comments about this article? Email the author at mathias.claussen@elektor.com or contact Elektor at editor@elektor.com.



RELATED PRODUCTS

- Raspberry Pi HDMI to CSI-2 Adapter Board (supports up to 1080p25fps) (SKU 19707)
www.elektor.com/19707

WEB LINKS

- [1] M. Claußen, "Raspberry Pi as a KVM Remote Control", Elektor 1-2/2022: <http://www.elektormagazine.com/200523-01>
- [2] Pi-KVM Homepage: <https://pikvm.org/>
- [3] Pi-KVM on Discord: <https://discord.com/invite/bpmXfz5>
- [4] Pi-KVM GitHub issues: <https://github.com/pikvm/pikvm/issues>
- [5] Pi-KVM v3 HAT on Kickstarter: <https://www.kickstarter.com/projects/mdevaev/pikvm-v3-hat>
- [6] Elektor Store: <http://www.elektor.com>

Autonomous Vehicle with 2D Lidar

ESP32 Pico Interprets Data from the Lidar Module



By **Clemens Valens** (Elektor Labs)

Lidar enables robots and other self-driving vehicles to detect obstacles around them without touching, providing much more freedom of movement. To get a feel for this technique, I built a simple remote-controlled cart and let it drive around in my livingroom.



Lidar is an acronym for light detection and ranging. Lidar is like radar, except that it uses light instead of radio waves. The light source is a laser. A lidar sends out light pulses and measures the time it takes for a reflection bouncing off a remote object to return to the device. As the speed of light is a known constant, the distance to the object can be calculated from the travel time of the light pulse (**Figure 1**).

Two-Dimensional Lidar

Lidar can be one-dimensional (1D), like laser distance meters. It can also be two-dimensional (2D), much like the radars used by ships

and traffic control towers on airports. 3D lidar exists too and is used for instance by planes to create three-dimensional renderings of the surface of the earth below them. For this project, a 2D lidar is used.

Basically, a 2D lidar is nothing more than a rotating 1D lidar. Instead of rotating both the laser and detector, it is often easier to shine the light on a rotating mirror. By sending out light pulses periodically 360° can be covered and a distance map with the lidar at the center can be created.

Note that the reflectivity of objects is import-

ant. An ideal black body cannot be seen by a lidar as it does not reflect any light at all.

Connecting the Lidar

For my experiments, I used the X4 from Ydlidar (**Figure 2**). It features a range of up to 10 meters and an angular resolution of 0.5° (for distances up to 50 cm). It has an infrared laser with a wavelength of 785 nm. This device rotates the laser and detector assembly; it does not use a rotating mirror.

The X4 lidar outputs distance data as a continuous serial stream at 128,000 baud. It is also controlled over this serial link. You

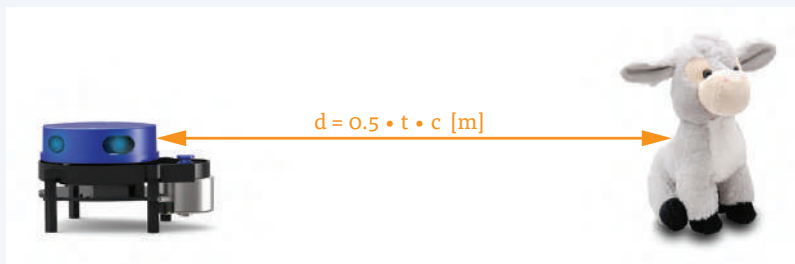


Figure 1: This is how you calculate distances. t is the travel time of the light pulse in seconds and c is the speed of light in m/s. The factor 0.5 corrects the fact that the pulse must travel to the object and back and therefore travels twice the distance.

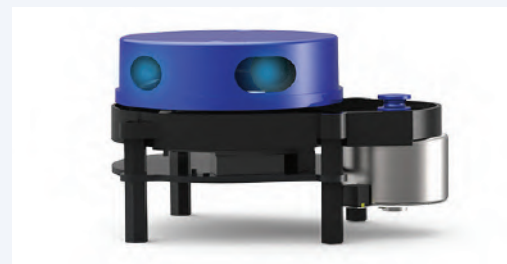


Figure 2: The YDLIDAR X4 is a low-cost 2D lidar with more than enough range and precision to make a robot move around without bumping into objects.

can start and stop scanning by sending it simple commands. You can also request some information about the device. The motor is controlled separately with two extra wires, one for on/off and one for speed. This means that this lidar can also work in one-dimensional mode when the motor is not turning.

I connected the serial port and the motor connections to an ESP32 Pico Kit (**Figure 3**). To save power, I only used the lidar motor on its lowest speed, which is about 400 rpm. The lidar then consumes approximately 400 mA.

Parsing Lidar Data

For programming the ESP32 Pico Kit, I used the Arduino IDE. After writing a function to parse the lidar data, I had to check my interpretation of the data. The X4 development manual is not very clear about how to do this and specifies two levels of detail. The second level provides better angle resolution, but involves many inverse tangent calculations, which are computationally intensive. Therefore, I first tried naïve interpretation.

I placed the lidar on a table inside a rectangular enclosed space and let it scan for a while. After stopping it, I made it spit out an averaged 360° scan on a serial port as comma-separated values (CSV) that I loaded into Excel. With the radar chart function, I could display it (**Figure 4**). The result looked a lot like my rectangle and the distances were correct too, and so I didn't bother to try to improve the quality by adding inverse tangent calculations.

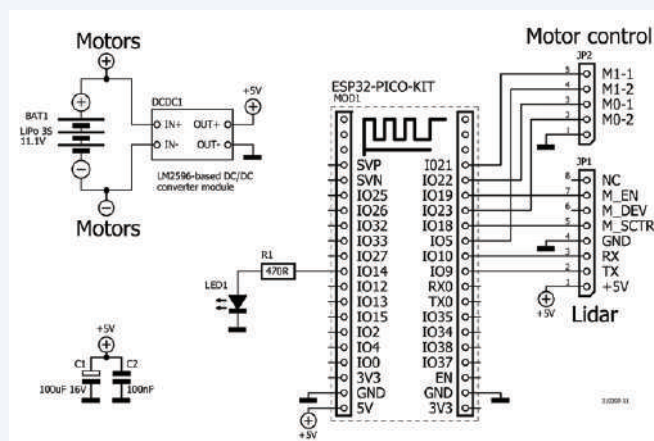


Figure 3: The ESP32 Pico Kit is used as the brains of the autonomous vehicle. The motor drivers and DC-DC converter are generic modules available online.

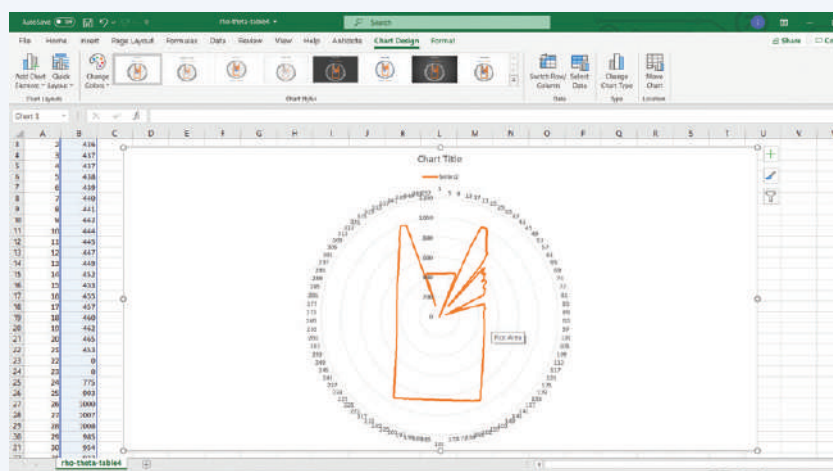


Figure 4: Microsoft's Excel let me quickly check my interpretation of the lidar data. The short horizontal line at 400 mm (and the shadow behind it) is due to my laptop recording the decoded lidar data.

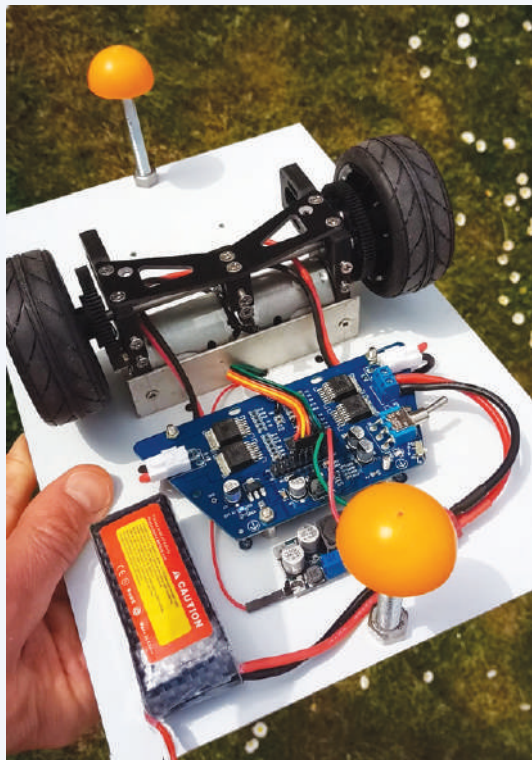


Figure 5: Bottom view of the vehicle. The motor assembly and motor driver board were once obtained from Landzo.com but are no longer available.

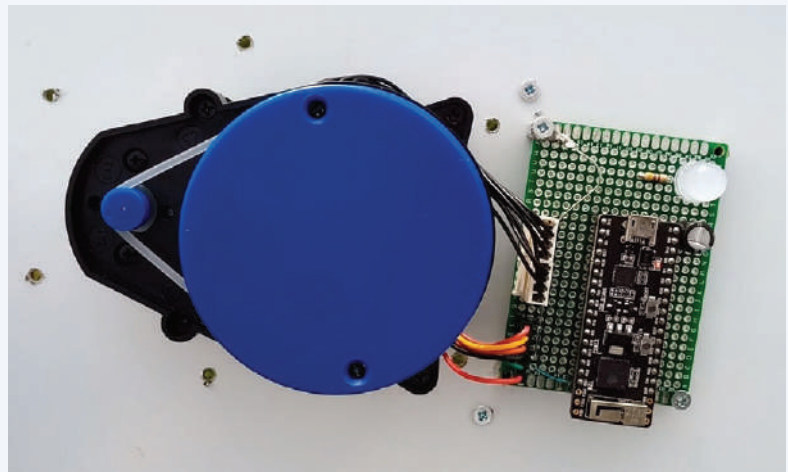


Figure 6: Top side of the vehicle showing the ESP32 Pico Kit and the Lidar.

Build a Little Robot

As a next step, I built a simple remote-controlled cart on which I mounted the lidar. The cart has two motorized wheels in the center and a stand-off on either end. I glued ping-pong balls cut in half to the standoffs to improve sliding behaviour. Everything is mounted on a sheet of double-sided copper FR4 PCB material. The wheels with motors, the motor driver board and the power supply, a 3S 11.1 V Li-Po battery, are mounted on the bottom side (Figure 5). The ESP32 module and the lidar are mounted on the top side (Figure 6). The plate is grounded and shields the ESP32 module from the noisy motors. The center of the lidar is in the center of the mounting plate. The wheel axle too is in the center. This simple car can spin around and is pretty agile and manoeuvrable.

Add Remote Control Over Bluetooth

As a remote control, I used the free and open-source Dabble library [1], which provides Bluetooth control for the ESP32 and Arduino together with a smartphone app featuring

multiple control surfaces (Figure 7). One of these is a gamepad, which was perfect for my application. It is really easy to use and it let me control the cart with my phone.

A Pathfinding Algorithm

My goal was to program the cart so that it would drive around all by itself without bumping into objects like furniture and things. A popular approach is to let the cart run around and back up or steer away when it comes too close to an object, but this requires the cart to make decisions. I wanted something simpler. There are many examples of simple algorithms that result in complex behaviour, for instance the way a flock of birds stays together [2], and I wanted something like that.

My idea was to make the cart always move in the direction of the greatest distance as reported by the lidar. To avoid making it run in circles, it only looks forward, in the range of -90° to $+90^\circ$. Implementing this rule was quite easy. For every scan, a table is updated with the average distance for each degree. Therefore, the table has 360 entries, one per

degree. This table is then searched for the 10-degrees-segment that has the highest averaged distance (the width or aperture of 10 degrees is a rather arbitrary value). The center degree of this segment is the direction the cart should take. To achieve this, the cart will turn so that the center degree, the direction, moves to zero degrees. We now have a classic control algorithm that tries to minimize an "error" (Figure 8).

First Test Run

To my great surprise, on its first run with this simple algorithm the cart managed to move through our living room without bumping into obstacles (Figure 9). It circled our couch and passed through narrow passages without difficulties. The cart does not have any knowledge of its environment or itself, like its dimensions. Also, I hadn't tried to optimize anything. All parameters like forward- and turning speed and search angle were just set to values that I thought might be reasonable.

The Bluetooth remote control turned out to be very handy for confusing the cart or to help

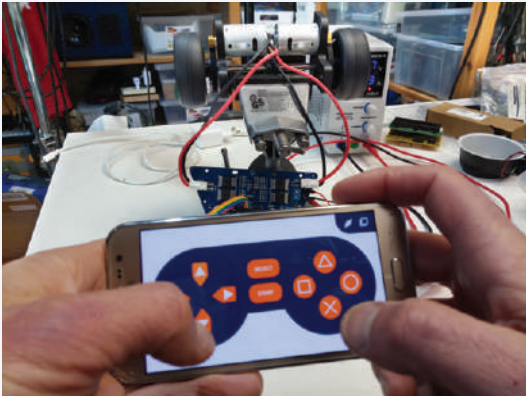


Figure 7: Trying out the Dabble-based smartphone remote control on the test bench.

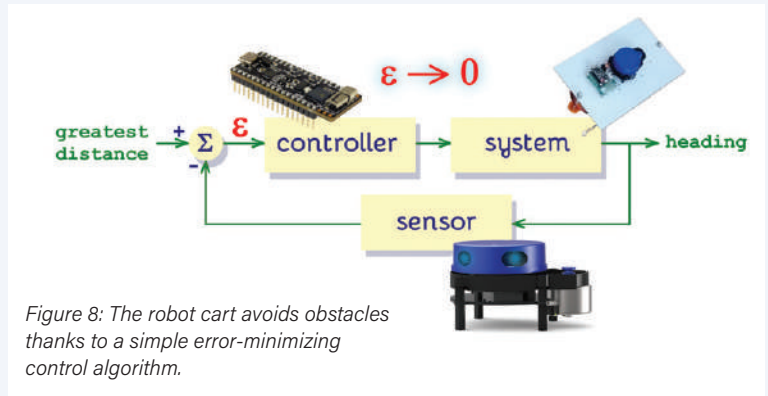


Figure 8: The robot cart avoids obstacles thanks to a simple error-minimizing control algorithm.

Figure 9: Go cart, go! Watch the video [4] to see how smooth it moves.



it out difficult situations. Also, it can be used to adjust parameters on the fly.

Now, as I am not into optimizing at all, I am more a proof-of-concept kind of person, I stopped at this point. If you want to have a play yourself, you can find the links to the code below. There are many possibilities for improving this design, and it is still a long way from an autonomous vacuum cleaner or lawnmower, but the obtained results are very encouraging. The software for this project, an Arduino sketch for the ESP32, can be downloaded from [3]. A video is available at [4] with a bonus at [5].

210399-01

Contributors

Idea, Design, Text and Photographs:

Clemens Valens

Editor: **Jens Nickel, C. J. Abate**

Layout: **Giel Dols**

Questions or Comments?

Do you have technical questions or comments about this article? Email the author at clemens.valens@elektor.com or contact Elektor at editor@elektor.com.



RELATED PRODUCTS

> **ESP32 Pico Kit (SKU 18423)**
www.elektor.com/18423

> **YDLIDAR X4 (SKU 18601)**
www.elektor.com/18601

WEB LINKS

[1] Dabble: <https://thetempedia.com/product/dabble/>

[2] Flocking behavior: [https://en.wikipedia.org/wiki/Flocking_\(behavior\)](https://en.wikipedia.org/wiki/Flocking_(behavior))

[3] ESP32 Arduino sketch for this project: <https://github.com/ClemensAtElektor/Lidar-controlled-autonomous-vehicle/>

[4] Watch this project on video: https://youtu.be/BmNelv_gR9Q

[5] Lidar-based parking light by Rob Reynolds from SparkFun: <https://youtu.be/KRfidalgJx8>

The Raspberry Pi Zero 2 W Goes Quad-Core

By **Mathias Claußen** (Elektor Lab)

The Raspberry Pi Zero with its ARM11 core is now getting a bit long in the tooth, and its performance for many applications could be described as marginal. Fortunately, the Raspberry Pi Foundation fairly recently announced a much anticipated upgrade, the Raspberry Pi Zero 2 W. What does this latest Raspberry Pi bring to the table?

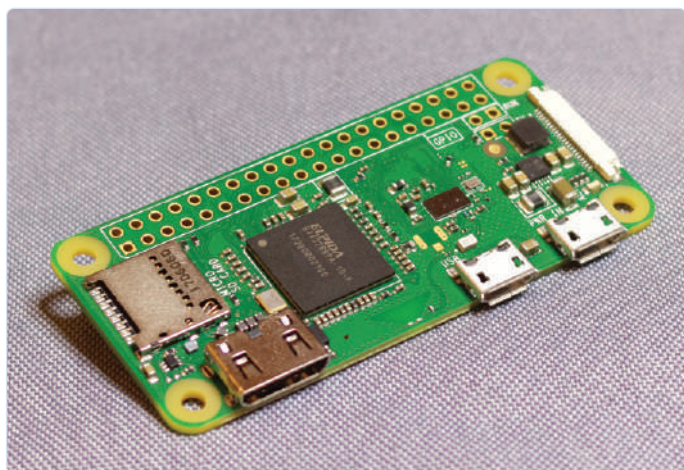


Figure 1: The Raspberry Pi Zero W.

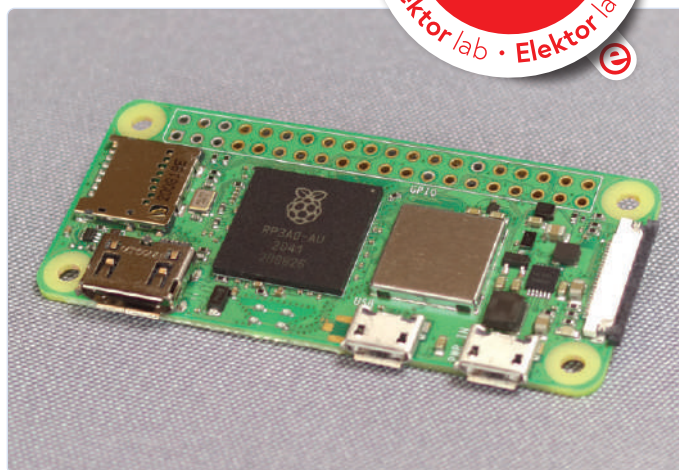


Figure 2: The Raspberry Pi Zero 2 W.



The Raspberry Pi Zero W (**Figure 1**) is a popular small development platform costing only around €10. With the necessary peripherals, it works as a standalone computer, but it's small and flexible enough for embedded applications such as a media player, camera, or as a simple controller device. When introduced in 2017, it was a small

Linux-capable computer, powered by a single-core SoC, which we first saw fitted to the original version of the Raspberry Pi 1 (introduced five years earlier). The speed of ongoing processor development is dizzying. From today's perspective, the 10-year-old ARM11 single-core processor is looking decidedly elderly, although it still does the job in many applications. Users have been crying out for a replacement offering better performance and lower power consumption.

In the fall of 2021, the Elektor Lab engineers were lucky enough to get our hands on a pre-release Raspberry Pi Zero 2 W (**Figure 2**). It was a nice opportunity to see what Eben Upton and the Raspberry Pi team had been working on.

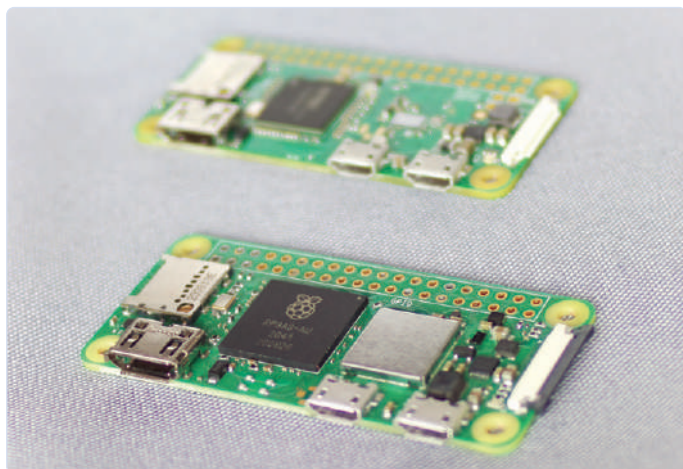


Figure 3: Raspberry Pi Zero 2 W and in the background the Raspberry Pi Zero W.

A Drop-In Replacement

Nothing has changed about the board's physical and its connectors. **Figure 3** shows a Raspberry Pi Zero W and a Raspberry Pi Zero 2 W, side by side. All accessories and enclosures for the original Pi Zero can still be used with this new board without any issue. As with the Raspberry Pi Zero W, we have two micro-USB ports, the usual 40-pin connector, and a mini-HDMI output. One of the two USB ports is again a USB OTG to which USB accessories can be connected via an adapter — or with which the Raspberry Pi Zero 2 W itself can become a USB device. As usual, a micro SD card is used for booting. A higher read speed smoothens the Raspberry Pi Zero 2 W's operation. **Figure 4** shows the underside of the board which again has no mounted components. **Figure 5** shows the topside of the Raspberry Pi Zero 2 W, where we can see the wireless chips now have shielding, and the SiP labeled RP3A0-AU is the obvious upgrade. The outline of this new SiP is only slightly larger than the one fitted to previous Raspberry Pi Zero models.

The RP3A0-AU SiP

The RP3A0 label on the SiP indicates 'RP' for Raspberry Pi, and '3' for the SiP generation (i.e., it has a Raspberry Pi 3 at its core). The exact

Table 1: Specifications

- > Broadcom BCM2710A1, 4-core Cortex-A53 (ARMv8)
64-bit SiP @ 1.0GHz
- > 512MB LPDDR2 DRAM
- > VideoCore IV GPU
- > IEEE 802.11.b/g/n Wireless LAN, Bluetooth 4.2/BLE
- > 40-pin GPIO-Header
- > Mini-HDMI-Port
- > Micro-USB 2.0 OTG Port
- > CSI-Camera-Port
- > Micro-SD-Slot
- > Micro-USB Power in (5 V DC/2.5 A)
- > Composite Video and Reset-Pin test points

specifications can be found in **Table 1**. Essentially, the quad-core engine from the (five-year-old) Raspberry Pi 3 A/B+ has been shoehorned into the Raspberry Pi Zero form factor and runs with a clock speed of 1 GHz. Anyone hoping for a Raspberry Pi 4-based Zero might feel a tinge of disappointment but no doubt this will appear sometime in a future upgrade.

The Raspberry Pi 3A+ uses a BCM2837B0 clocked at 1.4 GHz. In the case of the RP3A0, this uses a BCM2710A1 that in the Zero 2 W is clocked at 1.0 GHz. The lower clock rate reduces energy consumption and heat generation. Further optimizations of the energy consumption for the Raspberry-Pi-3-based systems are in the pipeline, as Eben Upton revealed during his interview [1]. The use of a SiP from the Raspberry Pi 3 series means that there is already a stable and broad software base available for use on the Zero 2 W. SD card images for the Raspberry Pi 3 will boot on the Zero 2 W without any further tweaking.

Some of the important questions I put to Eben were: How has the power consumption changed? Can I still use my previous power supplies? What new features are available and what about its performance?

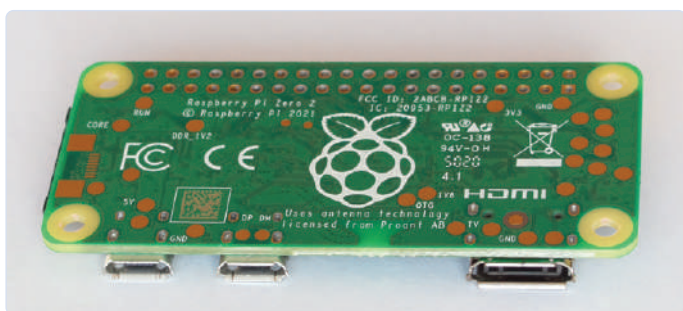


Figure 4: A view of the underside of the Raspberry Pi Zero 2 W.

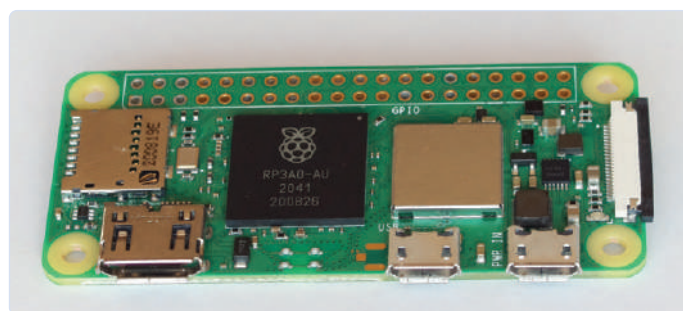


Figure 5: The Raspberry Pi Zero 2 W component side.

Smaller and Less Power Hungry

Low power consumption is a key feature of the Raspberry Pi Zero series. This is achieved using a minimalist approach to the board's design and choosing well-established and proven chipsets. Energy consumption of the Raspberry Pi Zero 2 W is around 2.5 W maximum with active HDMI output and Wi-Fi link. Most computer USB ports can supply up to 0.5 A at 5 V, which allows the board to be powered from a PC's USB port without the need for an external mains adapter. With the desktop idling, it draws around 0.7 W and other operational modes will reduce the power consumption even more. For comparison, a Raspberry Pi 3B+ draws around 2 W in idle mode, which is significantly more. Under load, the Raspberry Pi 3B+ clocked at 1.4 GHz reaches around 4.4 W.

Its compact outline enables the Raspberry Pi Zero to be used in a wide variety of applications, from web radio to mobile game consoles or simple control systems. The new Raspberry Pi Zero 2 W is also ideal as a server for home assistants; its lower energy consumption means that it has less of an impact on the environment than the larger models.

2.4 GHz Wi-Fi and Bluetooth

The Raspberry Pi Zero 2 W supports 2.4 GHz Wi-Fi in accordance with 802.11 b/g/n. Bluetooth and BLE 4.2 are also provided. Wi-Fi in the 5 GHz band is still reserved for its larger siblings such as the Raspberry Pi 4 B or Raspberry Pi 3 B+.

First Boot

When it comes to booting the original Raspberry Pi Zero W, it's usually a good opportunity to go and grab a coffee, by the time you get back everything should be stable and ready to go. With the Raspberry Pi Zero 2 W however, the same process takes around 30 seconds when loading a current version of Raspberry Pi OS (32 Bit) without any optimization to the boot loader. The system is then ready for use just like a Raspberry Pi 3. If you already use an original Raspberry Pi Zero, you will for sure have an OTG adapter (**Figure 6**) to connect a mouse and keyboard. The four cores of the Zero 2 W provide a noticeable increase in throughput, even

though the clock rate is the same as earlier versions of the Zero. In the Raspberry Pi 3B+, the same processor runs 400 MHz faster.

The 512-MB RAM also offers ample space for some applications, but you can't expect the Raspberry Pi Zero 2 W to be a replacement for a fully-fledged desktop. It gets a bit clunky loading web pages with only 512 MB RAM to play with. For this a Raspberry Pi 4B or Raspberry Pi 400 would be a better choice and offer significantly more computing power; the strengths of the Raspberry Pi Zero 2 W lie in other areas.

A Modest But Welcome Upgrade

This latest Raspberry Pi Zero 2 W is a welcome addition to the Zero family. The SiP from the Raspberry Pi 3 is admittedly already five years old, but it is a proven platform and represents a significant upgrade to the Raspberry Pi Zero's processing power. It enjoys good software support and its four-core Raspberry Pi 3 SiP will allow many existing software applications to run on the Zero's more compact board outline and benefit from its lower power requirements.

The Zero's form factor also opens up new possibilities in areas like automation, retro-gaming, or media players. You will find it to be a more economical option for projects where a Raspberry Pi 4 is too expensive or too power hungry. ◀

210536-01

Contributors

Project and Text: **Mathias Claußen**

Editor: **Jens Nickel**

Translation: **Martin Cooke**

Layout: **Giel Dols**

Questions or Comments?

Do you have any questions or comments about this article? Email the author at mathias.claussen@elektor.com or contact Elektor at editor@elektor.com.



RELATED PRODUCTS

- ▶ **IQaudIO Codec Zero – Sound Card for the Raspberry Pi Zero (SKU 19541)**
www.elektor.com/19541
- ▶ **Retroflag GPi – Game Boy case for the Raspberry Pi Zero (W) (SKU 19273)**
www.elektor.com/19273
- ▶ **ZeroDock – Prototyping Dock for the Raspberry Pi Zero (SKU 19760)**
www.elektor.com/19760



Figure 6: A USB-OTG adapter.



Project Idea: Wi-Fi camera with motion detection

One of the most popular add-ons to the Raspberry Pi Zero is a camera module to make a Wi-Fi enabled camera. The boot time of the setup is however quite long and the CPU quickly reaches its limits when it comes to motion detection. Live streaming can suffer from dropped frames and a certain degree of image latency. These effects are not evident when the same camera and software configuration are run on the full size Raspberry Pi 2 and later, more powerful, versions.

The extra processing power of the Raspberry Pi Zero 2 W should resolve the dropped frames and latency issues and make it a good compact platform for use as a webcam with motion detection and live streaming.



Project Idea: USB printer and scanner server

An old printer can be reborn as a network (in some cases wirelessly) attached device. Using the Raspberry Pi's USB port it's possible to connect the printer to a network. Depending on the manufacturer and model, this may require a little configuration work, but the printer can continue to be operated in its own network with fewer security gaps. This even applies to scanners and printers that can no longer be used with current Windows or MacOS versions [2].

CUPS and SANE are the classic representatives for printing and scanning under Linux. A printer server can be set up in connection with Samba. SANESWinDS enables Windows clients to access a scanner via the network, so that several computers can access the scanner.

The Raspberry Pi Zero 2 W can also be used as a scanner - using the GPIOs and a few Scripts you will be able to handle complex processing of documents.



Project Idea: Multimedia Player

The Raspberry Pi Zero 2 W is ideally suited for use as a small media player. Together with the IQaudio Codec Zero HAT, it becomes a small streaming device for audio. Distributions like Volumio are so easy to use, everything is optimally supported, since the RPi zero 2 W is essentially a Raspberry Pi 3. The MPD (Music Playing Daemon) can also be installed on the RPi Zero 2 W and operated on various devices with one of the various front ends. Using it as a Wi-Fi stick with Kodi is also not a problem.



Project Idea: Debugger for the RP2040 and other MCUs

The option to use a Raspberry Pi as a network-compatible debugger is somewhat overlooked in the documentation from the Raspberry Pi Foundation. While working on the articles for the release of the Raspberry Pi Pico, we carried out our first experiments with a Raspberry Pi Zero as a Wi-Fi-enabled debugger. The setup does require some manual tweaking and adjustment of the configuration, but the result is an inexpensive Wi-Fi-enabled debugger, not just for the Raspberry Pi Pico, but also a wide range of Cortex-M MCUs.

WEB LINKS

- [1] M. Claussen, "A Decade of Raspberry Pi: An Interview with Eben Upton," Elektor Industry 03/2021: www.elektormagazine.com/210464-01
- [2] Web UI frontend for scanner: <http://github.com/sbs20/scanservjs>
- [3] Elektor TV: The Raspberry Pi Zero 2 W

Notes From the 2021 World Ethical Electronics Forum

By Priscilla Haring-Kuipers (The Netherlands)

The first World Ethical Electronics Forum — which took place online and live at productronica on November 18, 2021 — inspired global innovators in electronics with an open discussion about ethics and Sustainable Development Goals (SDGs).

**WORLD
ETHICAL
ELECTRONICS
FORUM**

On the November 18, 2021, almost 200 people came together virtually and physically to be involved in the first World Ethical Electronics Forum (WEEF), which was organized by Elektor and Elektronik Praxis during productronica 2021 in München. Among the speakers and participants of this industry-driven get-together there was a sense that the time is right to see what we can do to make electronics more ethical. In the past decade, more people have been thinking about ethical issues in electronics. Many have been discussing the different facets of our industry and applying ethical practices in their own businesses. Now the time seems ripe to share these thoughts and practices with the industry as a whole and to learn from each other.

Watch WEEF 2021!



WEEF 2021 Speaker Snapshots

Let's take quick look at what some of the speakers covered at the WEEF 2021 event. You can watch all of the WEEF talks online at Elektor TV [2].

- Professor Dr. Stefan Heinemann opened WEEF with his keynote, "A Focus on SDGs in Electronics." In this excellent talk, he encouraged electronic businesses to "do more" than the current legislation demands. He stated that good morals are good business and not to believe the naysayers.
- Dr. Paula Palade spoke on how we might establish policy around the design and use of self-driving cars. She is working on a European project to draft the first guidelines.
- Amir Sherman (Edge Impulse) pointed to the opportunity of using machine learning in smaller hardware to combine real life metrics with advanced modelling.
- Margot Cooijmans of the Philips Foundation discussed how to provide health tech to "under serviced regions." This cannot be done by dumping donated tech as this has no lasting health effects.
- Dirk Akemann of SEGGER spoke of a "friendly licensing" approach where their products are free to use for private, educational and prototyping purposes.
- Felix Plitzko from AISLER is striving to "make hardware less hard" and promised to make carbon neutral PCB production available in the future. He is currently investigating this together with the Fraunhofer institute.[3]
- The Brenner family (Johannes Brenner and Markus Brenner) joined to talk about how integrating mindfulness at the core of your business, rather than chasing profit, sets you up to take good care of the people in your organisation.





More WEEF

Of course, we should not be satisfied with talking alone. If we are to be ethical in electronics, we must also take action. This was the first edition of WEEF, and this is to be a yearly event. The second WEEF is already planned for November 2022. Between now and then, we will be working on a WEEF Index featuring most the ethical people/companies in electronics, as well as a WEEF manifesto. We would love your input on the following questions and more:

- Who do you want to be in the WEEF Index?
- What should be in the WEEF manifesto?
- What are the important ethical questions in your practice?

Please follow the conversation and participate at www.elektormagazine.com/weef. 

210645-01

WEB LINKS

- [1] WEEF Webpage: <https://www.elektormagazine.com/weef>
- [2] ElektorTV, "World Ethical Electronics Forum 2021": <https://youtu.be/ELpxR6SuLb8>
- [3] AISLER, "AISLER Plans to Ship the First World-Wide Carbon Neutral Circuit Board in 2022," November 18, 2021: <https://www.elektormagazine.com/news/aisler-plans-to-ship-first-world-wide-carbon-neutral-circuit-board-in-2022>

Motor Control

How the Complexity of Motor Control Is Simplified



By Stuart Cording (Elektor)

The electric motor continuously reinvents itself to meet new application requirements from electronic toys and home appliances to power tools and electric vehicles. However, equally important are the diverse range of electronic control solutions that go hand-in-hand with the motor to meet the needs of such applications. Integration has been key, ensuring that the full performance of the motor can be attained. In the 1970s, the integrated H-bridge revolutionized home electronics, such as video cassette recorders. Today, highly optimized microcontrollers with dedicated motor-control capabilities ensure efficient operation using brushless motors.

Small permanent magnet brushed DC motors are found almost everywhere. Often termed fractional horsepower (FHP) motors, the majority sold provide motion control in automotive and home appliances. Thanks to the increase in comfort features, cars have augmented their electric motor-powered features beyond windscreen wipers and fans to include electric mirrors, window openers and sunroofs, doors, and adjustable seats. The DC motors used are simple to control, from an electrical point of

view, even if they are not the most efficient option on the market. Such motors remain in regular use except for applications requiring continuous operation, such as pumps and fans, that have moved to more efficient brushless motor technology.

Controlling Basic Brushed DC Motors

Simple permanent magnet DC motors construct their stator from a can within which typically two magnets are found.

Mounted centrally is an armature on a rotor shaft featuring the windings. These connect to a commutator on the rotor shaft. Carbon brushes, typically part of the cap fitted to the stator's can, enable electrical energy to be applied to the windings (**Figure 1**). The position of the commutator on the shaft is critical for ensuring optimal efficiency. Ideally, the commutator-brush position, known as the commutating plane, must enable the magnetic field generated by the rotor windings to be at right angles to the stator's magnetic field. However, during operation, this plane moves due to distortions in the stator's magnetic field. For simplicity, a commutating plane is selected that provides the highest efficiency at a specific rotation speed.

With the motor's commutation solved mechanically, only speed and direction of rotation need to be implemented electrically. If only speed adjustment is required, a pulse-width modulated (PWM) signal together with a suitably dimensioned power device, such as a MOSFET, is all that is required (**Figure 2**). Most microcontrollers (MCU) also feature a timer block that, once configured, can generate a PWM signal without regular interaction from the processor. An N-channel MOSFET as a low-side switch is preferred in such cases, as the gate can typically be driven directly in low-power applications. Should high-side switching be needed, an N-channel MOSFET must be used together with a MOSFET driver to generate the required gate voltage [1].

Monitoring Speed and Controlling Direction

For speed control, the control loop used requires the current rotation speed of

the motor shaft. One approach is to use a sensor, such as an optical detector, together with a slotted wheel attached to the rotor. In a digital control system, the period between signals from the optical detector can be converted into motor rotations per minute (RPM). Most MCUs also have an input block that can capture timer values on a signal's rising and falling edge. Their resolution depends on the clock speed of the timer and should be high enough to provide usable data at the highest rotation speeds. The output of this block feeds into a proportional-integral-derivative (PID) control algorithm [2] that modifies the mark-space ratio of the PWM to maintain the setpoint for the motor's speed (**Figure 3**).

Such rotor-mounted sensors often prove to be too expensive for the application being built. Instead, the back-EMF of the motor, which is proportional to its speed, can be used and offers more than enough accuracy for most applications. During the phases of the PWM control where the power switch is off, the back-EMF can be measured directly at the motor's terminals. Since most microcontrollers also feature an analog-to-digital converter (ADC), this is typically very economical. Hardware implementation of this approach only requires a couple of resistors to limit the voltage to the input range of the ADC. More complex is the software implementation. The ADC measurement has to be synchronized with the shutting-off of the MOSFET and requires an additional short delay to allow the voltage to settle (**Figure 4**). For MCUs designed with motor control in mind, there will probably be a hardware link that can be established between the PWM output and ADC trigger, as well as a delay block, allowing this measurement approach to be automated.

With the elements for motor speed control in place, all that remains is to handle the direction of rotation. The H-bridge (**Figure 5**) uses four power switches that allow the polarity of the supply applied to the motor terminals to be swapped (either Q2 and Q3, or Q1 and Q4). Shorting the motor terminals via the upper or lower pair of switches can be used as an electrical



Figure 1: The component parts of a fractional horsepower DC motor. The commutator is clear to see on the rotor shaft by the armature coils. (Source: Shutterstock/Pixel Enforcer)

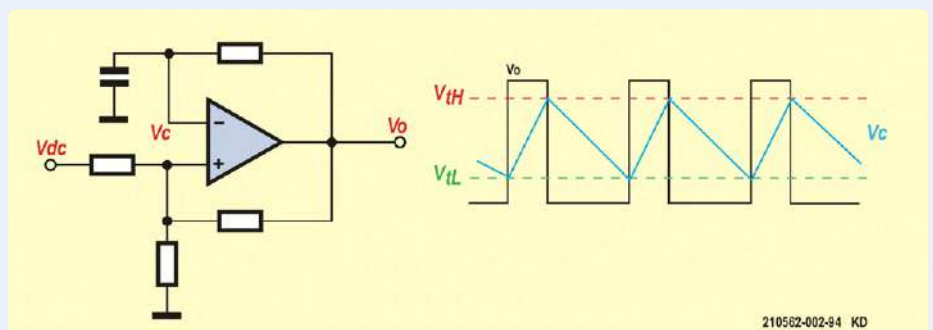


Figure 2: PWM generator based upon an operational amplifier.

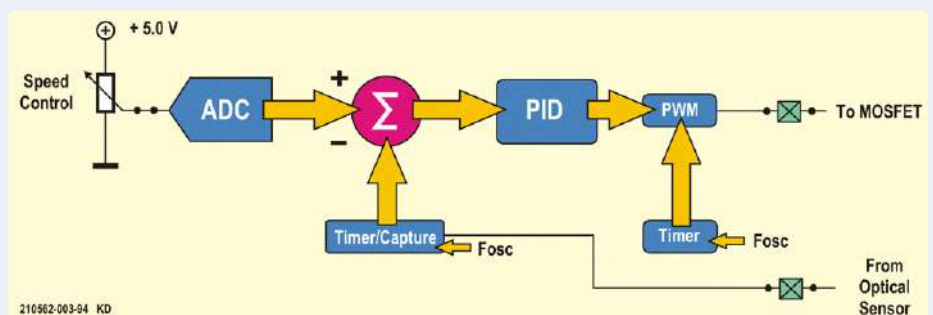


Figure 3: Output from a sensor attached to the motor's rotor can be used to capture timer values. These are combined with the speed setpoint and passed to the PID to regulate motor speed. (Source: Microchip Technology)

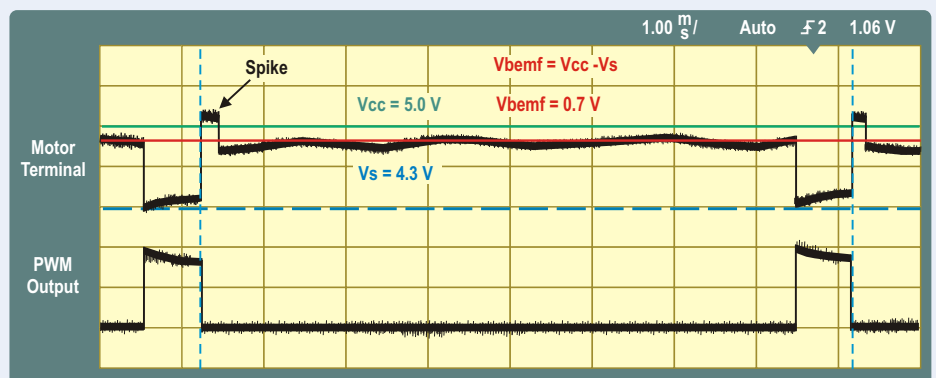


Figure 4: After the MOSFET switches off, a small spike appears at the motor's terminals. ADC measurements must wait until the back-EMF has achieved a steady-state. (Source: Precision Microdrives)

brake to slow the rotor speed. Speed control is again implemented using a PWM signal to one or both MOSFETs. However, it should be noted that there is potential for a short circuit if both high- and low-side MOSFETs are engaged simultaneously.

Most major silicon vendors offer highly integrated H-bridge solutions, simplifying its use and offering a host of protection features and diagnostics. The IFX9201SG [3] from Infineon is one such device, targeting currents of up to 6 A for motor control in industrial applications. The device can be controlled using general-purpose I/Os and a PWM signal, allowing motor speed and direction control. In parallel, an SPI interface provides access to the diagnostic regis-

ters. These provide feedback on shorts to the supply or ground, open load conditions, or undervoltage. If desired, speed and directional control can be switched to control via SPI rather than the direction and PWM input pins. The risk of an unintentional short circuit of the H-bridge is thus eliminated, and further protection features, such as over-temperature and overcurrent, are included. Those looking to quickly prototype a design can acquire the H-Bridge Kit 2Go [4] that integrates with Infineon's DAVE software development environment.

Optimizing the PID Controller

The selection of an MCU that meets the desired price point for the application can often result in an 8-bit device. Unfortuna-

tely, such MCUs are not well known for their math performance. This makes it tricky to implement a PID controller that responds promptly to changes in the speed setpoint or load variations. To tackle this, some PIC MCUs feature an integrated PID accelerator block (Figure 6) with a 35-bit accumulator, such as Microchip's PIC16F1619. Using the standard s-domain PID controller equation, three constants — K_1 , K_2 , and K_3 — are written to the module's registers before writing the desired setpoint and engaging the PID calculation module. The PID module reduces the number of clock cycles required to 9 from around 1,000 for a software PID on the same MCU [5].

The Advantages of BLDC and PMSM

While the simplicity of FHP DC motors is attractive, increasingly other design factors, such as efficiency, acoustic noise, and increased power density are pushing applications towards brushless DC (BLDC) motors and permanent magnet synchronous motors (PMSM). When driven optimally, BLDC and PMSM are much more efficient and, with their electrical windings in the stator, heat dissipation is simpler.

BLDC and PMSM motors can be thought of as inside-out FHP DC motors. The coils used for commutation are wound onto

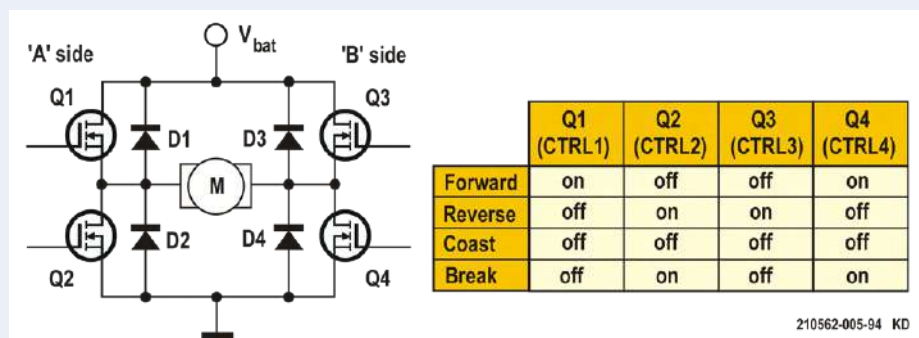


Figure 5: H-bridge circuit with table describing how switch operation controls direction. Shorting the motor terminals is also possible to implement an electrical brake. (Source: Microchip Technology)

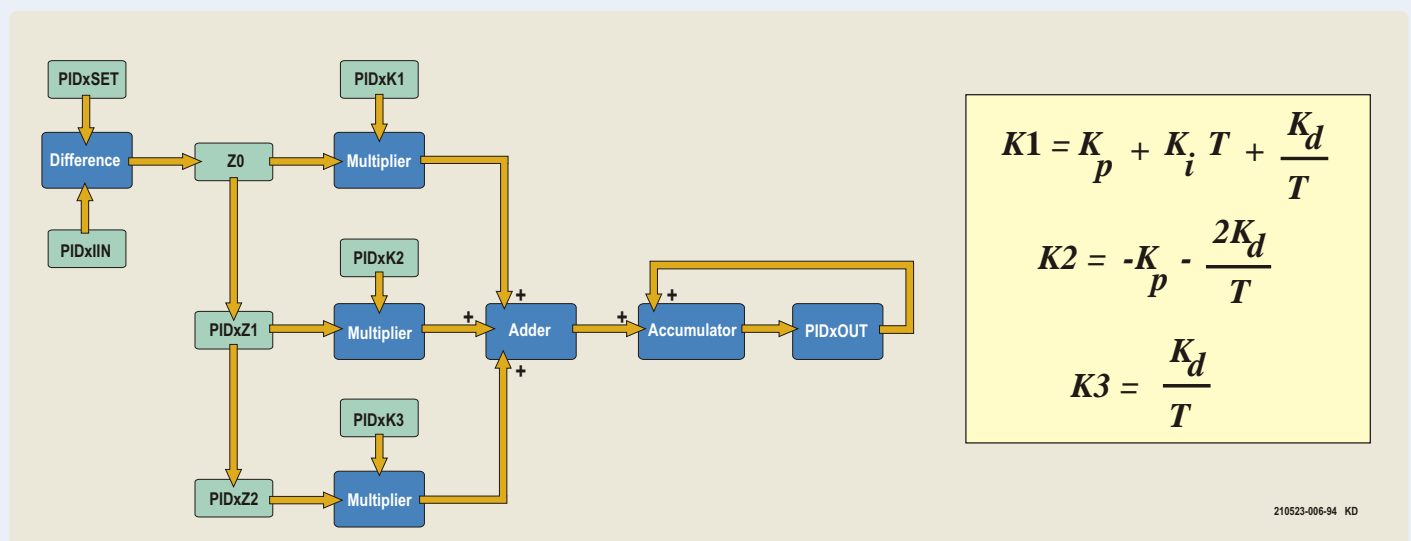


Figure 6: The PID block from the PIC16F1619 together with the equations to calculate the three standard constants used. (Source: Microchip Technology)

the stator, the motor's housing, while the magnet is wrapped around the rotor. This makes for light rotors with little inertia. However, there are no mechanics for implementing the commutation, which leaves this task to the development engineer. The key difference between the two motor types is that BLDC motors generate trapezoidal back-EMF and have simpler commutation demands than PMSM motors, which use a sinusoidal commutation and generate a sinusoidal back-EMF. PMSM motors are also quieter in operation.

The windings are typically implemented in a star formation. Six half-bridges are then required to energize the coils of a BLDC motor in a six-step pattern that applies power to two of the three (Figure 7). A magnetic field results that rotates around the stator, a field that the rotor simply has to follow and results in the desired mechanical rotation. Reversing the order of the commutation pattern causes the rotor to change direction. While this simple model of BLDC functionality makes operation clear, in reality motors are typically wound in a manner that requires several electrical rotations before a full mechanical rotation is achieved.

Sensorless Determination of Rotor Angle

The next challenge is determining the position of the rotor to ensure the next commutation point is output at the correct moment in time. Many BLDC motors integrate Hall sensors that indicate the rotor angle to a reasonable level of accuracy. The rising and falling edges of such sensors can be linked to many MCUs with edge-triggered interrupts, allowing the correct drive of the motor coils to be implemented simply in interrupt routines. Such signals can also be captured into a timer/counter to determine the rotational speed and feed the input of a PID controller. Motor speed is also controlled using PWM outputs. So, while the commutation of such motors falls upon the shoulders of the responsible development engineer, the approach is not that different from using their brushed DC counterparts.

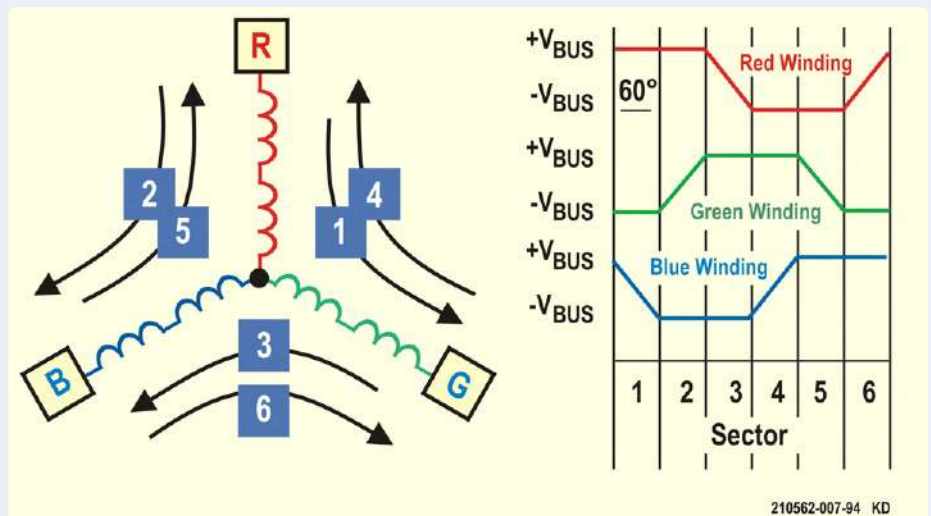


Figure 7: BLDC motor with star formation windings and the commutation steps for a single electrical rotation. Mechanical rotation may require several electrical rotations. (Source: Microchip Technology)

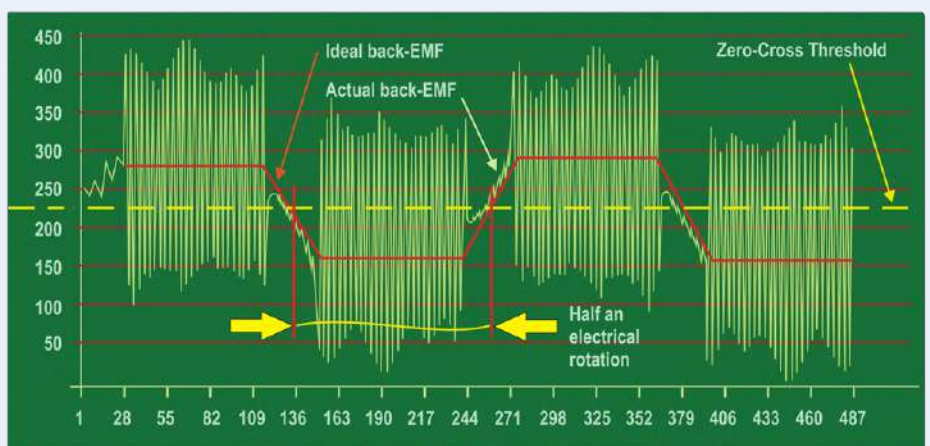


Figure 8: The back-EMF from the open phase can be measured and used to determine angular rotation, avoiding the need for a Hall or other sensor. (Source: Microchip Technology)

As already stated, sensors add to the cost of the bill-of-materials and there may simply be no place to accommodate them. Like brushed DC motors, BLDC motors also generate back-EMF. However, in these devices, this signal can be used to determine rotor position. As the back-EMF of the undriven phase crosses the zero-cross threshold (half of the current supply voltage), the rotor angle can be determined by measuring the time between these zero-crossing events (Figure 8).

While the theory seems simple, the practical implementation is exceptionally challenging. Firstly, motors do not generate back-EMF at a standstill, requiring an approach for spinning them up to a speed where an algorithm has enough back-EMF signal to measure. Next, the code required to evaluate the back-EMF data and apply filtering can have a significant execution time. This can mean switching between different algorithms and commutation approaches as the motor changes between lower and

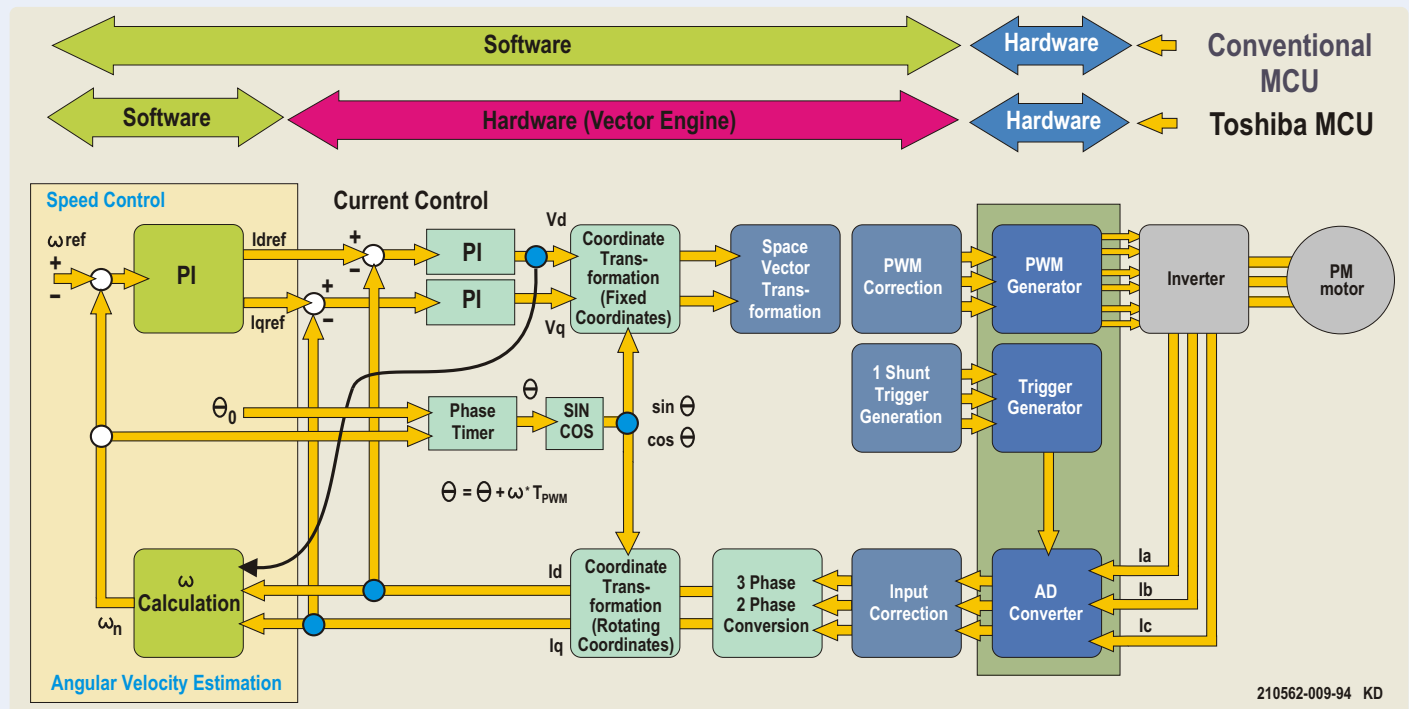


Figure 9: Highly integrated motor control modules, such as Toshiba's Advanced Vector Engine Plus, enable quasi-autonomous permanent magnet motor control. (Source: Toshiba Electronics Europe)

higher speeds. Suppliers of intelligent BLDC motor control solutions have solved many of these issues, enabling a new motor to be attached and 'learned' by attaching it to a motor controller with little input required from the design engineer.

Implementing Sinusoidal Motor Control

PMSM motor control algorithms have an additional layer of complexity. These demand that the PWMs of the MCU are modulated continuously to generate a sinusoidal output that energizes the motor windings. Such control approaches use field-oriented control (FOC), using the three-phase windings' currents to determine the current rotor angle. Using Clarke/Park transformations, this control method aims to maintain a 90° angle between the rotor and stator magnetic fields at all times, thus achieving maximum torque at all times.

While such complexity can be implemented in software on some MCUs, many

With so much complexity, leaders in motor control are focusing on simplifying their solutions so that developers can concentrate on creating differentiating features in their applications.

manufacturers have developed dedicated hardware modules that automate much of the complexity. One family of devices are the TXZ4A+ MCUs from Toshiba [6]. These Arm® Cortex®-M4 devices feature an Advanced Vector Engine Plus (A-VE+)

module that is tightly coupled with the PWM and ADC modules (Figure 9). Inside the A-VE+ module, the complete three-phase to two-phase and reverse Clarke/Park transformations are implemented in hardware, requiring only a PI controller and angular velocity estimation in software and some housekeeping. The rest of the time, the modules operate together quasi-autonomously.

Driving Innovation in Motor Control

With so much complexity, leaders in motor control are focusing on simplifying their solutions so that developers can concentrate on creating differentiating features in their applications. According to Jonas P. Proeger, Director Business Management at Trinamic, now part of Analog Devices (Figure 10), the core themes they see revolve around improving efficiency, reducing noise, and increasing intelligence. Efficiency improvements are needed to support the growing raft of battery-operated autonomous vehicles, such as Autonomous

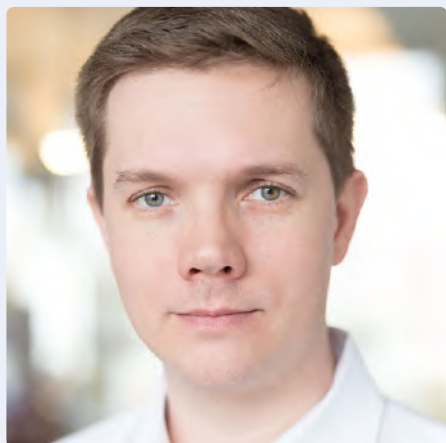


Figure 10: Jonas P. Proeger, Director Business Management at Trinamic, now part of Analog Devices. (Source: Analog Devices)

Mobile Robots (AMR), used in logistics. The motors used are also changing, with higher power densities and weight savings resulting in lower inductance motors. As a result, the motor control algorithms have to change with the control loops operating at up to 100 kHz. Devices such as their TMC4671 [7] achieve this using dedicated hardware, displacing alternative approaches requiring DSP or FPGA performance.

Silent operation is another growing requirement, offering a perception of higher quality compared to alternative products with their grinding motors and gearboxes. Here Jonas highlights their stealthChop technology that uses a voltage-controlled chopping technique to reduce audible noise by up to 10 dB in conjunction with a

bipolar stepper motor [8]. The resultant reduction in ripple current also lowers the power dissipation in the motor. The final growth area is in intelligent motors, where industrial installations require 'self-aware' motors. Such motor solutions contribute to the growing use of artificial intelligence at the system level by delivering real-time data from the very edge of the factory floor.

Power Is Nothing Without Control

It is easy to develop the perception that everything is moving to BLDC and PMSM motors. However, this is not the case. There remain plenty of applications where traditional FHP DC motors remain fit for purpose. And, with the wealth of H-bridge motor drivers and low-end MCUs with clever peripherals on offer, it is clear that the semiconductor industry still sees plenty of potential too. However, as applications target ever-higher efficiencies, move to battery operation, or require close to silent operation, engineers will need to get to grips with BLDC and PMSM. While many of the techniques used to control DC motors still apply, there are plenty of pitfalls surrounding the finer aspects of the controller implementation. A sound basis in the theory remains helpful and enables motor control solutions to be debugged. But, to save time, it makes sense to draw upon the experience of experts in this dynamic field and review the highly integrated silicon solutions the market has to offer. ◀

210562-01

Questions or Comments?

Do you have technical questions or comments about this article? Email the author at stuart.cording@elektor.com.

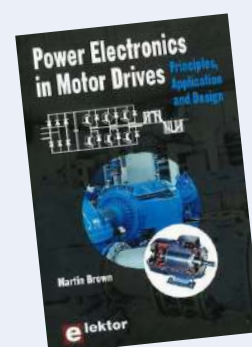
Contributors

Design and text: **Stuart Cording**

Editing: **C. J. Abate**

Illustrations: **Patrick Wielders**

Layout: **Giel Dols, Harmen Heida**



RELATED PRODUCTS

➤ E-book: Power Electronics in Motor Drives (SKU 18517)
www.elektor.com/18517

WEBLINKS

- [1] C. Valens, "How to Choose Between High-side and Low-side Switching," Elektor, August 2019: <https://bit.ly/3H38uA5>
- [2] "Understanding PID Control, Part 1: What Is PID Control?," The MathWorks, Inc., May 2018: <https://bit.ly/3CZfcor>
- [3] Infineon IFX9201SG Product Page: <https://bit.ly/3CZFEmD>
- [4] Infineon H-Bridge Kit 2GO Product Page: <https://bit.ly/3ww91FZ>
- [5] D. Hou, "PID Control on PIC16F161X by using a PID Peripheral," Microchip Technology Inc., 2015: <https://bit.ly/3DbGCHy>
- [6] Toshiba TXZ4A+ Series MCUs: <https://bit.ly/3bXUSId>
- [7] Trinamic TMC4671 Product Page: <https://bit.ly/3bWO2CA>
- [8] B. Dwersteg, "stealthChop Performance (qualitative)," TRINAMIC Motion Control GmbH & Co. KG, February 2018: <https://bit.ly/3CZgxvt>

Large Electric Motors

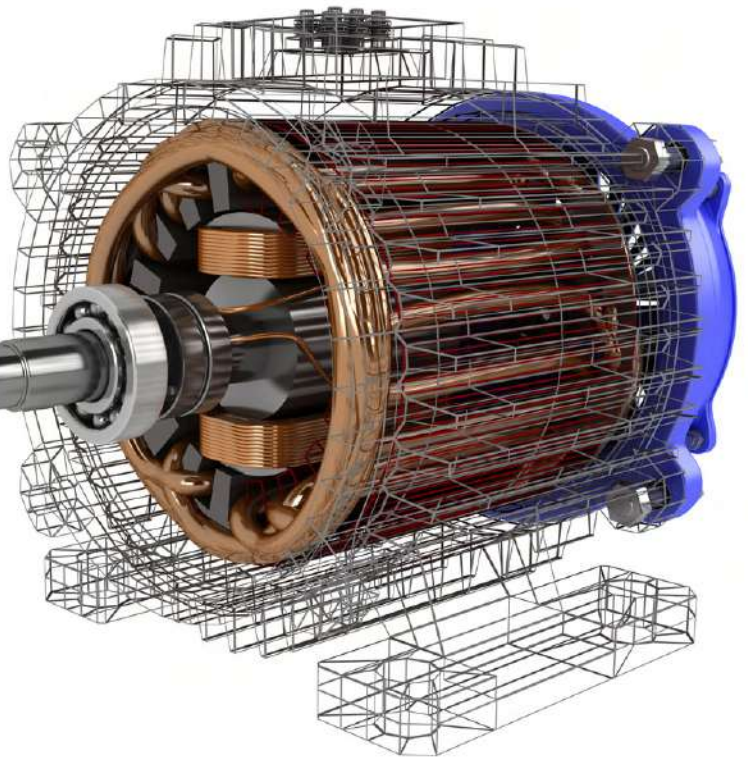
Basic Principles and Useful Information



By Dr Thomas Scherer (Elektor)

Electric motors come in all shapes and sizes and have found use in a vast range of applications for many decades. With the foreseeable widespread replacement of internal combustion engines by electric drive trains in vehicles, electric motors will soon be seen as the primary way to create mechanical motion. That alone makes them worth a closer look.

Electric motors can be found wherever electrical energy is to be turned into mechanical energy of whatever kind. They have many advantages over internal combustion engines and other non-electrical methods for producing mechanical energy such as water mills and windmills. Principal among these are small footprint, low weight, long service life and high efficiency. In larger machines, efficiency can be considerably more than 90%. The tiniest examples can be found in luxury non-mechanical watches. At power levels up to 100 W, we find the conventional 'universal' motors, and special variants such as stepper motors, servos, shaded-pole motors, small synchronous and induction motors and the increasingly popular brushless DC (BLDC) motors are widely found. An article by Mathias Claussen elsewhere in this issue of *Elektor* [1] describes how to drive smaller motors, and my article from three years ago [2] discusses motor drive techniques in general. Below we will be concentrating primarily on the principles behind the larger motors increasingly found not just in appliances (e.g., vacuum cleaners and washing machines), but also in electric vehicles and, most importantly, in industry (e.g., in machine tools).



Motor Types and Some Rules of Thumb

As mentioned above, this article will not look at lower-power motors, in particular the types of motor (shaded-pole, capacitor-run, stepper and servo motors) that are commonest at lower power levels. We will also not consider linear motors, although they are commonly used in hard disk drives for head positioning, and, at rather higher power, in other types of actuator. High-power linear motors (in the megawatt range) are also outside our scope: their military applications (e.g., in railguns) to accelerate projectiles require highly specific drive techniques, and we will not delve into this exotic subject.

Instead we will confine ourselves to the four types of motor listed in **Table 1**. In practice these cover more than 90% of application cases at power levels from under 1 kW to tens or hundreds of megawatts. In principle each motor type could equally well be made in small or large versions, but because of their different electrical and mechanical characteristics, each is differently suited to particular application areas, with considerable differences between each type.



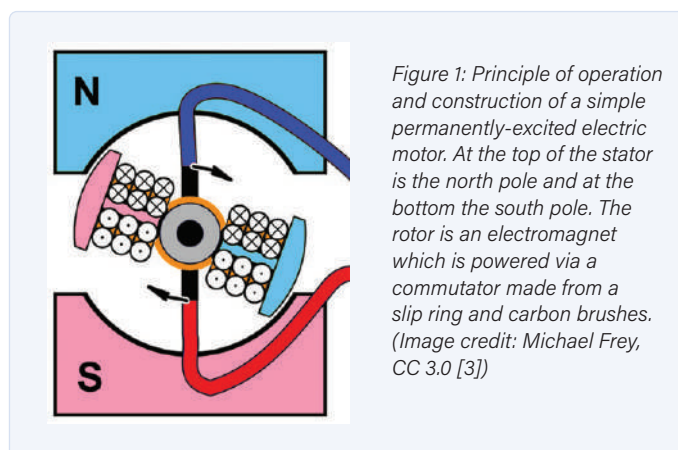
Table 1: Types of larger electric motors.

Type	Characteristics
Universal motor	A mechanical commutator creates an alternating field; many variants, series-wound and shunt-wound; high start-up torque.
Three-phase induction motor	Passive rotor (no slip rings); rotor slips behind stator field; protection against high inrush current (star-delta switching or variable frequency drive) required.
Three-phase synchronous motor	Permanent or electric excitation field required; rotor moves synchronously with rotating field; can use variable frequency drive; can be operated as a generator; can be configured with internal or external poles.
BLDC motor	Variant of synchronous motor with electronic commutation; suitable for drive trains; very reliable; permanent magnets often used in low- to medium-power applications.

It is interesting to see what myths about electric motors still circulate among engineers not specialized in the field. For example, take the excessively idealized proposition that at start-up (that is, at a speed of 0 rpm) electric motors would in theory have an 'infinite' torque. The 'in theory' part is where it goes wrong: in the absence of extreme current, extreme magnetic fields, negligible winding resistances, hyper-stable metals and pie-in-the-sky drive electronics, such a thing is impossible. Back down on Earth, all these parameters have their practical limits; all we are left with from the myth is the rule of thumb that (most types of) electric motors can indeed offer a high starting torque if their drive circuit and power supply are suitably designed. In electric vehicles this characteristic has the advantage, compared to internal combustion engines, that a multi-ratio gearbox (be it manual or automatic) can in general be dispensed with. That improves the reliability and simplicity of the drive train and reduces its weight: this last aspect is important even though the savings are more than offset by the weight of a large battery pack.

A further rule of thumb will come in useful for anyone not familiar with the subject: in an electric motor the correlation between torque and operating current at a given load is highly linear, because the current that flows is proportional to the magnetic field generated in the windings; this is proportional to the resulting magnetic attractive or repulsive force, which in turn correlates with torque.

And another, somewhat more rough-and-ready, rule of thumb is that the maximum possible unloaded rotation speed (ignoring mechanical considerations) depends on the power supply voltage. Why? When an electric motor is turning it creates what is called a 'back EMF' which, ignoring construction aspects, is broadly proportional to rotational speed. The term 'EMF' (electromotive force) was historically used to describe a voltage generated by a motor; by 'back EMF' we mean a voltage induced by the rotation of a motor which tends to oppose the voltage from its power supply. If we neglect inductances and losses, the resulting effective voltage across the motor's windings is the difference between the applied voltage and the back EMF. The resulting current depends almost exclusively on this difference and the ohmic resistance of the windings. If this



difference is 0 V then the maximum possible rotational speed has been achieved: the tiny residual current that flows is just enough to compensate for friction losses.

Principles of Operation

The historically earliest, and indeed the simplest, form of motor is the permanent magnet motor shown in **Figure 1**. The rotor consists of an electromagnet mounted on a bearing so that it can turn. The black dot in the middle represents the axis of rotation and the grey ring the bearing. The applied DC voltage is passed through a commutator comprising a slip-ring with copper contacts and brushes made of carbon. The polarity of the voltage applied to the electromagnet, and hence of the magnetic field, thus depends on the position of the rotor. If the polarity of the applied voltage is reversed, the direction of rotation will also reverse. It is also clear that this type of motor can act as a generator if its rotor is mechanically driven: in this case a pulsed DC signal will appear at its connections. If the commutator is dispensed with then the machine becomes an AC generator; and by applying an AC voltage we have a single-phase synchronous motor design which, however, is not really practical because it has problems starting up.

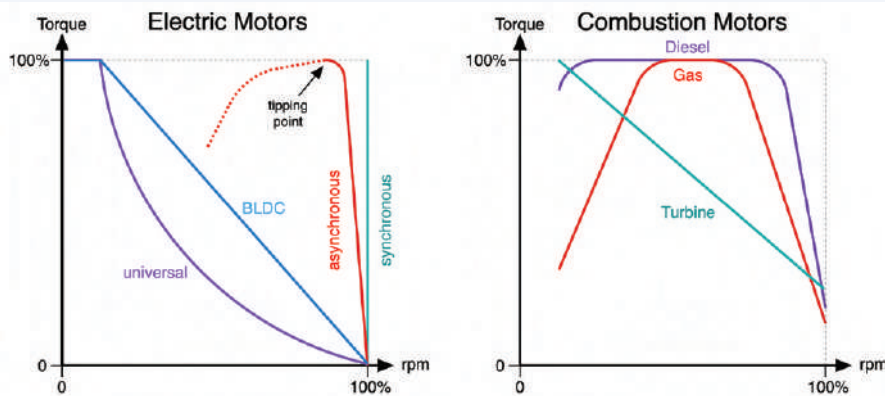
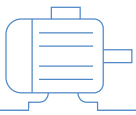


Figure 2: Idealized behaviour of torque as a function of rotational speed for various types of electric motors, compared to three types of internal combustion engines. (Image credit: Dr. Thomas Scherer)

From this simple motor design we can derive practically all other motor types. If the permanent magnet of the stator is replaced by a winding (creating another electromagnet) then the resulting design is known as a 'universal' motor. If the motor is equipped with multiple phases, then we can build induction, synchronous and BLDC motors which employ an electrically-generated rotating field. The simple motor has poor coasting properties when no voltage is applied, since the field lines of the permanent magnets always give rise to a braking effect. Separately excited motors exhibit this problem to a lesser degree.

The various principal characteristics of these motor types are shown (in idealized form) in **Figure 2**, including a comparison with current designs for internal combustion engines. The contrasting behaviours of torque as a function of rotational speed are particularly noteworthy. This comparison is far from exhaustive, and all the various characteristics have a significant influence on the suitability of various types of motor in different application scenarios.

The Universal Motor

Figure 3 shows the universal motor, invented over 117 years ago, which is the closest design variant to the simple motor shown in Figure 1. Here we simply replace the permanent magnet by a further winding which acts as an electromagnet. The advantages of the universal motor show themselves most in high power designs, where the permanent magnets required in the simpler design

become not only costly, but also very large and heavy. In the classical universal motor the field winding in the stator is connected in series with the rotor winding, forming what is known as a series-wound motor. The main advantage of the series-wound universal motor over the less widely used shunt-wound motor (not illustrated), where the stator winding is connected in parallel with the rotor winding, is that it has a very high starting torque.

Both variants can be operated either from DC or AC supplies. Designs optimized for AC operation at high rotational speeds use a core for the stator and rotor consisting of a stack of metal sheets isolated from one another in order to reduce losses due to eddy currents and hysteresis. Changing the polarity of the applied voltage during operation does not affect the direction of rotation in either design. To achieve this, it is necessary to reverse the polarity of either the stator winding or the rotor winding, but not both simultaneously.

Larger universal motors (in the kilowatt range) have a high starting current, and so, in order to protect the motor itself (and to prevent its fuses from blowing!) a start-up controller is used. In the simplest case this initially powers the motor via a series resistor to reduce the inrush current. After a short period of time, or when a certain rotational speed is reached, power is applied directly to the motor connections. Not only must the value of the series resistor be chosen according to the particular operating conditions, but also its load rating must be sufficient to cope with the integral of the current curve over the start-up period. The required start-up resistor can end up being rather physically large, and special constructions, or even devices such as liquid rheostats, can be used. For very large universal motors a single resistor is not sufficient: instead, several resistors are used in series. As the rotational speed reaches a series of thresholds, these resistors are short-circuited one by one. The simplest form of start-up protection controller is the time delay shown in **Figure 4**. More intelligent approaches measure the rotational speed, shorting the protection resistors as required, possibly in conjunction with slowly raising the operating voltage gradually using an electronic controller.

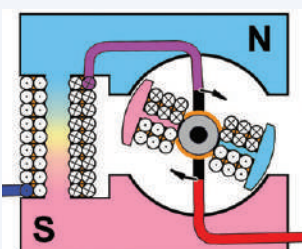


Figure 3: Principle of operation and construction of a universal motor, where the permanent magnet of a simple motor is replaced by an electromagnet. (Modifications from Figure 1: Dr. Thomas Scherer)

As mentioned above, universal motors can be run from an AC or a DC power supply. The rotational speed when loaded and under no-load conditions depends on the construction of the motor and the applied voltage. The output power under load and rotational speed can therefore be controlled using a PWM drive. If such a PWM drive circuit is already available, then it is easy to use it in conjunction with the existing microcontroller that controls it to provide start-up protection. This can be done by starting operation using a PWM signal with a very low mark-space ratio, and then increasing it gradually. If the motor should be stalled because of an excessive load, or even in the case of simple overload, it is possible for very high and potentially destructive currents to flow. It is therefore recommended to monitor not just the rotational speed of the motor but also the drive current, especially for larger motors. In simple cases this latter protection can take the form of a thermal overcurrent cutout. If the speed falls below the minimum threshold or if the current exceeds an upper limit then the motor is switched off, protecting the motor itself, its drive electronics, and any machinery that it is powering. Such protection has the advantage over simply blowing a fuse that power will not be interrupted to other machines on the same supply lines.

As the name implies, universal motors are used in a wide range of applications from household appliances, drills and other tools starting from under 100 W to motors in electric locomotives and EMUs rated in the megawatts, powered from either DC or AC.

The simple variant shown in Figure 3 suffers from uneven torque as a function of rotational position. More sophisticated designs are therefore naturally the rule. Single-phase multiple-pole designs with a variety of different stator and rotor constructions are common (see **Figure 5**) with various kinds of compensation windings which affect not just the torque behaviour but also the amount of brush sparking. In turn, this also affects the reliability of the commutator and other properties. Looking at the relationship between torque and rotational speed in Figure 2, we can see that at a given voltage the speed varies with the load. To stabilize the speed an active control loop is required, along with a speed sensor.

The Induction Motor

The induction motor, whose basic construction is illustrated in **Figure 6**, has some decisive advantages in comparison to the universal motor. The most important of these is the complete absence of a commutator: apart from the bearings the design is wear-free and hence extremely reliable. Larger induction motors are almost invariably three-phase designs, and multi-pole designs of this type give a steady torque over each rotation. The popular internal rotor design, as pictured, uses the rotating magnetic field of the stator: under load, the rotor turns slightly slower than the stator field and this induces a current in its windings, and hence also a magnetic field. The corresponding back EMF in turn induces an increased current flow in the rotor winding; hence the term 'induction motor'. **Figure 7** shows a cut-away induction motor rated in the low kilowatts. The number of poles in the design determines the nominal rotational speed of the motor: doubling the

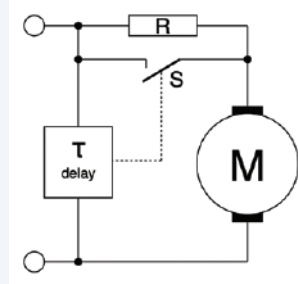


Figure 4: Simple start-up protection circuit using a series resistor to limit inrush current.

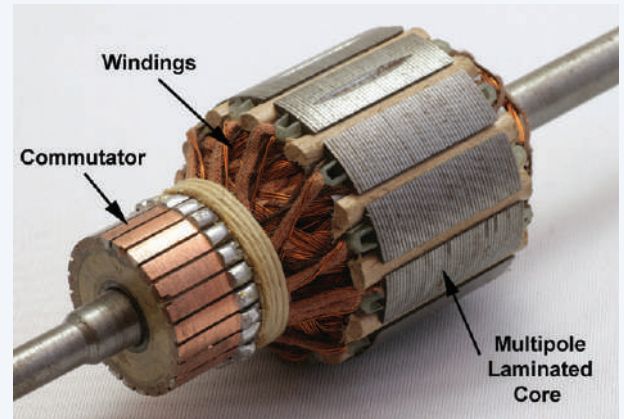


Figure 5: Rotor of a typical single-phase universal motor. The number of contacts on the commutator corresponds to the windings and to the number of poles. (Image credit: Sebastian Stabinger, CC 3.0 [4])

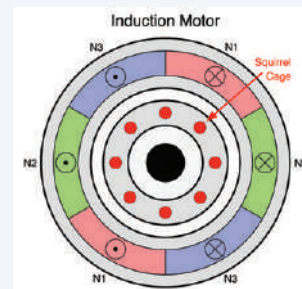


Figure 6: Diagram of the construction of a three-phase induction motor (internal rotor). The rotor consists of a 'squirrel cage' with short-circuited windings. (Image credit: Dr. Thomas Scherer)



Figure 7: Cut-away induction motor rated in the low kilowatts. The stator windings and the short-circuited rotor can easily be seen. (Image credit: S. J. de Waard, CC 3.0 [5])

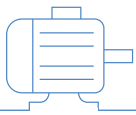


Figure 8: A small variable frequency drive for induction motors rated at a few kilowatts. This electronic switching circuit allows the rotational speed to be set. Larger versions are also available, offering more advanced functions. (Image credit: C. J. Cowie, CC 3.0 [6])

and the amount of slip depends on the load, but is nevertheless closely coupled to the speed of rotation of the field (and hence the frequency of the AC supply). As the load on the motor is increased a threshold is eventually reached after which the torque falls off sharply. The motor then stalls and the current rapidly increases. The usable range of rotational speeds can be adjusted (either higher or lower) over a considerable range by powering it via a variable frequency drive (see **Figure 8**). Such drives are particularly popular in industrial machines.

Larger induction motors require a high current at start-up: up to a factor of 10 more than the nominal current during operation. There are various ways to reduce the inrush current. In simpler cases, up to say 11 kW, a star-delta switch (**Figure 9**) is used: the current, and hence power draw, in the star wiring configuration is one third of that in the delta configuration. The switchover from star to delta configuration is typically made once the motor has reached at least 75 % of its rated speed. Care must be taken that the load is not too great during the start-up period so that the stall torque threshold is not reached. When using a variable frequency drive this precaution is not necessary as the drive frequency and hence the speed can be increased gradually at start-up.

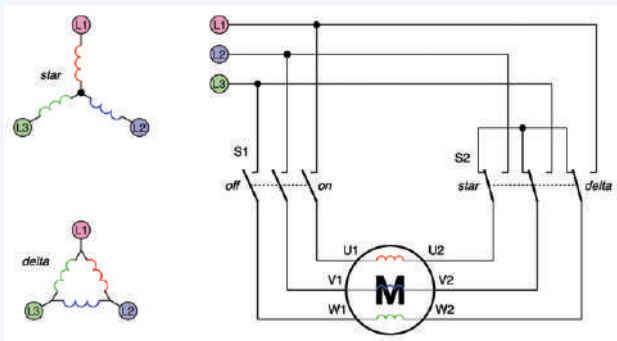


Figure 9: Star-delta protection switching. This offers a simple way to prevent large inrush currents at motor start-up.

Because of their extreme robustness and high efficiency this kind of motor can be found in machines of all kinds in industrial production as well as in electric vehicle drive trains. Tesla uses compact high-speed induction motors with a reduction gearbox in their current models, achieving the required very wide range of rotational speeds using a clever motor control system combining current and speed monitoring circuits. The use of electronic control by and large overcomes the most significant disadvantage of the motor, that the speed is closely coupled to the frequency of the AC supply. If the rotor is turning faster than the rotating stator field, then the induction motor functions as a generator: this can allow for energy recovery which is important both in vehicles and in fixed machines.

Synchronous and BLDC Motors

Essentially a synchronous motor is a universal motor operated from a three-phase supply. In one popular design the rotor winding is supplied via separate slip rings: in contrast to the universal motor a commutator is not required and so the rings are unbroken and hence friction is reduced. **Figure 10** shows one design; it is also possible to fit permanent magnets to the rotor, removing the need for slip rings entirely. For larger fixed motors the separately excited design illustrated is more economical and more reliable from a thermal point of view: permanent magnets can easily be damaged by even brief exposure to high temperatures. There are also designs with an external rotor, where the interior part is fixed and the external rotor is fitted with permanent magnets, obviating the need for slip rings. These types have a long service life and are maintenance-free. As can be seen from the plot of torque against rotational speed, the speed remains constant and is rigidly coupled to the speed of the rotating field. The rotor does not slip under load; instead, it lags by a phase angle relative to the rotating field by an

number of poles halves the motor speed at a given frequency of the three-phase supply.

As can be seen from the plot in Figure 2, the induction motor has a very steep torque curve and the rotor always rotates at just under the rotation rate of the applied field. This is known as 'slipping',

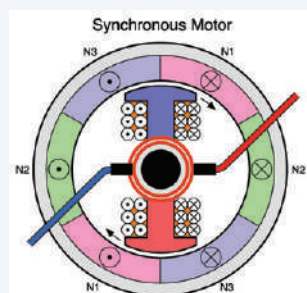


Figure 10: Diagram of the construction of a simple three-phase synchronous motor. The rotor can instead be fitted with permanent magnets to avoid the need for a commutator. (Image credit: Dr. Thomas Scherer)

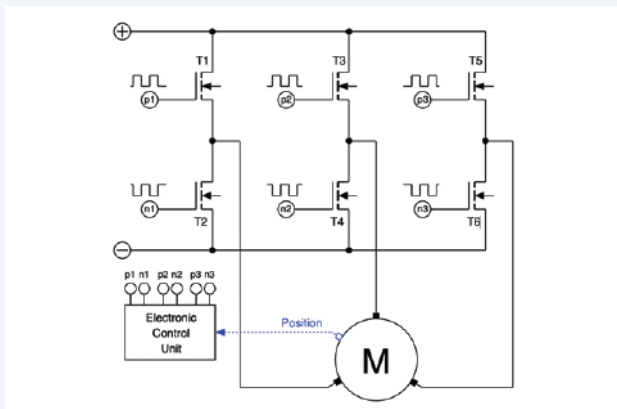


Figure 11: Principle of electronic commutation for a BLDC motor. Each half-bridge driver receives a pair of complementary drive signals with an appropriate phase delay.

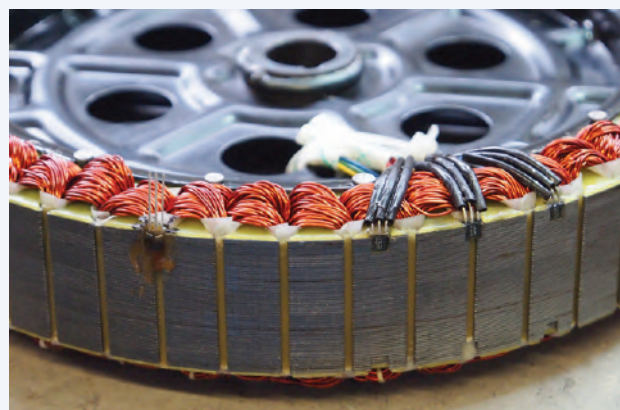


Figure 12: Stator of a 2 kW BLDC wheel hub motor with external rotor. The four-pole windings and laminations can easily be seen, as can the location of the Hall effect sensors used to detect the angular position of the rotor. (Image credit: Dr. Thomas Scherer)

amount that depends on the load. Like the other types, this kind of motor can be operated as a generator. Synchronous motors are made with multiple poles to give a more steady torque over each rotation; here also the number of poles is inversely proportional to rotational speed.

The big advantage of the synchronous motor is hinted at by its name: it is possible to achieve very precise, constant and load-independent rotational speed without a controller. Since, unlike in a universal motor, the slip rings are unbroken and the excitation energy is relatively small, the carbon brushes experience very little wear. If permanent magnets are used in the rotor then it is possible to construct extremely robust motors in small form factors. One disadvantage is that it can be relatively difficult to start the motor: for this reason larger synchronous motors sometimes include an extra rotor winding that is short-circuited, so that the motor can be brought up to speed as if it were an induction motor. At start-up the same protection techniques can (and, in some cases, must) be used as in the case of the induction motor, including the protection features offered by modern variable frequency drives.

The rotating field in the stator windings of a synchronous motor can be generated by driving the windings from a DC power source using three half-bridge drivers, rather than from a three-phase AC power supply (see **Figure 11**). The half-bridge drivers are electronically controlled to create the three-phase waveform, just like in a variable frequency drive. The result is called a brushless DC (or BLDC) motor. In this case we need a sensor to detect the angular position of the rotor, and we use this information to clock the half-bridge driver circuit. This forms a quasi-synchronous motor that autonomously creates its own rotating field and whose rotational speed depends on the supply voltage and on the load. In contrast to the universal motor the construction is straightforward, and if permanent magnets are used then the motor is practically maintenance-free.

The torque-versus-speed characteristic is also superior. If a PWM controller is added, the no-load rotational speed can be adjusted by changing the mark-space ratio of the control signals. If the PWM signal is regulated using the supply current as a reference signal, the current will correlate very linearly with torque at a given load and speed.

Both types of motor, over a wide range of power ratings, are used in fixed machines. BLDC motors are in general better suited to vehicle drive trains because of the possibility of building commutator-free solutions and because of the sophisticated degree of control that is available. A special variant is the 'wheel hub motor' (see **Figure 12**) which is frequently used in electric bicycles and other low-to-medium power drive applications. These motors feature an external rotor assembly, usually fitted with permanent magnets. The large number of poles results in a low rotational speed and high torque. Torque is constant as a function of rotational position and reliability is excellent. However, for higher-power applications, the necessary permanent magnets are both expensive and heavy. A wheel hub motor therefore represents a considerable unsprung mass, which impairs ride comfort. It is for this reason that electric vehicles more usually employ high-speed and compact induction motors which drive the wheels via flexible couplings.

Moreover...

Even though electric motors are in widespread use and seem likely to supersede all other motor technologies, there are still a few things to be aware of when using them. Today's electric motor designs are very refined and there does not seem to be scope for further significant innovation. However, there will doubtless be many evolutionary optimizations: these will come not only from improvements to existing materials and to construction and manufacturing technology, but also, most importantly, from intelligent electronic control. Power semiconductors are improving all the




Table 2: Top 20 manufacturers of electric motors (as of 2019) [8].

Ranking	Manufacturer	Country
1	Siemens	Germany
2	Toshiba	USA
3	ABB	Switzerland
4	Nidec Motor	Japan
5	Rockwell Automation	USA
6	Ametek	USA
7	Regal Beloit	USA
8	Johnson Electric	Hong Kong
9	Franklin Electric	India
10	Allied Motion	USA
11	Faulhaber	Germany
12	General Electric	USA
13	Danaher Motion	USA
14	WEG	Brazil
15	maxon motor	Switzerland
16	TECO Westinghouse	Taiwan
17	Hitachi	Japan
18	Lincoln Electric	USA
19	Piela Electric	USA
20	Dumore Corporation	USA

time, becoming faster and more efficient, and computing power is becoming ever cheaper. That means that complex algorithms can be implemented, improving the efficiency of the overall combination of the electromechanical motor and the electronic control system.

In the end the choice of an electric motor for a particular application is always a quest for the optimum trade-off among a large number of factors. Making that choice demands a lot of know-how, and customers look to the manufacturers of electric motors for high-quality support and specialist advice. There are hundreds of such manufacturers, and it is difficult to obtain a clear overview of

the market. The total turnover of all electric motor manufacturers in 2020 was some \$150bn [7], and analysts expect the market to grow at 6% per annum over the coming years. **Table 2** lists the twenty biggest manufacturers: firms based in the USA dominate, but the market leader is a German company. If you are looking for something a little out of the ordinary, you will need more than just a knowledge of the basic principles: be prepared to get deeply involved in the nitty-gritty of the subject! 

210555-01

Contributors

Author: **Dr Thomas Scherer**

Editor: **Jens Nickel**

Translation: **Mark Owen**

Layout: **Giel Dols**

Questions or Comments?

If you have technical questions or comments on this article, feel free to e-mail the Elektor editorial team at editor@elektor.com.



RELATED PRODUCTS

- **SmartPi 2 – Smart Meter for Raspberry Pi (including three current probes) (SKU 18165)**
www.elektor.com/18165
- **Current Probe for SmartPi 2 (SKU 18167)**
www.elektor.com/18167
- **PeakTech 4350 Clamp Meter (SKU 18161)**
www.elektor.com/18161

WEB LINKS

- [1] M. Claussen, 'Motor Drive using H-Bridges', Elektor January/February 2022: <https://www.elektormagazine.com/210491-01>
- [2] T. Scherer, 'Electric Motor Control', Elektor January/February 2019: <https://www.elektormagazine.com/magazine/elektor-70/42362>
- [3] Simple electric motor: https://commons.wikimedia.org/wiki/File:Animation_einer_Gleichstrommaschine.gif
- [4] Rotor of a universal motor: https://commons.wikimedia.org/wiki/File:Kommutator_universalmotor_stab.jpg
- [5] Cut-away induction motor: [https://commons.wikimedia.org/wiki/File:Rotterdam_Ahoy_Europort_2011_\(14\).JPG](https://commons.wikimedia.org/wiki/File:Rotterdam_Ahoy_Europort_2011_(14).JPG)
- [6] Variable frequency drive: https://commons.wikimedia.org/wiki/File:Small_variable-frequency_drive.jpg
- [7] The market for electric motors: <https://www.grandviewresearch.com/industry-analysis/electric-motor-market>
- [8] Ranking of electric motor manufacturers: <https://blog.technavio.com/blog/top-20-electric-motor-companies>

Getting Started with the ESP32-C3 RISC-V MCU

By **Mathias Claußen** (Elektor)

Curious about Espressif's ESP32-C3? A single-core, cost-effective alternative to the ESP8266, the ESP32-C3 uses the open RISC-V instruction set architecture. Let's take a look.

The ESP32-C3 from Espressif has been eagerly awaited. It has just one core humming away at its heart instead of the usual two cores in ESP32-based controllers. This core, however, uses the free and open RISC-V instruction set architecture which competes with ARM-based controllers widely used for IoT applications. I recently took a look at the ESP32-C3 and the ESP32-C3-DevKitC-02 to see how it performed in the lab. Let's review what I learned.

Anyone following the me on Twitter (@ElektorMathias) [1] might have noticed a couple of ESP32-C3 DevKits sitting on my desk. Followers of our *Lab Notes* in June 2021 [2] will also be aware that software support for the ESP32-C3 is still a work in progress. The development kit under the spotlight in this review is fitted with the revision 2 version of the ESP32-C3 chip. These preliminary versions come with an A4 errata sheet (printed on both sides) listing the problems which have so far been identified.

Amongst other things, the revision 2 suffers from insomnia — that is, it draws a high level of power in deep sleep mode, and the USB/JTAG serial adapter in the chip doesn't function. The more recent revision 3 version should have these problems resolved, as I noted on Twitter [3].

The Espressif ESP-IDF repository supporting the ESP32-C3 is also a work in progress and may contain bugs. Since Arduino support is based on it, any bugs will migrate onto this platform too. At the time I am writing this article, the Development Release of the Arduino Support Package will need to be installed in the Arduino IDE you are running. To do this, enter the following link under *Preference* in the *Additional Boards Manager* URLs:



Figure 1: The ESP32-C3 DevKitC-02.

https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_dev_index.json
Which installs the 2.0.0-rc1 version.

Purchasers of dev kits based on the ESP32-C3 should be supplied with the latest rev.3 version of the chip, as can be read in the forum entry at ESP32.com [4]. The preliminary rev.2 buggy version was only fitted to a limited numbers of the first dev kits supplied.

The ESP32-C3 DevKitC-02

First, we will take a look at the ESP32-C3 and the DevKitC-02 (**Figure 1**). The ESP32-C3 chip is effectively a successor to the Espressif ESP8266 microchip. Like the ESP8266, it also uses a single core processor which can be clocked at up to 160 MHz and has a 2.4-GHz BGN Wi-Fi communications chip with full TCP/IP stack implementation. But the similarities with the ESP8266 end there. The ESP32-C3 includes many

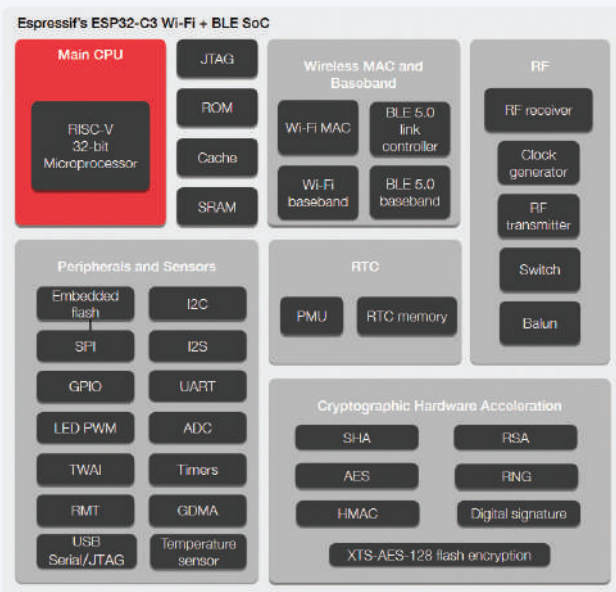


Figure 2: Block diagram of the ESP32-C3.

- > RISC-V CPU clocked at 160 MHz
- > 400 KB SRAM (16 KB Flash Cache)
- > Integrated 2.4 GHz Wi-Fi (BGN)
- > Bluetooth LE 5.0
- > Crypto Hardware Accelerator
- > 22 programmable GPIOs
- > 2 x 12-Bit SAR ADC
- > 3 x SPI (supports SPI, Dual SPI, Quad SPI and QPI)
- > 2 x UART (supports RS232, RS485 IrDA up to 5 MBd)
- > 1 x I²C (up to 800 kbit/s)
- > 1 x I²S
- > RMT (remote control peripheral devices)
- > TWAI (CAN 2.0 b compatible / ISO 11898-1)
- > PWM
- > Internal USB/JTAG adapter

Table 1: ESP32-C3 specs.

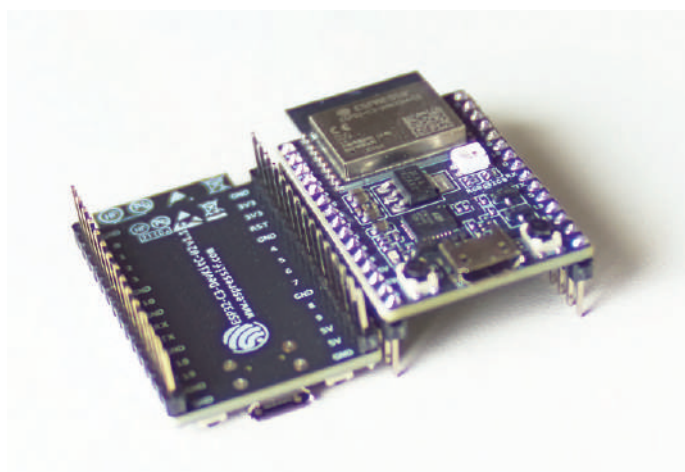


Figure 3: Both sides of the ESP32-C3-DevKitC-02.

more of the peripherals associated with the ESP32 which is Espressif's successor to the ESP8266. In addition to Wi-Fi, the ESP32-C3 includes BLE 5.0 and Bluetooth Mesh communications. It also employs the useful GPIO matrix, so that almost any function can be assigned to almost any pin. **Figure 2** shows the block diagram of the ESP32-C3, which includes a USB serial/JTAG adapter.

A block diagram of the controller functions is given in **Figure 2** and shows a clear inheritance from the ESP32 device, with a specification overview in **Table 1**. With its 384 kB of RAM, the ESP32-C3 offers almost five times more RAM space than the ESP8266 (80 kB). The main component which distinguishes the ESP32-C3 from all other ESP32 or ESP8266 chips is the processor core. While its predecessors use the Tensilica L106 or LX6/LX7 [5] RISC processor, the ESP32-C3 has a RISC-V CPU. This means that compilers and other programs from the RISC-V toolchain can be used with this core. Improvements to these compilers and the corresponding tools will thus benefit users of the ESP32-C3. It is not necessary for the community to set about creating a toolchain as it was when the ESP8266 was first introduced.

The DevKitC-02A board includes a USB-to-serial converter as well as a WS2812-compatible RGB LED. A detailed circuit diagram of the board can be found at [6]. **Figure 3** shows both sides of the board.

A Different Take on Blinky

The WS2812-compatible RGB LED installed on the board is controlled using a serial communication protocol so that we can't get away by just setting an I/O pin to turn the LED on. Fortunately, the Adafruit NeoPixel library contains routines that work with this type of LED. If it is not already installed, you can add it to your Arduino IDE in the normal way. The code in **Listing 1** makes the RGB LED flash red. You don't need to make any special tweaks to the code for it to run on the ESP32-C3. Uploading the code is just as easy as it is with an ESP32.

Thanks to the I/O matrix, WS2812 LEDs can also be used with other pins.

Read the Chip Type and Version

With ESP32 chips, it is possible to read out information stored on-chip showing its basic properties. **Listing 2** shows this information read out via the serial interface running at 115200 baud (communication at 9600 baud, with chip revision 2 using the built-in USB-to-serial converter outputs garbled characters). Here you will be able to read which version of the chip is fitted to your ESP32-C3 kit. When you see the list of known bugs associated with revision 2, you will be glad when you can confirm the chip version fitted is not affected.

Porting ESP32 Projects

Some of the most useful features of the ESP32-C3 are its built-in Wi-Fi and BLE communication capabilities. Together with SPIFFS or the LittleFS file system to manage web pages and other data on the ESP32, it makes the platform ideal for a wide range of Wi-Fi applications. Those working with an older 1.X version of the Arduino IDE will need to install a patched plugin [7] for uploading files to the ESP32 file system. The original version by *me-no-dev* [8] does not work with the ESP32-C3.

As an example, we have used the code [9] which builds an ESP32 Mini-NTP server [10]. This can provide the time of day via NTP in its own network and is written for the Arduino-Framework. After a few tweaks to the pin assignments, the code can be compiled without problems and uploaded to the ESP32-C3. This basic example shows that a lot of existing code and knowledge from the ESP32 will be directly portable to the ESP32-C3, so even newbies to RISC-V environments shouldn't find the transition too daunting. Most of the existing ESP32 examples

will also run on the ESP32-C3, and since FreeRTOS is working in the background here too, you can make use of all the advantages and disadvantages that it brings, as with an ESP32.

Since the ESP32-C3 is currently only supported in a developer branch of the ESP32 chips in the Arduino framework, not all IDEs fully support the chip yet. As the framework matures, we can expect the support level to improve.

A Single-Core Solution

The ESP32-C3 is a single-core, cost-effective alternative to the ESP8266 and has many of the peripherals found on the ESP32. The integrated USB/serial and JTAG adapter makes it easy to exchange files and



Listing 1: Blinky

```
/* This sample needs the Adafruit NeoPixel library */
#include <Adafruit_NeoPixel.h>
#define PIN      8
#define NUMPIXELS 1
Adafruit_NeoPixel pixels(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);

void setup() {
  pixels.begin(); // INITIALIZE NeoPixel strip object (REQUIRED)
  pixels.clear(); // Set all pixel colors to 'off'
}

void loop() {
  delay(1000); // wait for a second
  pixels.setPixelColor(0, pixels.Color(255, 0, 0));
  pixels.show();
  delay(1000); // wait for a second
  pixels.setPixelColor(0, pixels.Color(0, 0, 0));
  pixels.show();
}
```

WEB LINKS

- [1] @ElektorMathias on Twitter: <https://twitter.com/ElektorMathias/status/1386623477604626433>
- [2] Elektor lab notes June 2021: www.elektormagazine.com/news/elektor-lab-notes-june-2021
- [3] Tweet about USB Serial / JTAG support on ESP32-C3: <https://twitter.com/ElektorMathias/status/1392475706647584776>
- [4] Forum Thread concerning ESP32 C3 revisions: <https://esp32.com/viewtopic.php?t=21040>
- [5] Wikipedia - Cadence Tensilica Products: <https://en.wikipedia.org/wiki/Tensilica>
- [6] ESP32 C3 DevKitC-02 circuit diagram: https://dl.espressif.com/dl/schematics/SCH_ESP32-C3-DEVKITC-02_V1_1_20210126A.pdf
- [7] ESP32 Data uploader for Arduino IDE (patched version): <https://github.com/lorol/arduino-esp32fs-plugin/releases>
- [8] ESP32 Data uploader for Arduino IDE: <https://github.com/me-no-dev/arduino-esp32fs-plugin>
- [9] Github Repository Mini NTP Server with GPS: <https://github.com/ElektorLabs/180662-mini-NTP-ESP32>
- [10] Mini NTP Server with GPS at the Elektor Lab: www.elektormagazine.com/labs/mini-ntp-server-with-gps
- [11] Materials for this article: <https://bit.ly/3CD6iNs>

data via USB. You can even start debugging code (as long as nothing unforeseen occurs in revision 3 of the chip).

Thanks to the Arduino framework, existing code can be reused with the ESP32-C3 and the generous amount of RAM and Flash allows larger projects to be considered. Looking to the future, we are sure to see a whole slew of third-party ESP32-C3 boards begin to appear via European outlets, it will be interesting to see what new features these boards will offer. Anyone with an ESP32-DevKitC-02 at home

will be in a good position to begin writing and testing code now for this environment. Listings of the Arduino sketches used here can be found on our GitHub page [11].

210466-01



Listing 2: Read the chip information

```
/* From the IDF documentation at https://github.com/espressif/esp-idf/components/esp_hw_support/include/
esp_chip_info.h */
/*
typedef enum {
    CHIP_ESP32    = 1, //!< ESP32
    CHIP_ESP32S2  = 2, //!< ESP32-S2
    CHIP_ESP32S3  = 4, //!< ESP32-S3
    CHIP_ESP32C3  = 5, //!< ESP32-C3
    CHIP_ESP32H2  = 6, //!< ESP32-H2
} esp_chip_model_t;
// Chip feature flags, used in esp_chip_info_t
#define CHIP_FEATURE_EMB_FLASH    BIT(0)    //!< Chip has embedded flash memory
#define CHIP_FEATURE_WIFI_BGN    BIT(1)    //!< Chip has 2.4GHz WiFi
#define CHIP_FEATURE_BLE         BIT(4)    //!< Chip has Bluetooth LE
#define CHIP_FEATURE_BT          BIT(5)    //!< Chip has Bluetooth Classic
#define CHIP_FEATURE_IEEE802154  BIT(6)    //!< Chip has IEEE 802.15.4
typedef struct {
    esp_chip_model_t model; //!< chip model, one of esp_chip_model_t
    uint32_t features;      //!< bit mask of CHIP_FEATURE_x feature flags
    uint8_t cores;         //!< number of CPU cores
    uint8_t revision;      //!< chip revision number
} esp_chip_info_t;
*/

void setup() {
    Serial.begin(115200);
}

void loop() {
    delay(5000);
    Serial.println("esp_chip_info()");
    Serial.println("-----");
    esp_chip_info_t info;
    esp_chip_info(&info);
    Serial.print("Chip Model: ");
    switch(info.model){
        case 1:{
            Serial.println("ESP32");
        }break;
        case 2:{
            Serial.println("ESP32-S2");
        }break;
```


Questions or Comments?

Do you have any technical questions or comments about this article?
Email the author at mathias.claussen@elektor.com or contact the
Elektor team at editor@elektor.com.

Contributors

Text and images: **Mathias Claußen**
Editors: **Jens Nickel** and **C. J. Abate**
Layout: **Giel Dols**



RELATED PRODUCTS

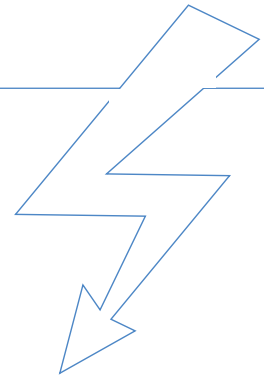
- **ESP-C3-12F-Kit Development Board with Built-in 4 MB Flash (SKU 19855)**
www.elektor.com/19855

```
        case 4:{
            Serial.println("ESP32-S3");
        }break;
        case 5:{
            Serial.println("ESP32-C3");
        }break;
        case 6:{
            Serial.println("ESP32-H2");
        } break;
        default:{
            Serial.print("Unknown Chipmodel");
            Serial.println(info.model);
        }
    }
}

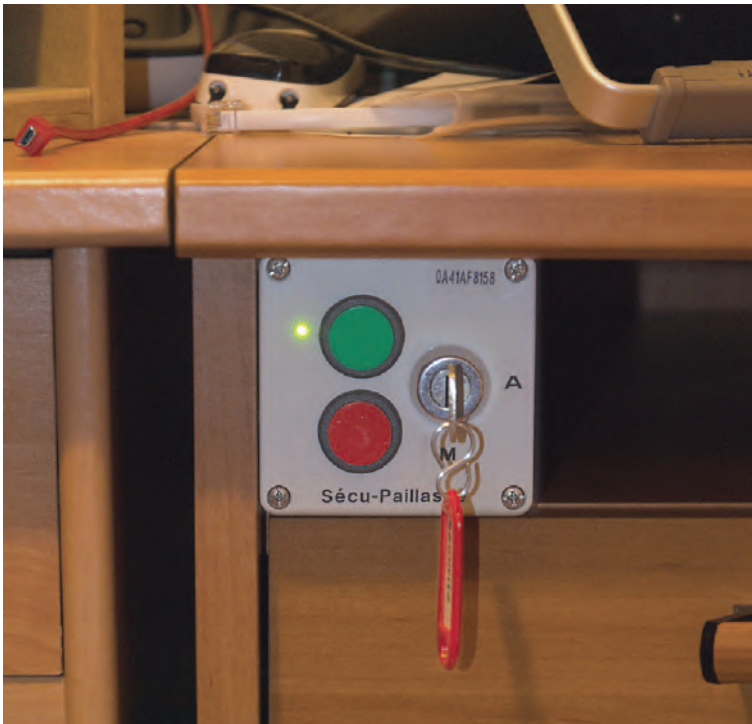
Serial.print("Featues :");
if(info.features&CHIP_FEATURE_EMB_FLASH){
    Serial.print(" Embedded Flash ");
}
if(info.features&CHIP_FEATURE_WIFI_BGN){
    Serial.print(" WiFi (BGN) ");
}
if(info.features&CHIP_FEATURE_BLE){
    Serial.print(" BLE ");
}

if(info.features&CHIP_FEATURE_BT){
    Serial.print(" BT CClassic ");
}
if(info.features&CHIP_FEATURE_IEEE802154){
    Serial.print(" IEEE802.155.4 ");
}
Serial.println("");
Serial.print("Cores: ");
Serial.println(info.cores);
Serial.print("Chip Revision: ");
Serial.println(info.revision);
Serial.println("-----");
Serial.println();
}
```

Protect Yourself and Others!



DIY Master Power Switch for the Lab Bench



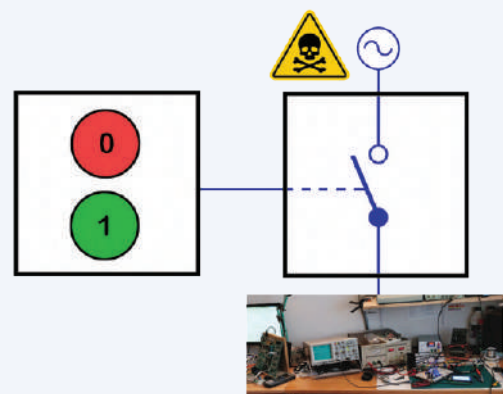
By **Philippe Le Guen** (France)

When unattended, is your electronics workspace safe for visitors? You may know what you are doing, but what about family members and friends? Will they get electrocuted when they accidentally touch something? Avoid all risks with this master bench power switch!

Why did I find it necessary to make my lab bench safer? There are several reasons, although some are quite personal. First of all, I wanted to be able to switch on and off all the equipment on the bench — instruments as well as devices under development or repair — with a simple flick of a switch. Furthermore, I do not want the bench to be automatically repowered after a power failure or (accidental) power cut. Power must be reapplied manually in such cases.

The third reason is that after my son left home, I have installed my little lab in his former bedroom where it is accessible to my grandchildren. As their tiny hands explore and

Figure 1: The circuit is split in two separate devices, the control box (left) and the power stage (right). The control box also has an additional keylock master switch.



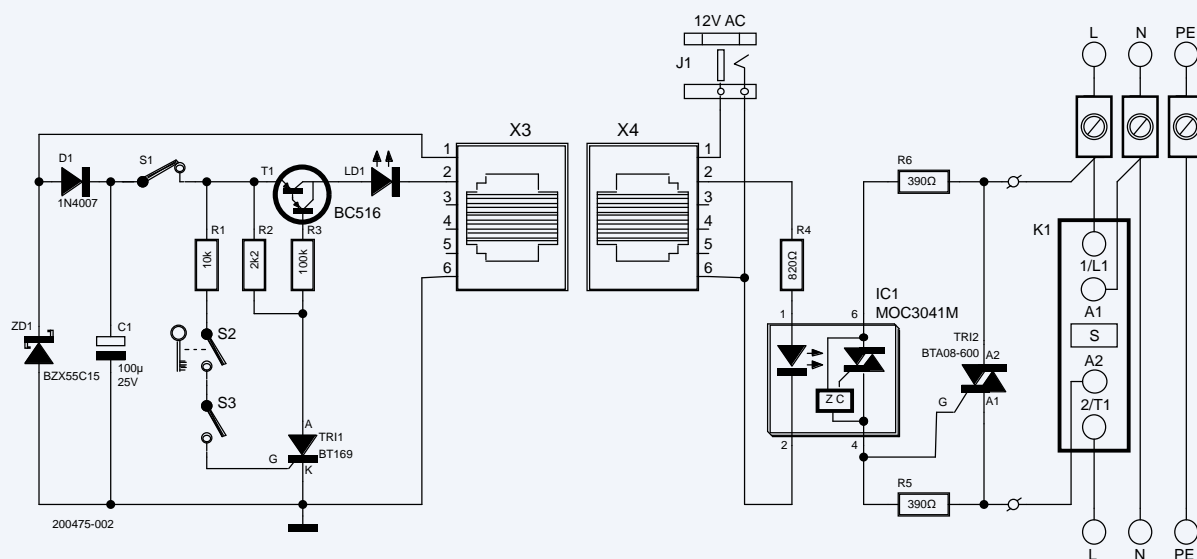


Figure 2: The left part of the circuit, the control box, is powered from the right part, the power switching stage.

Editor's note: Switching off the N conductor to disconnect from the mains does not make sense in all cases and is also not allowed, since the energy stored in a switched-off (for example inductive) consumer can in this case not flow away via the N conductor. For this reason, only a single-pole contactor is shown in the circuit diagram. If you still want to switch off the N conductor, a contactor with a leading/lagging N contact should be used.

touch everything they can, I wanted to protect the bench power supply with a key. Power off should remain possible in all cases.

Specifications

For extra safety, I wanted the control circuit to be low voltage (12 V) with the power stage galvanically isolated by an optocoupler. All this had to be a DIY electronic system, easy to build and repair thanks to the use of standard components only.

I could very well have used one of these commercially available no-volt-release (NVR) safety switches as used on machine tools, but I wanted to design it myself and furthermore, I wanted it to fit inside the electrical switch board that I installed in my lab for this reason.

A Design in Two Parts

I designed a simple electronic system using only standard components (Figure 1). The green and red pushbuttons for switching the bench on and off together with the keylock

switch are housed in a small polycarbonate case. I chose this enclosure because I already had it, and because it fits perfectly in my lab.

I don't have a reference for the keylock switch as it was scrounged off something. It should be a non-momentary type, lockable in both positions. You can refer to [1] for inspiration.

The power stage is housed in a small DIN rail enclosure fixed on one of the rails of the electrical switch board. It is assisted by K1, a 230-V, 20-A contactor for off-peak rates. This allows switching on the bench manually (with 's' inside K1) in case of failure of the control circuit (you never know what happens). Besides, I had one lying around.

The complete circuit is shown in Figure 2. The control box is located at some distance from the contactor and its interface, and the two are connected by a cable with RJ11-type plugs on both ends. I used standard Cat 5 Ethernet cable for this.

The Low-Voltage Supply

The cable transports the 12-VAC supply voltage for the control box and its output signal. The AC voltage is rectified by D1 and filtered by C1. A 15-V Zener diode (ZD1) protects against overvoltage.

The 12-VAC power supply used to be a 12 VDC power adapter until its rectifier died. Today it is nothing more than a transformer with Class II insulation. A 12 VDC supply will work too. In this case D1 would serve as a protection against polarity inversion. C1 provides filtering. Of course, nothing prevents you from increasing its value to make the circuit less sensitive to noise and glitches.

Thyristor Memory

The On state — after pressing the green pushbutton S3 — is memorized by means of a thyristor (TR1), which drives Darlington transistor T1 and optocoupler OK1. The pulse produced by the pushbutton through resistor R1 causes TR1 to conduct, which will remain

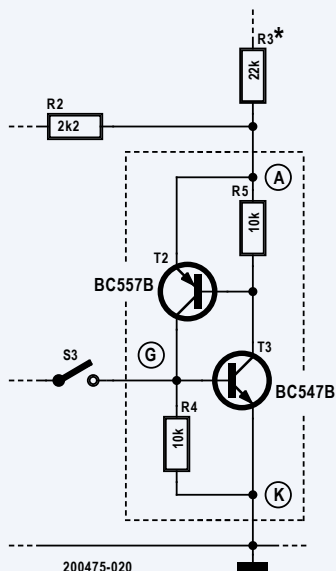


Figure 3: A PNP/NPN transistor pair can replace the thyristor.

in this state until its load current (through R2) disappears. This is possible by either applying a negative pulse to its gate or by removing its supply. The latter method was easily implemented with the red *Off* pushbutton.

When TRI1 is conducting, the base of T1 is pulled to 0 V and so it conducts too. R3 polarizes T1, while R2 limits the current flowing through the thyristor. The LED LD1 and the LED of OK1 both turn on, switching on triac TRI2 that in turn activates the power relay. The current through the LEDs is limited by R4 to about 10 mA.

When the red pushbutton (S1) is pressed, TRI1's supply voltage disappears. TRI1 blocks immediately, which makes T1 block too. The current through OK1 stops flowing and triac TRI2 stops conducting, deactivating the power relay in the process. Note that S1 must be a normally-closed (NC) type.

Optocoupler OK1 only switches during the zero crossing of the AC mains voltage, when

the current is zero. The two resistors R5 and R6 have the recommended values as found in the datasheet of OK1.

No Snubber Circuit

Using a triac with an inductive load such as a motor or relay normally requires a protective snubber circuit, typically consisting of a low-value RC network placed between the two anodes of the triac. This is not needed here because TRI2 is a so-called snubberless type.

Your trained eye will have noticed the absence of fuses and circuit breakers. In fact, the power stage input comes from a 30-mA differential switch while the output of the power relay has a circuit breaker. But maybe I was wrong to do so. Nothing prevents you from adding fuses, of course.

Now With a Pseudo-Thyristor

I also propose a second version of my project, which does not use thyristor TRI1, but a pair of PNP/NPN transistors of type BC557B and BC547B instead. Together with two 10 kΩ

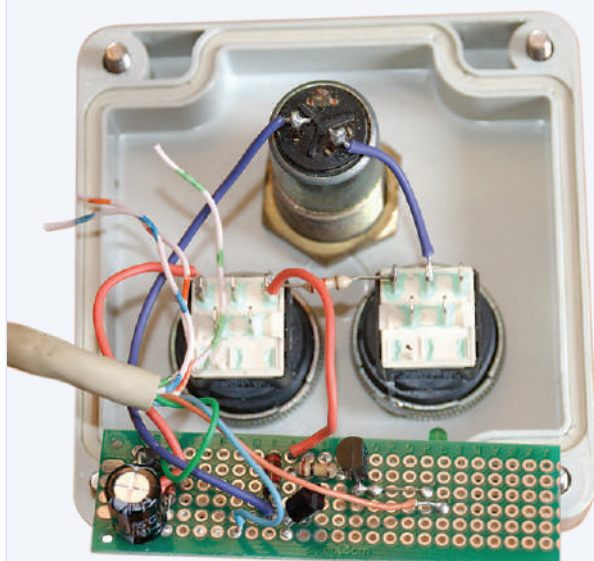


Figure 4: The insides of the control box. The small circuit board can be fixed next to the pushbuttons with a drop of hot glue.

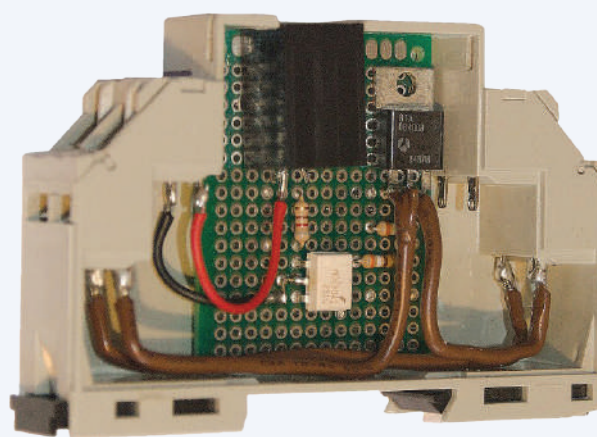
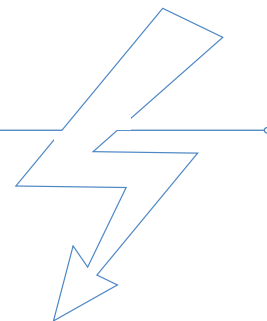


Figure 5: The power stage is built into a DIN rail module. The black thing is the RJ11 connector to which the control box is connected.



bias resistors these two transistors form a pseudo-thyristor (**Figure 3**).

On power-up, both are blocked. T1 is also blocked and no current flows into optocoupler OK1. When the On pushbutton is pressed, T3 starts conducting and pulls the base of T2 close to 0 V. Now T2 starts conducting too and applies a positive voltage to the base of T3, keeping it open even when the pushbutton is released. The current is limited by R2 and R5. As before, a current of about 10 mA will start to flow into OK1 via T1 as it has also become conductive. Power-off is again achieved by pressing the Off pushbutton.

This second circuit functions exactly as the first one built around a real thyristor. Personally, I prefer the thyristor due to its simplicity. On the other hand, in the second version there is no need to use a Darlington transistor for T1 and it may be replaced by a BC557B. If you do so, also lower the value of its base bias resistor to 22 kΩ. I used a BC516 because I couldn't find my bag of BC557s in my drawers, even though they are rather well organized.

Construction

As mentioned before, I used two enclosures to house the system that I divided into two parts:

- An almost square polycarbonate enclosure with dimensions 82 x 80 x 55 mm (Bopla M210, see **Figure 4**). This enclosure hosts the two pushbuttons and the keylock switch together with the low-voltage control electronics. The control box fits snugly under one of my two desks. The pushbuttons are accessible at all time from any of them.

- A 17.5-mm-wide DIN Rail enclosure



RELATED PRODUCTS

- **PeakTech 2240 AC Power Source (SKU 19315)**
www.elektor.com/19315
- **Fumetractor with LED Light (SKU 19092)**
www.elektor.com/19092



to host the power stage including the optocoupler and the triac (**Figure 5**). It also provides the 12-VAC supply for the control box on an RJ11 socket. I did not use a heatsink for the triac because of the low current.

I did not design any printed circuit boards for this project, I simply used a set of perforated prototyping boards bought somewhere online. The assembled circuit board can be fixed with a little hot glue inside its enclosure.

Always keep in mind that you are working with the mains, so please be very careful! The pins of the triac are quite close to each other!

Safety Matters

The system presented in this article is simple and practical to use and makes your bench a lot safer, so why not build one for your own lab? The bill of materials can be downloaded from [2].

200475-01

Contributors

Idea, design, text & illustrations:

Philippe Le Guen

Schematic drawing: **Patrick Wielders**

Editor: **Clemens Valens**

Layout: **Giel Dols**

Questions or Comments?

Do you have technical questions or comments about this article? Contact Elektor at editor@elektor.com.

WEB LINKS

- [1] Keylock switch examples: <https://www.mouser.com/datasheet/2/140/KO-345961.pdf>
- [2] Design files, datasheets & BoM: <https://www.elektormagazine.com/labs/4209>
- [3] Author's website: <https://www.pleguen.fr/index.php/securiser-sa-paillasse>

Create GUIs with Python: Spy name chooser

Make an interactive GUI application



Laura Sach

Laura leads the A Level team at the Raspberry Pi Foundation, creating resources for students to learn about Computer Science.

@CodeBoom



Martin O'Hanlon

Martin works in the learning team at the Raspberry Pi Foundation, where he creates online courses, projects, and learning resources.

@martinohanlon

So far in this series, you've learnt how to customise your GUI with a variety of different options. It's now time to get into the really interactive part and make a GUI application that actually responds to input from the user. Who could resist pushing a big red button to generate a super secret spy name?

Since you already know how to create an app, why not go ahead and create a basic window and add some text if you like? Here is some code to get you started, and this code also includes some comments (the lines that start with a #) to help you structure your program:

```
# Imports -----
from guizero import App, Text

# Functions -----

# App -----
app = App("TOP SECRET")

# Widgets -----
title = Text(app, "Push the red button to find out your spy name")

# Display -----
app.display()
```

Run this code and you should see a window with the text (**Figure 1**).

Add a button

Let's go ahead and add a button to the GUI. Add `PushButton` to your list of imports so that you can use buttons. (Be careful to use a capital B!)



Figure 1

Figure 2

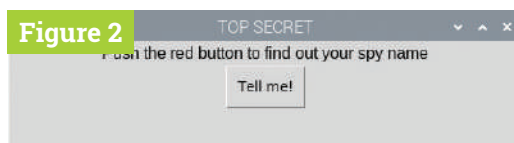


Figure 2 You now have a button

Underneath the Text widget, but before the app displays, add a line of code to create a button.

```
button = PushButton(app, choose_name,
text="Tell me!")
```

Your code should now look like **spy1.py**. Run it and no button will appear, but you'll see an error in the Shell window:

```
NameError: name 'choose_name' is not defined
```

This is because `choose_name` is the name of a command which runs when the button is pressed. Most GUI components can have a command attached to them. For a button, attaching a command means "when the button is pressed, run this command." A GUI program works differently to other Python programs you might have written because the order in which the commands are run in the program depends entirely on the order in which the user presses the buttons, moves the sliders, ticks the boxes, or interacts with whichever other widgets you are using. The actual command is almost always the name of a function to run.

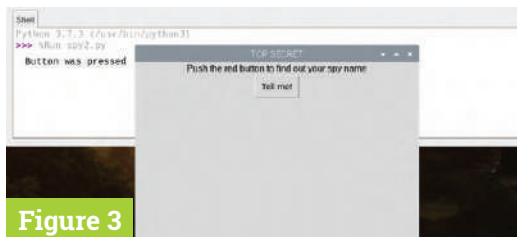
Create a function

Let's write the function `choose_name` so your button has something it can do when it is pressed.

Look at your program and find the functions section. This is where you should write all of the functions which will be attached to GUI widgets, to keep them separate from the code for displaying the widget. Add this code in the functions section:

```
def choose_name():
    print("Button was pressed")
```

Figure 1 Displaying the text in a window



▲ **Figure 3** Text is output to the Shell window

Your code should now look like **spy2.py**. The button will now appear (**Figure 2**). If you press the button, it may appear that nothing has happened, but if you look in your Shell or output window, you will see that some text has appeared there (**Figure 3**).

Instructing your function to first print out some dummy text is a useful way of confirming that the button is activating its command function correctly when it is pressed. You can then replace the **print** statement with the actual code for the task you would like your button to perform.

Inside your **choose_name** function, type a **#** symbol in front of the line of code that prints "Button was pressed". Programmers call this 'commenting out', and what you have done here is told the computer to treat this line of code as if it were a comment, or in other words you have instructed the computer to ignore it. The benefit of commenting a line of code out instead of just deleting it is so that if you ever want to use that code again, you can easily make it part of your program again by removing the **#** symbol.

Add some names

On a new line, add a list of first names. You can choose the names in your list and there can be as many names as you like, but make sure that each name is between quotes, and the names are each separated by a comma. A collection of letters, numbers, and/or punctuation between quotation marks is called a *string*, so we say that each name must be a string.

```
first_names = ["Barbara", "Woody",
               "Tiberius", "Smokey", "Jennifer", "Ruby"]
```

Now add a list of last names as well:

```
last_names = ["Spindleshanks", "Mysterioso",
              "Dungeon", "Catseye", "Darkmeyer",
              "Flamingobreath"]
```

Now you will need to add a way of choosing a random name from each list to form your spy name. Your first job is to add a new import line in your imports section:

```
from random import choice
```

spy1.py

► Language: **Python 3**

**DOWNLOAD
THE FULL CODE:**



magpi.cc/guizero

```
001. # Imports -----
002. from guizero import App, Text, PushButton
003.
004. # Functions -----
005.
006. # App -----
007. app = App("TOP SECRET")
008.
009. # Widgets -----
010. title = Text(app, "Push the red button to find out your spy name")
011. button = PushButton(app, choose_name, text="Tell me!")
012.
013. # Display -----
014. app.display()
```

spy2.py

► Language: **Python 3**

```
001. # Imports -----
002. from guizero import App, Text, PushButton
003.
004. # Functions -----
005.
006. # App -----
007. app = App("TOP SECRET")
008.
009. # Widgets -----
010. title = Text(app, "Push the red button to find out your spy name")
011. button = PushButton(app, choose_name, text="Tell me!")
012.
013. # Display -----
014. app.display()
```

This tells the program that you would like to use a function called **choice** which chooses a random item from a list. Someone else has written the code which does this for you, and it is included with Python for you to use.

In your code for the **choose_name** function, just below your lists of names, add a line of code to choose your spy's first name, and then concatenate it together with the last name, with a space in between. Concatenate is a fancy word that means

Big red button

At the moment, your button is not big or red! You used properties in part one of this tutorial series to change the appearance of your text on the 'Wanted' poster, so can you use the properties of the **PushButton** widget to change the background colour and the text size?

Note that it may not be possible to change the colour of a button on macOS, as some versions of the operating system will not allow you to do so, but you should still be able to alter the text size.

spy3.py

► Language: Python 3

```
001. # Imports -----
002. from guizero import App, Text, PushButton
003. from random import choice
004.
005. # Functions -----
006. def choose_name():
007.     #print("Button was pressed")
008.     first_names = ["Barbara", "Woody", "Tiberius", "Smokey",
009.     "Jennifer", "Ruby"]
010.     last_names = ["Spindleshanks", "Mysterioso", "Dungeon",
011.     "Catseye", "Darkmeyer", "Flamingobreath"]
012.     spy_name = choice(first_names) + " " + choice(last_names)
013.     print(spy_name)
014.
015. # App -----
016. app = App("TOP SECRET")
017.
018. # Widgets -----
019. title = Text(app, "Push the red button to find out your spy name")
020. button = PushButton(app, choose_name, text="Tell me!")
021. button.bg = "red"
022. button.text_size = 30
023.
024. # Display -----
025. app.display()
```

03-spy-name-chooser.py

► Language: Python 3

```
001. # Imports -----
002. from guizero import App, Text, PushButton
003. from random import choice
004.
005. # Functions -----
006. def choose_name():
007.     #print("Button was pressed")
008.     first_names = ["Barbara", "Woody", "Tiberius", "Smokey",
009.     "Jennifer", "Ruby"]
010.     last_names = ["Spindleshanks", "Mysterioso", "Dungeon",
011.     "Catseye", "Darkmeyer", "Flamingobreath"]
012.     spy_name = choice(first_names) + " " + choice(last_names)
013.     #print(spy_name)
014.     name.value = spy_name
015.
016. # App -----
017. app = App("TOP SECRET")
018.
019. # Widgets -----
020. title = Text(app, "Push the red button to find out your spy name")
021. button = PushButton(app, choose_name, text="Tell me!")
022. button.bg = "red"
023. button.text_size = 30
024. name = Text(app, text="")
025.
026. # Display -----
027. app.display()
```



▲ Figure 4 Outputting a spy name

‘join two strings together’ and the symbol in Python for concatenation is a plus (+).

```
spy_name = choice(first_names) + " " +
choice(last_names)
print(spy_name)
```

Your code should now resemble **spy3.py**. Save and run it. When you press the button, you should see that a randomly generated spy name appears in your console or Shell, in the same place where the original “Button was pressed” message showed up before (**Figure 4**).

Put the name in the GUI

That’s good, but wouldn’t it be nicer if the spy name appeared on the GUI? Let’s make another Text widget and use it to display the spy name.

In the widgets section, add a new Text widget which will display the spy name:

```
name = Text(app, text="")
```

When you create the widget, you don’t want it to display any text at all as the person won’t have pressed the button yet, so you can set the text to `""`, which is called an ‘empty string’ and displays nothing. Inside your `choose_name` function, comment out the line of code where you print out the spy name.

Now add a new line of code at the end of the function to set the value of the name Text widget to the `spy_name` you just created. This will cause the Text widget to update itself and display the name.

```
name.value = spy_name
```

Your final code should be as in **03-spy-name-chooser.py**. Run it and press the button to see your spy name displayed proudly on the GUI (**Figure 5**).

You can press the button again if you don’t like the name you are given, and the program will randomly generate another name for you. 🕵️



Figure 5

▲ Figure 5 The finished spy name chooser

The official Raspberry Pi magazine



The advertisement shows several covers of The MagPi magazine. The central cover is Issue 100, October 2020, featuring 'RASPBERRY PI SMART GADGETS' and 'Hack home devices with super small computers!'. Other covers include 'SPOOKY PROJECT UPGRADES FOR HALLOWEEN!', 'YOUR OFFICIAL RASPBERRY PI & SID Synth', 'JOYSTICK', 'MIDI FIGHTER', 'PICO ROBOT', '42 PAGES OF', 'WIN! AIR QUALITY KITS UP FOR GRABS!', 'FIRST PERSON SHOOTER', 'GET STARTED', '30 PROJECTS FOR MAKERS & HACKERS', and 'The Official RASPBERRY PI PROJECTS BOOK'. A large yellow and red circular badge in the center contains the following text:

£10
for 3 issues

You also receive
a **FREE** book of
your choice

From an approved selection

SUBSCRIBE magpi.cc/freebook

CALL 01293 312193

Grrr. Got stuck?

You love our Elektor print projects but need help or have an idea, question or comment about an article?

No worries. Elektor engineers, editors and community members are also **active on social media**.

You can find us here:



www.elektor.com/FB



www.elektor.com/TW



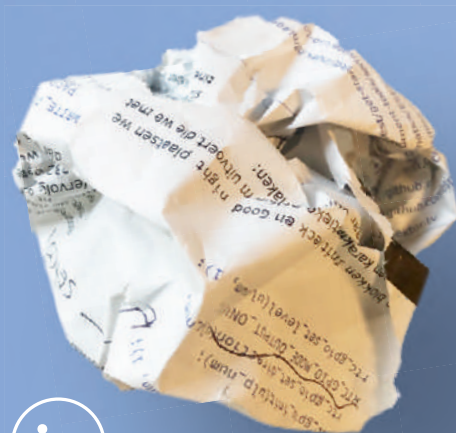
www.elektor.com/YT



www.elektor.com/Insta



www.elektor.com/LI



elektor
design > share > sell

productronica Fast Forward 2021 Winners

Exciting Technologies and Creative Engineering Solutions

By Clemens Valens (Elektor)

productronica fast forward 2021, the start-up platform powered by Elektor, offered innovative companies a unique opportunity to present their technologies to the worldwide electronics community. Here are the winners!



As it remained long unclear if productronica 2021 was going to take place as a real, live tradeshow or not, the only way forward was online. Therefore, instead of pitching in front of a live jury and audience, the 2021 productronica Fast Forward Award candidates from countries such as Hungary, Germany, Israel, and France had to record their presentations and submit them online. Supplementary information about each electronics-related start-up was provided in the forms of business plans and pitch decks.

After watching the pre-recorded elevator pitches and studying all the documentation, the productronica Fast Forward jury — which included Benjamin Klingenberg (NCAB Group, Platinum Sponsor of the event), Elektor engineering and editorial representatives — selected three winning start-ups.

Watch the
pFFWD ceremony!



FIRST PRIZE

€50,000 Elektor marketing budget

ioTech Group — Based in Jerusalem, Israel, ioTech develops new technologies for additive mass manufacturing and multi-material printing.

www.elektormagazine.com/news/iotech-group-productronica-fast-forward-2021



SECOND PRIZE

€25,000 Elektor marketing budget

Reedu GmbH & Co. KG — Reengineering Education aka Reedu aka re:edu from Münster, Germany is known for senseBox and openSenseMap.

www.elektormagazine.com/news/reedu-gmbh-co-kg-productronica-fast-forward-2021



THIRD PRIZE

€15,000 Elektor marketing budget

Pozi Development Ltd — Pozi from Budapest, Hungary provides full visibility in manufacturing and logistics, and identifies and eliminates lean waste throughout the entire supply chain.

www.elektormagazine.com/news/pozi-development-ltd-productronica-fast-forward-2021

The winning three start-ups were chosen by the jury because of their high commercial potential and the quality of their teams, two important criteria for a start-up to grow into a successful business. Congratulations to the winners!

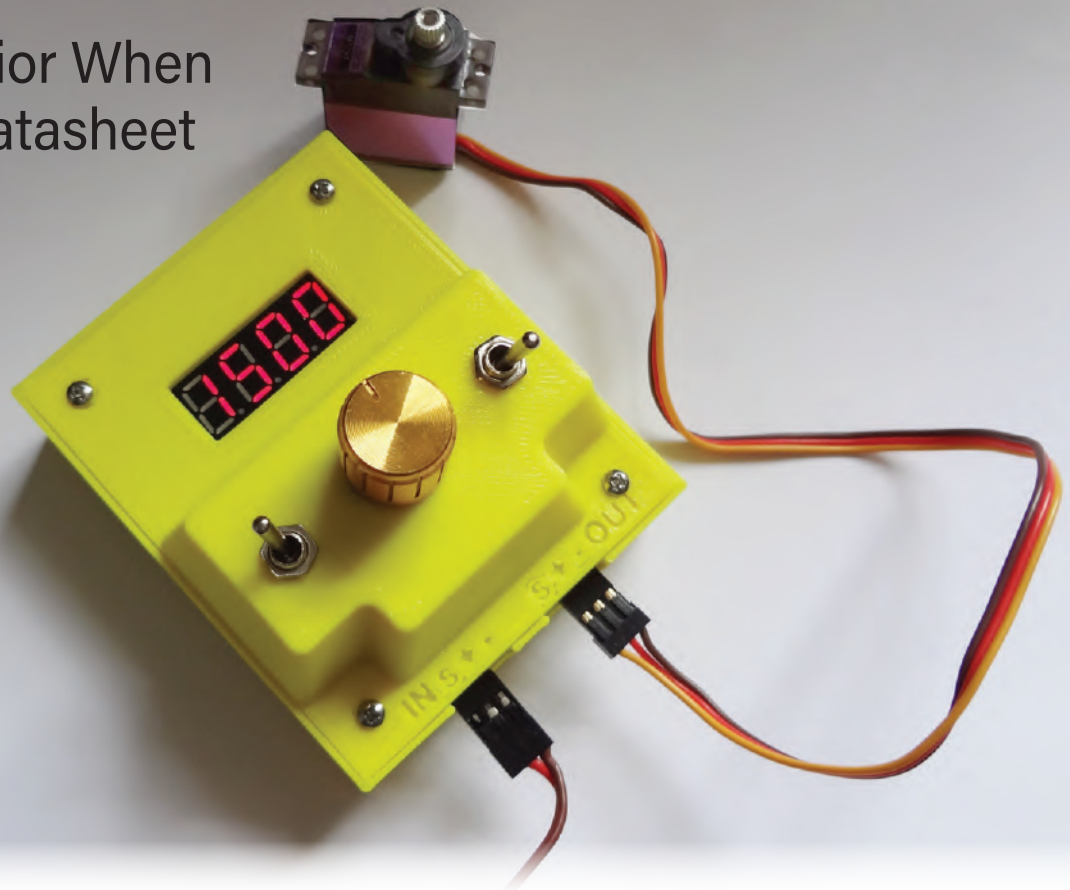
Going to electronica 2022?

The Fast Forward Award is planned to roll again at electronica 2022 (<https://electronica.de/de/>) in Munich in November of this year. Hopefully, things will be back to normal by then and we'll see you there in person!

210658-01

Versatile Servo Tester

Check Behavior When
There's No Datasheet



By **Marcelo Maggi** (United States)

Knowing the precise behavior of a servo, particularly when the datasheet is not available, is fundamental to make sure that the end application will have the expected performance. This is even more relevant nowadays, with the proliferation of all sorts of servos varying in price, quality, and performance.

The general definition of servomechanism, or servo, describes a device used to provide control of a desired operation using feedback. In electronics, and particularly when considering electromechanical actuators, a servo is a device that generates a mechanical motion controlled by an electrical pulse.

Examples of such servos can be found in various fields; for the purpose of this description, I will focus the attention on those used in radio-controlled models, since they are extremely popular and widely used, not just for models but for hobbies in general. There are several types of servos, analog and digital, with plastic or metal gears, with or without ball bearings, etc. However, all share one thing

Features and Specifications

Modes: manual (low and high resolution) and automatic (low and high speed).

Minimum pulse width: adjustable by software.

Maximum pulse width: adjustable by software.

Center position: can be adjusted manually.

Minimum step/pulse resolution: 1 μ s.

Maximum step: adjustable by software.

Signal period: fixed at 20 ms.

Pulse stability/accuracy: crystal controlled.

Power supply: 4.5 to 6.0 V.

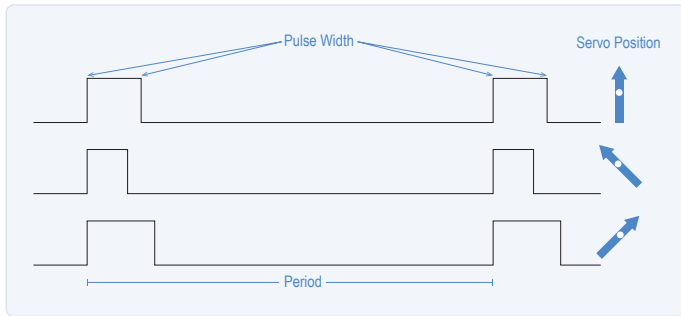


Figure 1: Servo control pulses.

in common: they can be controlled by the same electrical signal, for all practical purposes.

This signal consists of pulses, of variable width, spaced by a fixed period of time (**Figure 1**). The fixed period is usually 20 milliseconds (ms), and the pulse width is what effectively controls the servo rotation. The center position is set at 1500 microseconds (μ s) for most servos, while the extreme positions could reach up to 500 μ s on the lower end and 2500 μ s on the upper end. Usually these extremes correspond with

the -90 and +90 degrees rotation from the center point at 0. Regarding the direction, it could follow the pulse clockwise or counterclockwise, depending on the manufacturer.

The pulse amplitude, in volts, is usually the same as the servo power supply; the most common supplies are 4.8 V and 6.0 V, although nowadays other amplitudes are possible, from 3.7 V up to 14 V and over.

Why a Servo Tester?

While the manufacturer's datasheet gives all the specifications of the servo in terms of maximum displacement, speed, dead band, etc., the datasheet is not always available. And this is particularly true for the wide range of low-cost servos currently available on the market. In many cases, although some data may be available, testing the actual performance may give quite different results. At the end of the day, what matters is the actual performance of the servo you will use, not its theoretical behavior.

Designing a Servo Tester

A simple servo tester is something very easy to build, and there are plenty of examples on the web. Suffice to say that with a 555 (very popular integrated circuit) and a few other components, you can create an analog servo tester. However, if you want to measure the servo behavior in a precise way, and with visual indication of the pulse being sent to control it, a slightly more complex approach is required. One

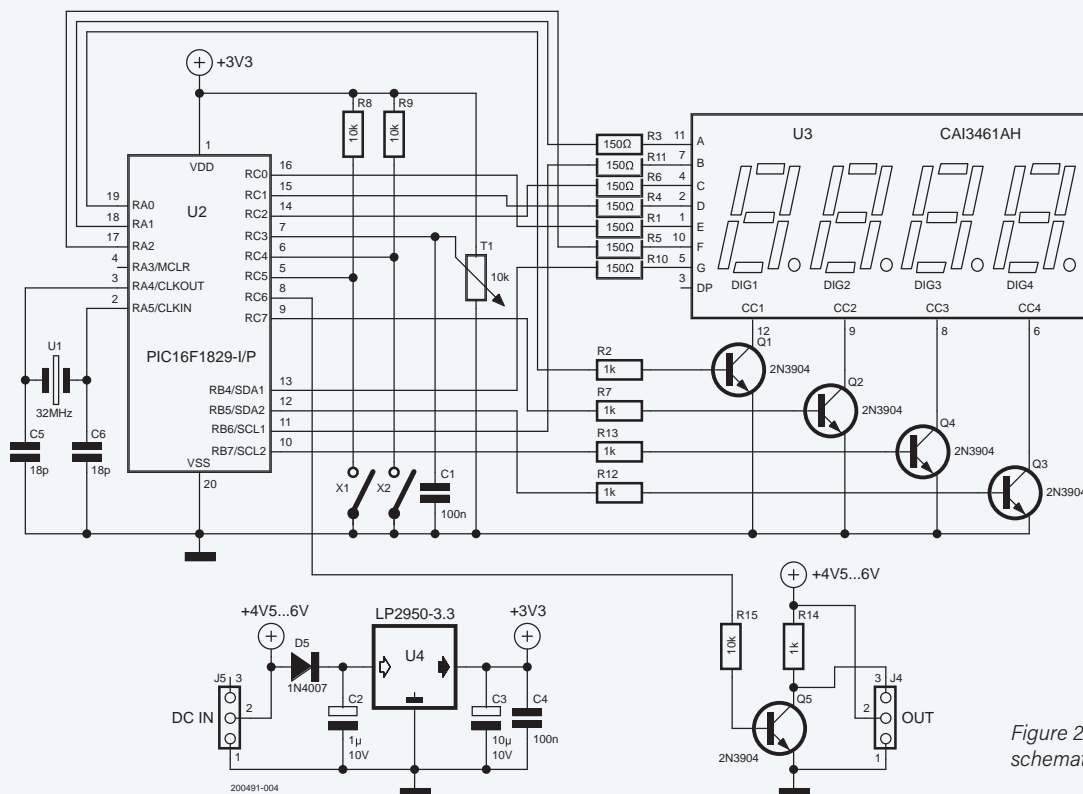


Figure 2: Circuit schematic diagram.

step above the simple 555 tester would be one with precise control of the pulse width, including a display showing this value. Cheap versions can be found on the web, but they have one disadvantage: most parameters are already programmed and cannot be changed by the user. Wouldn't it be great to have a fully customizable servo tester, in which the user has full control of all the parameters to create the desired signal, including manual and automatic testing features?

This is exactly that: a fully programmable unit that allows the user to control all signal parameters, featuring manual and automatic modes, with a clear and intuitive user interface.

Hardware

The full circuit schematic is shown in **Figure 2**. The external connections are shown in the lower part; to the left is the power input (J5 – DC IN), which can be connected to the ESC of the RC model, or to any power supply capable delivering the required current for the servo under test. The unit uses less than 15 mA, so it will not impose a considerable load to the supply. The voltage can be anything from 4.5 to 6.0 V, depending on the servo requirements.

Please note: The servo will receive the full voltage supplied by the power source. Be careful not to exceed the servo maximum ratings! To comply with most manufacturers specifications, the center connector is always the +V terminal, while the extremes are GND and SIGNAL.

The power input only requires two lines, +V and GND, but a 3-terminal connector has been mounted to receive any standard ESC connector. On the right side of the schematic is the servo output (J4 – OUT), with the three lines connected (+V, GND and SIGNAL). This will be connected to the servo under test.

The top-mid left shows the microcontroller (U2) and the user interface, a single potentiometer (T1) and two SPST switches (X1 and X2). The rotation of the potentiometer will change the pulse duration, while the switches will select the four operation modes, as shown in this chart (the various modes will be explained in the software section):

SW1	SW2	Mode
0	0	Auto, high speed
0	1	Auto, low speed
1	0	Manual, high resolution
1	1	Manual, low resolution

Finally, the top right is U3: the 4-digit 7-segment display, with its required components.

Board Layout

While this hardware can work mounted on a breadboard, a proper PCB design will make it more durable and reliable. A possible PCB layout is presented in **Figure 3**. The design (NI Multisim) and Gerber files to order the PCB at your preferred supplier are available for download at [1].

Please note that the capacitors of the microcontroller's oscillator circuit are missing in this PCB design; apparently the author's servo tester also works like a charm without C5 and C6. The capacitors can easily be added on the copper (bottom) side of the board, you may need to scratch the soldermask off the GND-plane. But of course, you can also download the design files and add the two capacitors to the PCB.

Software

Probably the most important part of this design is the software; here is where the precision and the versatility lie. The program runs in a



COMPONENT LIST

Resistors

R1,R3,R4,R5,R6,R10,R11 = 150 Ω
R2,R7,R12,R13,R14 = 1 kΩ
R8,R9,R15 = 10 kΩ
T1 = 10 k potentiometer

Capacitors

C1,C4 = 100 nF
C2 = 1 µF/10 V radial
C3 = 10 µF/10 V radial
C5,C6 = 18 pF

Semiconductors

D5 = 1N4007
Q1,Q2,Q3,Q4,Q5 = 2N3904
U2 = PIC16F1829-I/P
U3 = 4-Digit 7-segment CC LED display 3461AH
U4 = 3.3 V LDO LP2950-3.3

Miscellaneous

U1 = 32 MHz crystal
X1,X2 = switch SPST

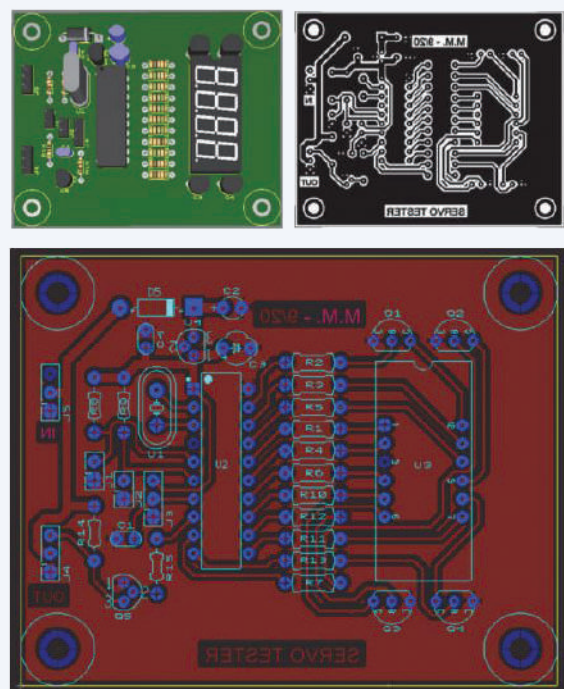


Figure 3: Mounted board layout; the potentiometer is J3, and the switches are J1 and J2.

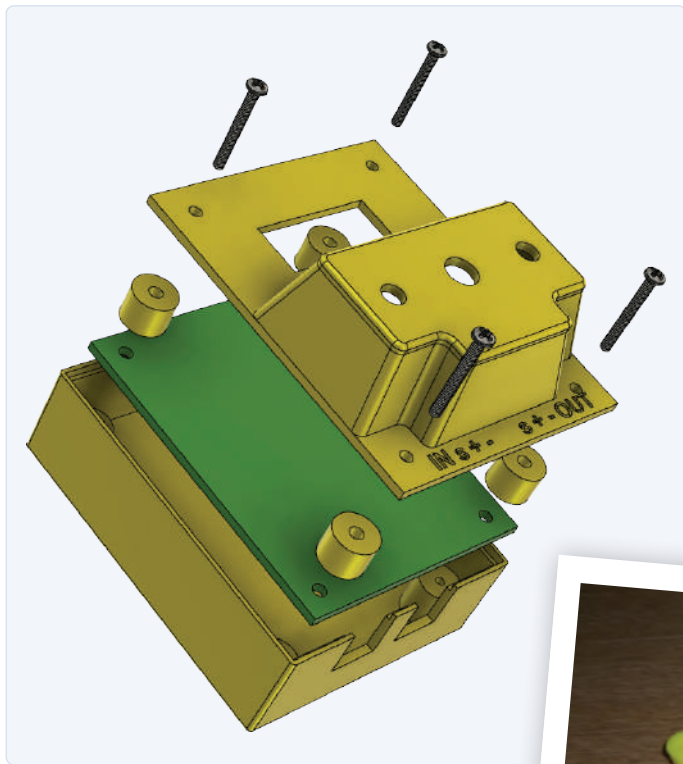


Figure 4: 3D-design of the enclosure.

PIC16F1829, using a 32 MHz crystal as the main oscillator; it is written in C, using the CCS C Compiler. The source code and the HEX-file for programming the microcontroller are available for download at [1].

The program starts with the definitions of hardware and connections, as well as key reference values: The letters **a** to **g** represent the 7 segments of the numerical display, while **d1** to **d4** correspond to the four digits. **PWM** is the output signal to control the servo and **SW1** / **SW2** are the switches. The potentiometer is connected to the analog input AN7 (PIN_C3) and it is set in the **main** function.

At the bottom, there are four very important definitions:

- **BRI**: brightness of the LED display. This number controls the persistence of each digit (in μs), thus its apparent brightness. It is set at 50 but can be changed from 1 to 200.
- **SPD**: speed of the servo movement in the auto mode. This number represents the step size (in μs) when incrementing/decrementing the pulse width; the higher the number, the faster the servo moves. Any number from 1 to 100 will work.
- **MIN**: minimum pulse width (in μs). This will be the shortest value of the pulse both in manual and auto mode. While any number

may work, it is recommended not to go lower than 500, since the servo might not be able to reach such extreme.

- **MAX**: maximum pulse width (in μs). This will be the longest value of the pulse both in manual and auto mode. While any number may work, it is recommended not to go higher than 2500, since the servo might not be able to reach such extreme.

These four parameters add huge versatility to the tester, which cannot be found in any cheap commercial unit. If you mount the microcontroller in a socket, you can easily remove, reprogram, and have a new tester whenever you need it; or, if you are satisfied with one set of parameters, just keep the same program forever.

Each digit is fully defined within a function (num) and can be selected using a **switch** statement; each time this function is called with a

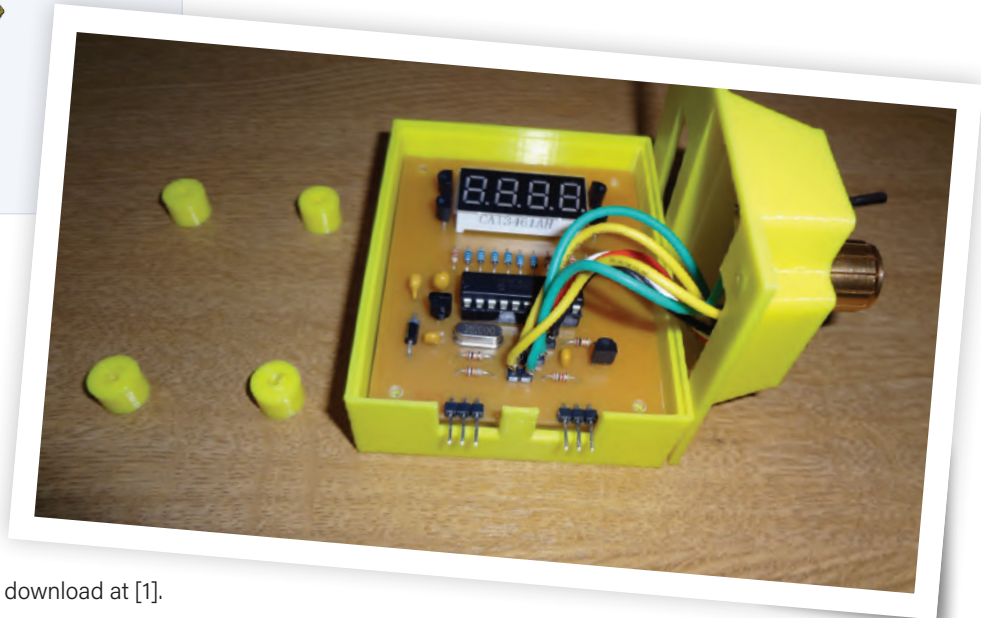


Figure 5: The PCB and connections inside the 3D-printed case.

number stored in **n**, the number will be shown in the display. To select which of the four digits is active at a given time (since the display is scanned sequentially) the function **digit** includes the selection, also within a **switch** statement. Whenever it is called, the digit stored in **z** will be illuminated. If **z** contains "5", then all digits will be off; this is used to have a clean transition from one digit to the next, with no "ghosts" in the display.

The function that brings all together (digit selection and number output) is **display**; this function takes a 4-digit number, breaks it into the individual numbers and shows them in sequence in the display, from right to left (units first), by calling the two previous functions. Here is where the display brightness is controlled, by adjusting the delay before clearing a digit.

The precise timing of the period and pulse duration is achieved by using TIMER1 interruption. Being a 16-bit counter, it will overflow at 65536, triggering the interrupt; by preloading TIMER1 with an offset value, a precise time can be measured by detecting this interruption. In sequence, the first time this INT is called we load TIMER1 with the value required for the pulse duration; next time, it will be the remaining of the total period. This process is repeated continuously (controlled by the variable `pulse`), so a precise pulse width is achieved, while keeping a constant 20 ms period.

Here we also check the status of the switches to decide if we are in *auto* mode (SW1 = 0). If so, SW2 will determine the speed of the automatic sweep (from MIN to MAX and back to MIN continuously): SW2 = 1 means normal speed (the pulse varies 1 μ s at a time) and SW2 = 0 means high speed (the pulse varies SPD μ s at a time). You may recall that SPD can be any number from 1 to 100; if set to 1, then normal and high speed will be the same (not very useful indeed).

Finally, `main` brings all together; besides the initialization of the Analog to Digital Converter (ADC), TIMER1 and interrupts, the main program loop resides here. This loop will read the ADC (thus the value of the potentiometer) and set the pulse duration accordingly.

If we are in manual mode (SW1 = 1), then SW2 is evaluated; if "1", the *low resolution* mode is active, and the potentiometer will be able to sweep the full range of the pulse width (MIN – MAX). This mode is called "low resolution" for a simple reason: the ADC can only acquire 1024 different levels, but the pulse width range may be 1400 or more μ s, so it is not possible to achieve a 1 μ s resolution within the full range. If SW2 = 0, then the *high resolution* mode is enabled. Here the full pulse width range is limited from 1469 to 1531 μ s. The potentiometer can easily be moved to change 1 μ s at a time, which is extremely useful to:

- > Put a servo in the center point (1500 μ s).
- > Measure the dead band, how many μ s the pulse needs to change to move the servo.

Making It Look Professional

With the proper case, this unit will look quite professional, and it will also protect the hardware. A simple 3D-printed enclosure (**Figure 4**) will do the job and also this design file is available for download at [1]. The PCB will be "sandwiched" between four spacers (cylinders) and the bottom case; the top case will be secured with four M2.5 x 20 mm (#2-56 x $\frac{3}{4}$) Phillips Pan Head screws. The assembly of the final unit is shown in **Figure 5**. ◀

200491-01

Contributors

Text: **Marcelo Maggi**

Illustrations: **Marcelo Maggi, Patrick Wielders**

Editor: **Luc Lemmens**

Layout: **Giel Dols**

Questions or Comments?

Do you have technical questions or comments about this article?

Email the editor at luc.lemmens@elektor.com or contact Elektor at editor@elektor.com.



RELATED PRODUCTS

- > **FeeTech Mini 360 Degree Continuous Rotation Servo FS90R (SKU 19784)**
www.elektor.com/19784



- > **FeeTech FS90MG Metal Geared Servo with Accessories (SKU 19788)**
www.elektor.com/19788



WEB LINK

[1] Software, PCB and 3D-design downloads: <http://www.elektormagazine.com/200491-01>



Modbus

Over WLAN (Part 2)

Software for the Modbus TCP WLAN Module

By **Josef Bernhardt** and **Martin Mohr** (Germany)

Built around an Espressif NodeMCU board, the module described in the first part of this series enables you to use the Modbus protocol over a WLAN. This article covers the software and Modbus module configuration.

In the first part [1] of this article series, we dealt with the project hardware. The module is built around a NodeMCU board fitted with an ESP8266 microcontroller, with an additional baseboard providing industry-compatible ports. After a brief introduction to the operating principle of the Modbus protocol, here we focus on how the module is controlled and the software used for this purpose. The lift door controller from the first part is again used in the example.

The Modbus Protocol

The Modbus protocol is widely used in the automotive sector. It operates according to the master/slave principle, and the bus master can control up to 246 slave devices. The bus nodes can be assigned addresses from 1 to 247. Address 0 is reserved for broadcast data; all data sent to this address is received by all nodes. The consistency of the individual data packets is ensured by CRC checksums. The slave devices have internal registers with diverse functions. **Table 1** provides an overview of the Modbus functions.

There are three variants of the Modbus protocol:

- Modbus RTU: Binary transmission over RS485 (EIA485).
- Modbus ASCII: Plain text transmission over RS485 (EIA485). This is less efficient than binary transmission, but human-readable. Commands can be sent using a simple terminal utility.
- Modbus TCP: In this variant the Modbus commands are transmitted using TCP/IP. This is usually over Ethernet, but with this module it is over WLAN.

For more detailed information and protocol descriptions, visit the Modbus website [2].

Preparing the Modbus WLAN Module

In order to use the WLAN module with the Modbus protocol, you first have to load the required firmware. For this, remove the NodeMCU board from the Modbus board and connect it to a USB port on your computer. Then open the Arduino IDE, configured as described in the first part of this article. You can download the firmware, which turns the module into a Modbus client, from the Elektor project page [3].

Open the downloaded file *OpenPLC_ESP8266_1_0_MUX_V1_1.ino* in the Arduino IDE (*File -> Open*). The Arduino IDE will ask whether the project should be moved to the sketchbook. Answer this with 'Yes'. If the IDE creates a new folder, copy the header file (*modbus.h*) to the new folder containing the *ino* file, so the compiler will be able to find the header.

To enable access to the module over WLAN, the WLAN access data must be modified at the start of the source code (see **Listing 1**). There you should enter the access data of your own WLAN.

Table 1: Overview of Modbus functions.

Description	Mode	Bits
Individual "Coil" input/output	Read/write	1
Individual "Discrete Inputs"	Read only	1
"Input Registers" (analog/digital)	Read only	16
"Holding Registers" inputs/outputs (analog/digital)	Read/write	16

After making this modification, you can load the program into the Modbus module as described in the first part. The module works with DHCP, which means it will be assigned an IP address automatically by your router. You can check this in the router firmware. Alternatively, you can view the program output on the Arduino IDE serial monitor (opened by the looking glass icon at the top right). In the dropdown box at the bottom right, set the interface data rate to 115,200 baud.

Listing 2 shows an example of how to output the IP address to the serial monitor. To avoid having to repeat this procedure every time you power up the Modbus module, you should assign a static IP address. All recent routers allow this option. In any case, you should note the module IP address output by the program so that you can access the module over WLAN. If you do not see any output, press the Reset button on the NodeMCU board to restart the software, since the IP address is only issued during the start-up process.

First Test

To check whether the Modbus WLAN module is working properly, you can use the EasyModbus tool, which can be downloaded from SourceForge [4]. EasyModbus provides a server, a client, and a library. For this test you only need the client, which can be downloaded directly at [5]. In order to use the tool, you must have a Java version installed on your PC. If Java is not yet installed, you can use the free OpenJDK [6], which is very popular in the developer and maker community and is a standard component of all Linux distributions. In Ubuntu, for example, you can install it with the terminal command `sudo apt install openjdk-11-jdk`.

To run the EasyModbus client, enter the command `java -jar EasyModbusJavaClient.jar`. **Figure 1** shows the output of the



Listing 1: Modbus module WLAN configuration.

```

/*****NETWORK CONFIGURATION*****/

const char *ssid = "<YOUR_SSID>";
const char *password = "<YOUR_PASSWORD>";

/*****/

```



Listing 2: Modbus WLAN module IP address output.

```

Connecting to Vodafone-3980
.....
WiFi connected
Server started
My IP: 192.168.0.85

```

EasyModbus client (on the left under Linux; on the right under Windows). Here you should select *Modbus TCP* and enter the IP address of your module. Leave the *Starting Address* set to 1 and change the *Number of Values* to 4. If you now click *Read Discrete Inputs - FC2*, the four digital inputs of the module will be read and displayed. You should apply a signal to at least one of the inputs, so that you can see whether everything is working as expected.

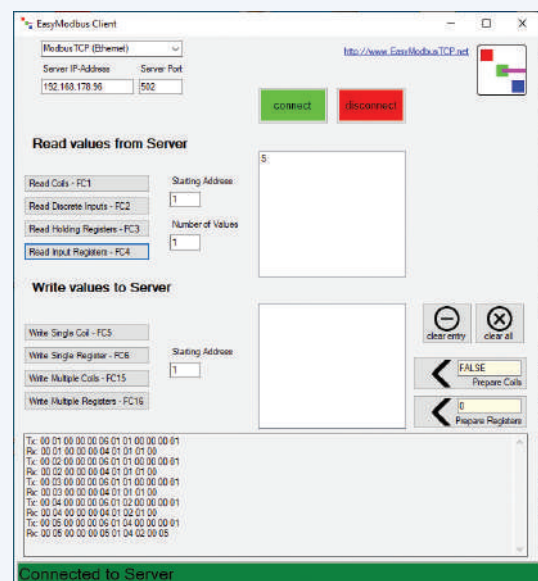
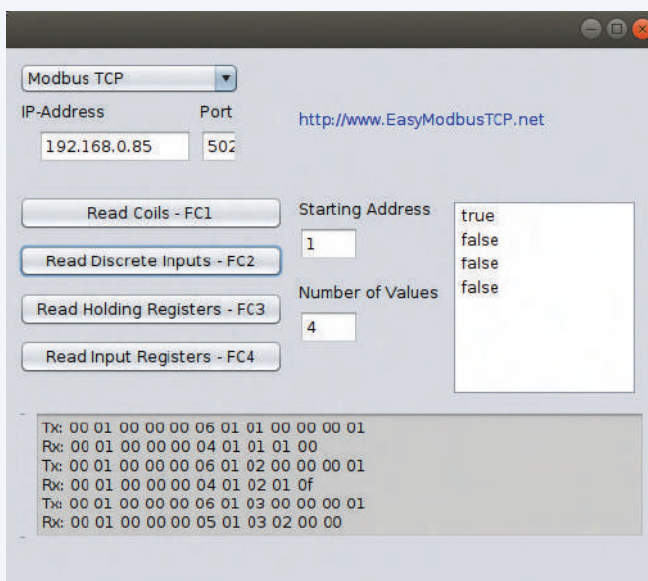


Figure 1: The EasyModbus client reading data from the module. The Linux version is shown on the left, and the Windows version on the right.



Listing 3: Example program `tor.py` for door control.

```

from pyModbusTCP.client import ModbusClient
client = ModbusClient(host="192.168.0.85",
    port=502, auto_open=True, debug=False)
while(True):
    inputs=client.read_discrete_inputs(0,4)
    end_switch_top    = inputs[0]
    end_switch_bottom = inputs[1]
    push_button_down  = inputs[2]
    push_button_up    = inputs[3]
    motor_up          = 0
    motor_down        = 1
    # request end_switch
    if(end_switch_top):
        client.write_single_coil(motor_up,False)
        print("gate open")
    if(end_switch_bottom):
        client.write_single_coil(motor_down,False)
        print("gate closed")
    # request push button
    if (push_button_up and not end_switch_top):
        client.write_single_coil(motor_up,True)
    if (push_button_down and not
        end-switch-bottom):
        client.write_single_coil(motor_down,True)

```

The EasyModbus client can only read data from Modbus; it cannot write data. To test writing, you can use the example program described below. The data actually transmitted can be seen at the bottom of the EasyModbus client window. This function is very helpful for debugging. If the first test was successful, you can now proceed to programming the module with Python.

Installing the Modbus +Library

To make the test more realistic, here we are using the lift door model from the first part [7]. We wrote a program for the PC that receives the positions of the buttons and the limit switches over Modbus in order to transmit corresponding motor actions (again over Modbus).

Our Python program needs a library [8] in order to access the Modbus. This is nothing unusual; there are Modbus libraries for virtually all programming languages. If necessary, you can even generate the bit sequences of the commands yourself and send them over the network. However, we don't want to go into that here.

First, you should make sure that the Python package installer *pip* is on board, which should always be the case with fairly recent Python installations. You can check this with either `$ python -m pip -version` under Linux or `C:\> py -m pip -version` under Windows.

If the *pip* installer is not present, you can install it via Python with

`get-pip.py` [9] (don't use the Linux package manager for this):

```

wget https://bootstrap.pypa.io/get-pip.py
python get-pip.py
get-pip.py

```

Then install the Modbus library:

```

$ python -m pip install -U pymodbusTCP
C:\> py -m pip install -U pymodbusTCP

```

You can find extensive documentation for the library at [10], along with many code examples.

Example Program

Let's have a detailed look at the example program in **Listing 3**. The first line imports the Modbus library. The next line creates a new connection object in order to communicate with a specific Modbus device. Several parameters can be passed for this. In our case we pass the IP address of the target device as `host`. Port 502 is the default port for Modbus TCP communication. The parameter `auto_open` determines whether the connection should be set up automatically or manually. In our example we chose automatic for the sake of convenience, but if you want to have complete control over connection setup, you should choose `False`. With manual connection setup you have more options for control and for error handling. If the parameter `debug` is set to `True`, the software will output all transmitted data to the console. This is very helpful when you are looking for a bug.

Using a `while` loop, we repeatedly run through the program as described in the first part. This means the program works cyclically, just like all industrial controllers. The code inside the loop is the main part of the program, which first reads the inputs and then converts the input values into meaningful variables. We also define variables for the two digital outputs. This makes the following program lines easier to read.

There you can see that the motor is switched off when the end points are reached. A text message is output to the console so that you can see the current door position on the computer. Last but not least, the motor is run in the respective direction when the Up or Down button is pressed. The logical AND between the limit switches prevents the relays from chattering when the door is already at an end point.

To run the program, enter the command line `python tor.py`. As usual, you can stop the program with `Ctrl-C`.

AdvancedHMI


Convenient remote control with visual feedback that you can adapt to your own purposes can be realised with a variety of PC frameworks. The program AdvancedHMI is open-source software for creating human-machine interface (HMI) applications that communicate with your PLC or other I/O devices. This software differs from other standard packages in that it allows you to create executable files instead of just configurations that are interpreted by a runtime engine. This results in very fast and efficient applications.

AdvancedHMI [11] is based on the Microsoft .NET framework. Applications are generated in *Microsoft Visual Studio Community 2019*, which is available free of charge. This allows you to create basic HMIs by drag & drop without having to write any code. The .NET framework is used by a large number of developers and has a wealth of support networks. You can find much more support for AdvancedHMI than for all other standard HMI packages combined.

The author used this platform to generate a user interface to control the baseboard. It shows the status of the inputs and provides buttons that can be used to switch the outputs (**Figure 2**).

.NET programs, such as this user interface to control the baseboard, can also run under Linux with Mono. It has been tested successfully on a Raspberry Pi.

Get Control

The Modbus WLAN module makes it easy to control Modbus-compliant devices from a PC or a smartphone. This module is also a good learning project if you want to get the hang of the Modbus protocol. The hardware design, however, is so robust and error-tolerant that you can also use it in your own ESP8266 projects if you want to control and read industrial devices such as solenoid valves, motors, and other kinds of sensors or actuators. 

200507-B-01

Questions or Comments?

Do you have any technical questions or comments? If so, please contact the author at josef@bernhardt.de or the editorial staff at editor@elektor.com.

Contributors

Idea, Design and Text: **Josef Bernhardt, Martin Mohr**
 Editor: **Rolf Gerstendorf**
 Layout: **Giel Dols**



Figure 2: The AdvancedHMI software generates human-machine interface applications.



RELATED PRODUCT

> **NodeMCU Module (SKU 17952)**
www.elektor.com/17952



WEB LINKS

- [1] Modbus over WLAN (Part 1), ElektorMag 9-10/2021: <http://www.elektormagazine.de/200507-01>
- [2] Modbus website: <https://modbus.org/>
- [3] Elektor project page: <https://www.elektormagazine.de/200507-B-01>
- [4] EasyModbus website: <https://bit.ly/2QDIDJz>
- [5] EasyModbus client download: <https://bit.ly/3d2zzFp>
- [6] OpenJDK download: <https://bit.ly/2OVK4m6>
- [7] Lift door model on YouTube: <https://youtu.be/VHIBQswdA0E>
- [8] Modbus library for Python: <https://pypi.org/project/pyModbusTCP/#description>
- [9] get-pip.py: <https://bootstrap.pypa.io/get-pip.py>
- [10] Modbus library examples: <https://pymodbusTCP.readthedocs.io/en/latest/>
- [11] AdvancedHMI: https://www.advancedhmi.com/index.php?main_page=page&id=14&chapter=0
- [12] More infos by the author: <https://bit.ly/3ch1PFI>

Understanding the Neurons in Neural Networks (Part 3)

Practical Neurons

By **Stuart Cording** (Elektor)

So far, we have created a neural network and developed an understanding of how it functions. We've even been able to teach it how the XOR function works, something that requires the classification of input patterns. In this article, we will examine how to implement a part of an autonomous driving system: recognizing the state of traffic lights.

There have been many predictions regarding self-driving cars [1], none of which have come to fruition. Much of the complexity arises from attempting to understanding the intent of other road users and pedestrians. However, a lot of the autonomous driving system is about classifying everything the many sensors and cameras 'see' around the vehicle. This ranges from traffic lights and road signs, to street or road

markings, and vehicle types and people. Now that we have a working neural network in the form of our multilayer perceptron, we will use it to detect a traffic light's colors.

One of the great things about the Processing environment we have used thus far is the ease with which it can access webcams. The examples used here will run on just about any laptop or PC running Windows, Linux

or macOS with an integrated or external camera.

Finding Your Camera

Before we can get started, a new library needs to be added to Processing that provides access to any attached webcams. From the menu, select *Sketch -> Import Library... -> Add Library...* (**Figure 1**), which opens the window shown in **Figure 2**. Ensuring that

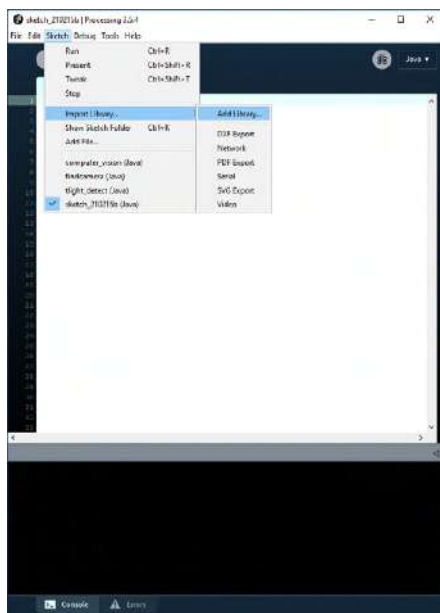


Figure 1: Adding a new library to Processing.

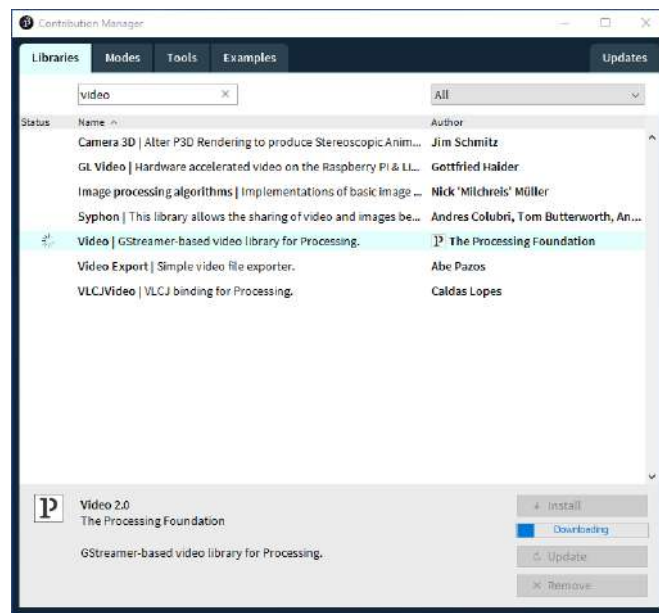


Figure 2: Searching for and installing the Video library.

the *Libraries* tab is selected, enter “video” into the search box. From the list that appears we select *Video | GStreamer-based video library for Processing* and then click on *Install*. As before, we will be using the code from the GitHub repository prepared for this series of articles [2].

With the library installed, we can run the project */trafficlight/findcamera/findcamera.pde*. This code simply requests the list of available cameras connected to the computer and allows the user to select one by entering a number between 0 and 9. Make sure the small *findcamera* window has the context (click inside the window) before pressing a number. If the Processing IDE window has context, you will simply end up typing a number somewhere in the source code.

Once a camera source has been selected, its output is shown in a low-resolution 320×240-pixel output in the window (**Figure 3**). The text console will also provide the source code needed to select your camera for the next two projects we will look at (**Figure 4**).

Finally, we need a traffic light. Since you are unlikely to have one to hand, one has been prepared in *trafficlight/resources* in various file formats. Simply print one out and keep it to hand.

When stopping Processing projects that use the camera, you may see the message “WARNING: no real random source present!”. This seems to be a bug related to using the

Video library but has no impact on the code’s functionality.

How Do Cameras See?

Perhaps one of the most important aspects of using neural networks is to develop an understanding of how a computer can interpret the incoming data. In the case of camera input, the computer image is displayed by mixing the three primary colors, red, green, and blue. This is commonly termed the RGB format. The three values vary between 0 and 255 (or 0x00 and 0xFF in hexadecimal). If we point the camera at something blue, we would expect the B value to be relatively high and the other values quite low. Point it at something yellow, and then there should

be a high R and G value as, together, red and green make yellow. To get a better feel for additive color mixing, the project */trafficlights/additive/additive.pde* may be of interest (**Figure 5**).

With this knowledge, it would seem to make sense to determine how our camera “sees” the colors of our traffic light and note the RGB values it reports. We can then teach our neural network the three traffic light colors so that it may classify them as red, amber, and green.

Determining RGB Values

The project *trafficlight/computer_vision/computer_vision.pde* is the next project we will need. This displays the selected

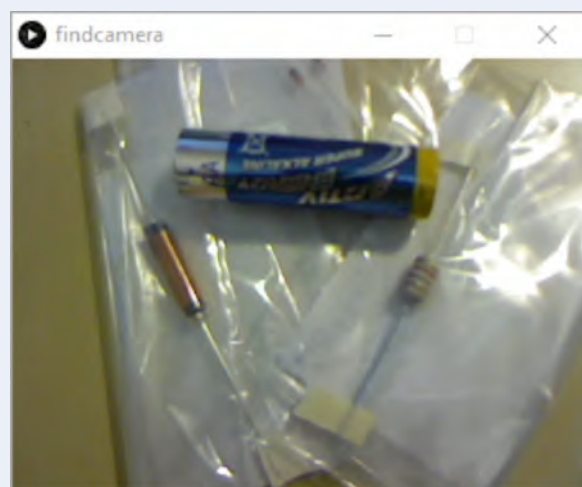


Figure 3: Example output from *findcamera.pde*.

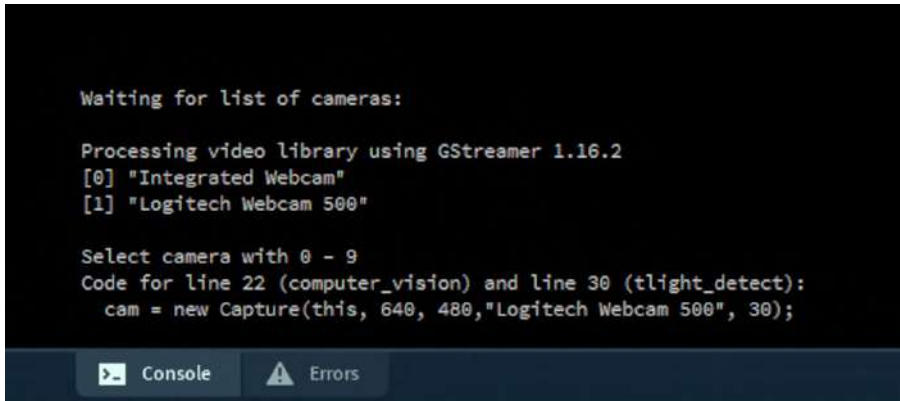


Figure 4: The console output of findcamera.pde provides the initialization code for the other projects to use the desired camera.

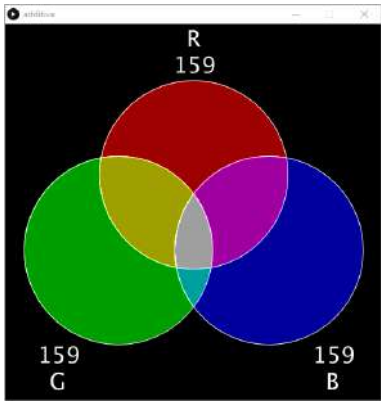


Figure 5: additive.pde demonstrates additive color mixing.

camera's output next to the RGB values it is reporting. Using this project, we can record the RGB values the camera sees. Before starting the code, don't forget to paste the line of code you determined in findcamera.pde to line 22 in this project. This ensures your chosen camera is used.

While pointing the camera at the red traffic light (keep the 'light' approximately within

Table 1: RGB values captured for the three traffic light colors.

Light color	R	G	B
Red	220	56	8
Amber	216	130	11
Green	123	150	128

the dashed circle), note the color seen (top) and the RGB values that define it (Figure 6). The RGB values are actually the average of all the pixels the camera captures in the center's red bounding square. Despite being an average, the RGB values will change up and down rapidly. To capture a single value, press 'p' on your keyboard, and the code pauses on a single set of values. Be sure to click inside this window before pressing 'p' to give it the context. Pressing 's' restarts the camera capture and RGB generation again.

The traffic light printout used here was laminated so there are some reflections in places. If you have printed your traffic light image out using a laser printer, you may also have a slightly reflective surface. As a result, although the camera is pointing at the red traffic light, the "Color seen" may be pinkish or even close to white. This is, obviously, not much use for the neural network. It needs to know the color 'red traffic light' under optimal conditions, so move the camera or change the lighting until you feel you have achieved an RGB value representing the color optimally.

This also raises another issue concerning accuracy. Drivers will know how difficult it is to discern the active traffic light lamp when the sun is shining into our eyes or directly into the lamps. If we humans cannot distinguish which lamp is lit, how can a neural network figure it out? The answer is, it can't. If we need a more robust algorithm, we need to train it with 'poor visibility' data. We may also need to provide another input, such as where the sun is shining from, so that the neural network knows when there is 'poor visibility' and that it should use the 'poor visibility' data

set. Finally, we could improve the incoming data from the camera, perhaps by adding an infrared image of the traffic light (which lamps are hot) or some other clever filtering. However, don't despair! We shall add some robustness into our training data to handle some variation in the color of our traffic light input.

The critical task at this point is to collect the RGB data for the red, amber, and green traffic lights using your camera and your lighting conditions. The data collected using the author's setup is shown in Table 1.

Detecting Traffic Lights

Armed with our data, we can now train the neural network the colors of our traffic light. For this final stage we will need the project traffilight/tlight_detect/tlight_detect.pde open in Processing. Start by ensuring that the correct camera initialization code is pasted into line 30 (from find_camera.pde).

This project uses a three-input, six-hidden, and four-output (3/6/4) node MLP. The three inputs are for the three colors, R, G, and B. Three of the four outputs are to classify the 'red,' 'amber,' and 'green' of the traffic light. The fourth will be used later. The use of six hidden nodes was chosen arbitrarily as 'enough' to handle the classification task. We will see that this configuration does work and, as before, readers are encouraged to experiment with fewer or more hidden nodes.

If you run the code 'as is,' the neural network operates without any learning. The results (Figure 7) show that any color is classified as all the colors the network is defined to detect.



Figure 6: computer_vision.pde provides us with the RGB values the camera 'sees' for each traffic-light color.



Figure 7: The response of tlight_detect.pde prior to learning any colors.

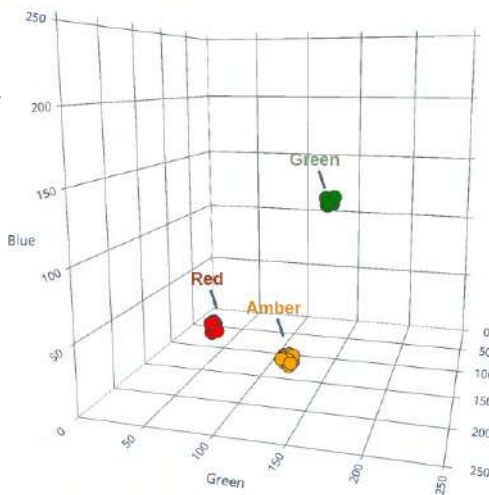


Figure 8: The three traffic light colors in RGB color space after randomization.

The teaching of the neural network takes place around line 51. Uncomment the first three methods and add the RGB values you acquired earlier. The three methods used to define red, amber, and green are:

```
teachRed(159, 65, 37);
teachAmber(213, 141, 40);
teachGreen(128, 152, 130);
```

Each time these functions are called, the network is trained to classify that RGB combination as the associated color (**Figure 8**). To allow for some variation in lighting and automatic changes in exposure setting by the camera, a small amount of variation (± 4) to the RGB values is applied using a `randomise()` function (line 336). Again, you can experiment with the effectiveness of this approach and the amount of randomization applied.

Training of the network is swift compared to previous projects as we are no longer writing the error values during learning to a file. Simply run the project and, after a few seconds, the neural network will start evaluating the color in the red bounding square of the camera window (**Figure 9**).

The traffic light in the view of the camera is determined on line 156 onwards. The RGB colors captured are applied to the inputs of the MLP and the network's output is calculated:

```
network.setInputNode(0, (float) r
/ 255.0);
network.setInputNode(1, (float) g
/ 255.0);
network.setInputNode(2, (float) b
/ 255.0);
network.calculateOutput();
```

The decision on the color seen is then made starting at line 171. The output of each output node is evaluated. If classification certainty is above 90% (0.90), the color seen is displayed in a circle along with the classifier 'Red,' 'Amber,' or 'Green'.

```
// If likelihood of 'Red' > 90%...
if (network.getOutputNode(0) >
0.90) {
    fill(200, 200, 200);
    // ...write "Red"...
    text("Red", 640+(100), 320);
    // ... and set color to the
    color seen.
    fill(r, g, b);
} else {
    // Otherwise, set color to
    black
    fill(0, 0, 0);
}
// Now draw the color seen in a
circle
ellipse(640+(50), 300, 40, 40);
```

You can experiment with the accuracy of the MLP by pointing the camera at the different traffic light at different angles and under different lighting conditions. Also, try pointing the camera at objects in your vicinity that, in your opinion, approximate the colors the MLP has learnt.

One thing you may notice is the misclassification of a wide range of colors as a known color. In the author's example, the 'green' classification is also given for the traffic light surround, frame, and overall image background (**Figure 10**).

Tightening-Up Neural Network Classification

From the RGB values displayed, it is clear



Figure 9: tight_detect.pde having learned red, amber, and green.

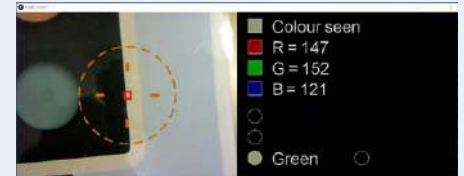


Figure 10: Misclassification of other colors as 'green.'

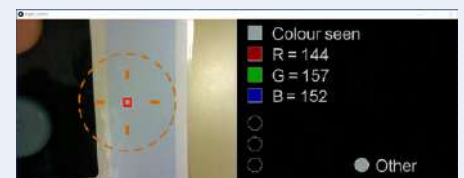


Figure 11: Misclassification of other colors as 'green' resolved using 'other' output node.

Table 2: RGB values captured for the 'unwanted' colors.

Object	R	G	B
Dark traffic light surround	76	72	35
White frame	175	167	138
Blue background	152	167	161

that the colors passed to the MLP are reasonably close to that of the wanted classification for 'Green.' There are several ways to tighten up the classification. The first would be to raise the 90% bar for accurate classification to a higher value. Another may be to increase the number of epochs of learning. The other is to rethink the classification implementation.

So far, we have focused on what we want to positively classify. However, sometimes it can help to teach the neural network what does not belong to the patterns we are looking for. Essentially we can say, "these are three things we are looking for, but here are some similar things we are definitely not looking for." This is where our fourth output node comes into play.

Returning to the *computer_vision.pde* project once more, we can capture the RGB values for colors that we want to be classified as 'other.' As an example, RGB values for the dark traffic light surround, the white border, and the blue background were collected, with the results as shown in **Table 2**.

Back in *tlight_detect.pde*, these values can be taught as 'Other' unwanted colors using the `teachOther()` method. Simply uncomment the code around line 60 as follows and add

your RGB values:

```
teachOther(76, 72, 35);
teachOther(175, 167, 138);
teachOther(152, 167, 161);
```

Rerunning the project results in a noticeable improvement. The area around the traffic lights (surround, frame, background) is classified as 'Other' instead of 'Green' (**Figure 11**).

For Next Time

In this article, we've now seen a neural network resolve a real-world classification problem. We've also learned that it can help to teach both the desired classification and the undesired classification data.

Why not try building upon this example to explore the following?

- How 'robust' can you make the MLP to camera angle and changes in exposure? Is it better to more heavily randomize the learning data or raise the bar for classification (> 90%)?
- Does accuracy improve if you increase the number of 'Other' output nodes and teach each one an unwanted color?
- What impact does reducing or increasing the number of hidden nodes have

on the system?

- Would a fourth input for 'general brightness' help to increase recognition accuracy under varying lighting conditions?

While it is great to run neural networks on powerful laptops and PCs, many of us will desire to have such a capability on the smaller processors used on boards such as Arduino. In our final article of this series, we will use an RGB sensor and an Arduino to implement a color-detecting neural network. Perhaps it will form the basis for your next brainy Arduino project! **◀**

210160-C-01

Questions or Comments?

Do you have technical questions or comments regarding this article? Then email the author at stuart.cording@elektor.com.

Contributors

Idea, Text and Images: **Stuart Cording**
 Editors: **Jens Nickel, C. J. Abate**
 Illustrations: **Patrick Wielders**
 Layout: **Harmen Heida**



RELATED PRODUCTS

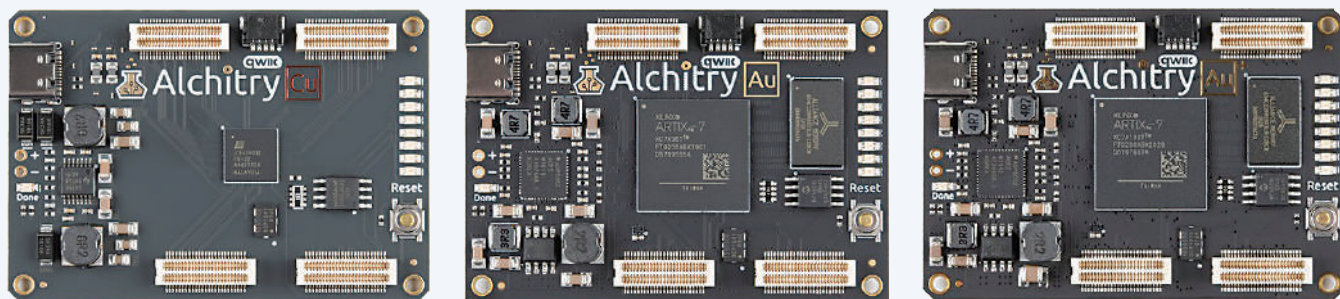
- **B. van Dam, *Artificial Intelligence* (E-book) (SKU 18090)**
www.elektor.com/18090
- **Google AIY Vision Kit for Raspberry Pi (SKU 19365)**
www.elektor.com/19365
- **HuskyLens AI Camera with Silicone Case (SKU 19248)**
www.elektor.com/19248

WEB LINKS

- [1] M. Anderson, "Surprise! 2020 Is Not the Year for Self-Driving Cars," IEEE Spectrum, April 2020: <http://bit.ly/2ZSwm5f>
- [2] GitHub repository - simple-neural-network: <https://bit.ly/2ZHLv9p>
- [3] H. Zhang et al., "StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks," CVF, December 2017: <https://bit.ly/3qZccCs>

Inside an Open-Source Processor

Sample Chapter: Lattice and Xilinx FPGA Results



Alchitry FPGA development boards mentioned in this article. Left to right: Cu — Au — Au+.

By **Monte Dalrymple** (USA)

This instalment of Elektor Books presents material excerpted from Monte Dalrymple's book *Inside an Open-Source Processor* published by Elektor. FPGA technology in combination with (V)HDL is hot among electronicists and RISC-V adds the open-source path to professional applications. In this article, we examine some side-by-side comparisons and results obtained from an example application running on both Lattice and Xilinx FPGA boards of the 'Alchitry' series now available from Elektor. But first, a few tips!

Editor's Note. This article is an excerpt from the book *Inside an Open-Source Processor* — an Introduction to RISC-V, formatted and lightly edited to match Elektor Magazine editorial standards and page layout. Being an extract from a larger publication, some terms in this article may refer to discussions elsewhere in the original. The Author and Editor have done their best to preclude such instances and are happy to help with queries. Contact details are in the **Questions or Comments?** box.

In the past, setting up a project in an FPGA tool flow was a daunting task, but as the technology has matured this setup has gotten much simpler. In the case of the example here (a 24-hour clock

described elsewhere in the book) most of the work has already been done.

Just be aware that large projects, or projects taking maximum advantage of the capabilities of the FPGA, will inevitably be much more complex.

In the case of both Lattice and Xilinx, besides selecting the appropriate FPGA target for the project only two or three files will need to be linked in the tools. All of the project files need to be in the same directory in this simplified approach. As mentioned previously, readers are strongly encouraged to review the Alchitry tutorials for FPGA

software installation and project setup because only the essential details will be covered here.

Lattice (Alchrity Cu) Tips

The FPGA used on this board requires the Lattice iCEcube2 software. This is a mature product without a lot of bells and whistles, but there are a few things to remember:

1. Be sure to set up the project with the correct device options. The Alchrity Cu uses the HX8K device from the iCE40 family, in the CB132 package. All of the I/O banks use a 3.3 V supply.
2. Only the `yrv_alchrity.v` file needs to be entered as a design file, because all of the other design files will be automatically included in the correct order.
3. For the synthesis tool, be sure to select the Lattice LSE Synthesis. No constraint files are necessary during logic synthesis.
4. After the synthesis completes, the `yrv_alchrity.pcf` pin constraint file and `timing.sdc` timing constraint file need to be linked before running the remainder of the tools.

There will be a number of warnings in the various log files but none of them should be important enough to require any action.

Xilinx (Alchrity Au and Au+) Tips

The FPGAs used on these boards require the Xilinx Vivado software. This is a very complex product, and this project will only use a small subset of the capabilities of this software. Here are a few things to remember:

1. Only the `yrv_alchrity.v` file needs to be entered as a design file.
2. Only the `yrv_alchrity.xdc` constraint file is required. This file contains the pin constraints, timing constraints as well as the programming voltage specifications.
3. The correct device option is the xc7a35tftg256-1 for the Au board and the xc7a100tftg256-1 for the Au+ board.

There are even more warnings in the log files for these devices, but again, none of them should be important enough to require action.

FPGA Results

This example project has been implemented on all three target FPGA development boards and the individual results will be covered in the following sections. But at this point it is interesting to do a side-by-side comparison of some of the results. **Table 1** shows the reported clock speed for the 100 MHz divider.

The 100 MHz divider was designed for maximum speed with only one level of logic between flip-flops. This means that this result should represent the capabilities of the technology.

The reported speed for the Lattice device, if true, is impressive. What is most interesting is that the reported speed gets better, and presumably more accurate, as the implementation goes from

Table 1: FPGA Results.

100 MHz Divider	Alchrity Cu	Alchrity Au	Alchrity Au+
Speed (at synthesis)	313.9 MHz	114.1 MHz	114.2 MHz
Speed (at placement)	542.3 MHz	-	-
Speed (final)	625.4 MHz	111.9 MHz	111.9 MHz

Table 2: FPGA Results.

FPGA Attribute	Alchrity Cu	Alchrity Au	Alchrity Au+
Logic elements	7,680	20,800	63,400
LEs Used	4,646	2,189	2,185
LEs Used (%)	60.6%	10.5%	3.5 %
Flip-flops	1331	1377	1377
Speed (at synthesis)	11.8 MHz	28.1 MHz	27.7 MHz
Speed (at placement)	25.8 MHz	-	-
Speed (final)	30.8 MHz	34.4 MHz	34.2 MHz

logic synthesis to logic placement to the final routed result. While it is best to have the final performance be better than the initial estimate, being off by a factor of two is not ideal because it might lead to abandoning a project as unfeasible erroneously.

The Xilinx tools do not give a separate speed estimate at logic placement, but the reported speed is nearly identical between logic synthesis and the final result. From the point of view of a designer, this is preferred.

Table 2 shows the resources required for the project as well as the maximum clock frequency for the CPU itself. These results do not include any of the 64-bit counters that are part of the RISC-V Privileged Architecture.

The second line of this table demonstrates why comparing FPGA results can be complicated. From the table, it appears that the Xilinx implementation requires less than half of the resources required by the Lattice implementation. But a Xilinx logic element contains significantly more logic than a Lattice logic element. In addition, the Xilinx design uses the dedicated DSP blocks which the Lattice device does not have.



Listing 1: Lattice Report Excerpt.

Total Logic Cells: 4646/7680

Combinational Logic Cells: 3315 out of 7680 43.1641%

Sequential Logic Cells: 1331 out of 7680 17.3307%

Logic Tiles: 847 out of 960 88.2292%

Registers:

Logic Registers: 1331 out of 7680 17.3307%

IO Registers: 0 out of 1280 0

Block RAMs: 12 out of 32 37.5%

Warm Boots: 0 out of 1 0%

Pins:

Input Pins: 36 out of 95 37.8947%

Output Pins: 50 out of 95 52.6316%

InOut Pins: 0 out of 95 0%

Global Buffers: 2 out of 8 25%

PLLs: 0 out of 2 0%

Looking at the percentage of logic elements used reveals the size disparity between the three FPGAs here. A little over half of the Lattice device is required for the `yrv_mcu` design, but roughly one-tenth of the mid-range Xilinx device is used and less than one-twentieth of the large Xilinx is required. This usage is in line with the cost of the different development boards.

As expected, the number of flip-flops required for the design is similar across the three FPGAs, and the exact count is more a function of the logic synthesis tool than anything else. Logic synthesis tools will replicate flip-flops to increase performance or to simplify routing.

The speed rows show that all three FPGAs provide roughly the same performance, which is somewhat surprising given that the Lattice FPGA does not contain the dedicated DSP blocks that are used in the Xilinx implementation.

Alchrity Cu Details

Listing 1 is extracted directly from the Lattice implementation report, showing the details of the resource utilization. **Figure 1** is taken from a screenshot of the Lattice floorplanner tool, showing the usage of logic elements as distributed across the device. This distribution is fairly even across the entire device even though the overall usage is only about 60 percent.

Alchrity Au Details

Listing 2 is extracted directly from the Xilinx implementation report, showing details of the resource utilization. The full Xilinx

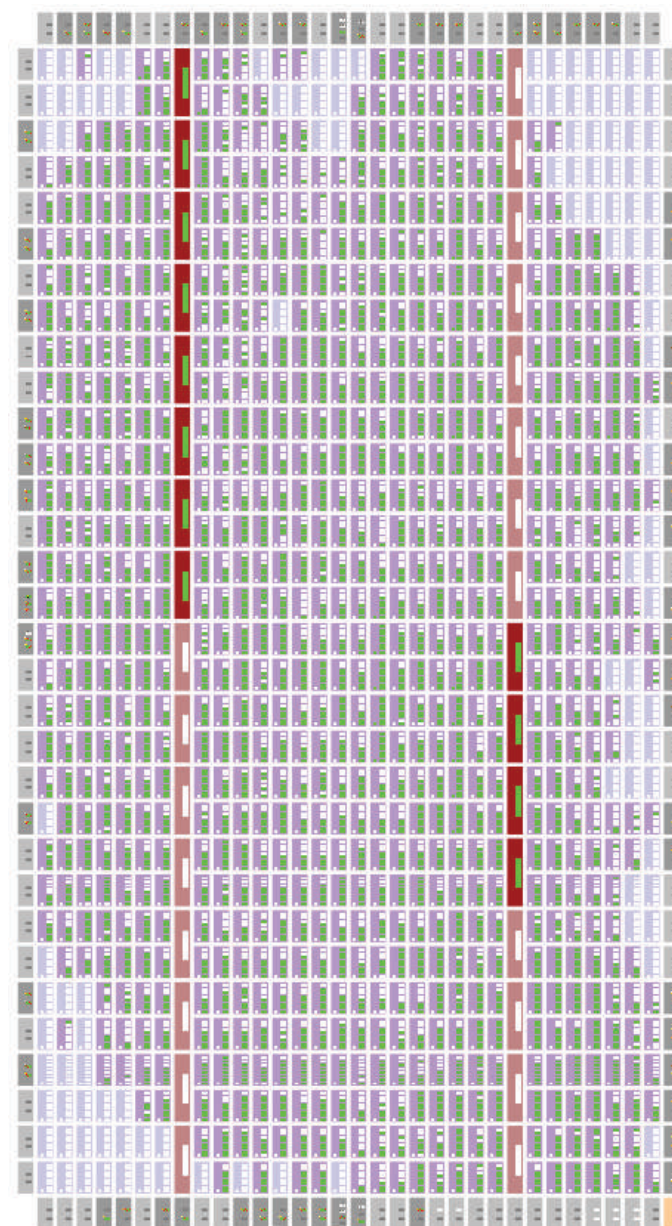


Figure 1: Lattice Floorplanner View (Alchrity Cu).

report is much larger, covering all of the dedicated hardware available in the FPGA.

Figure 2 is taken from a screenshot of the Xilinx floorplanner tool, showing the usage of logic elements as distributed across the device, along with how the dedicated logic blocks are distributed throughout the device. It seems that placement tools always start in the lower-left corner of a device. It is not clear why the logic seems to be split into two pieces.



Listing 2: Xilinx XCA35T Report Excerpt.

1. Slice Logic

Site Type	Used	Fixed	Available	Util%
Slice LUTs	2189	0	20800	10.52
LUT as Logic	2189	0	20800	10.52
LUT as Memory	0	0	9600	0.00
Slice Registers	1377	0	41600	3.31
Register as Flip Flop	1377	0	41600	3.31
Register as Latch	0	0	41600	0.00
F7 Muxes	3	0	16300	0.02
F8 Muxes	0	0	8150	0.00

Alchrity Au+ Details

Listing 3 is also extracted directly from the Xilinx implementation report, showing details of the resource utilization. Notice that the numbers for resources used are nearly identical between the two Xilinx devices.

Figure 3 is also taken from a screenshot of the Xilinx floorplanner tool. What is interesting here is that even though this FPGA contains three times the number of logic elements, the overall area required by the example here seems to be spread out over nearly the same area as in the first Xilinx implementation.

Hardware Programming

Downloading the FPGA bitstream to an Alchrity development board is simple when using the Alchrity Loader program. This stand-alone program is automatically installed as part of the Alchrity

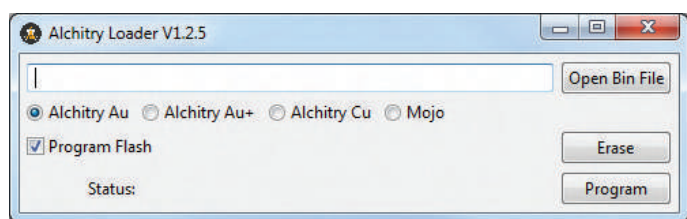


Figure 4: Alchrity Loader View.

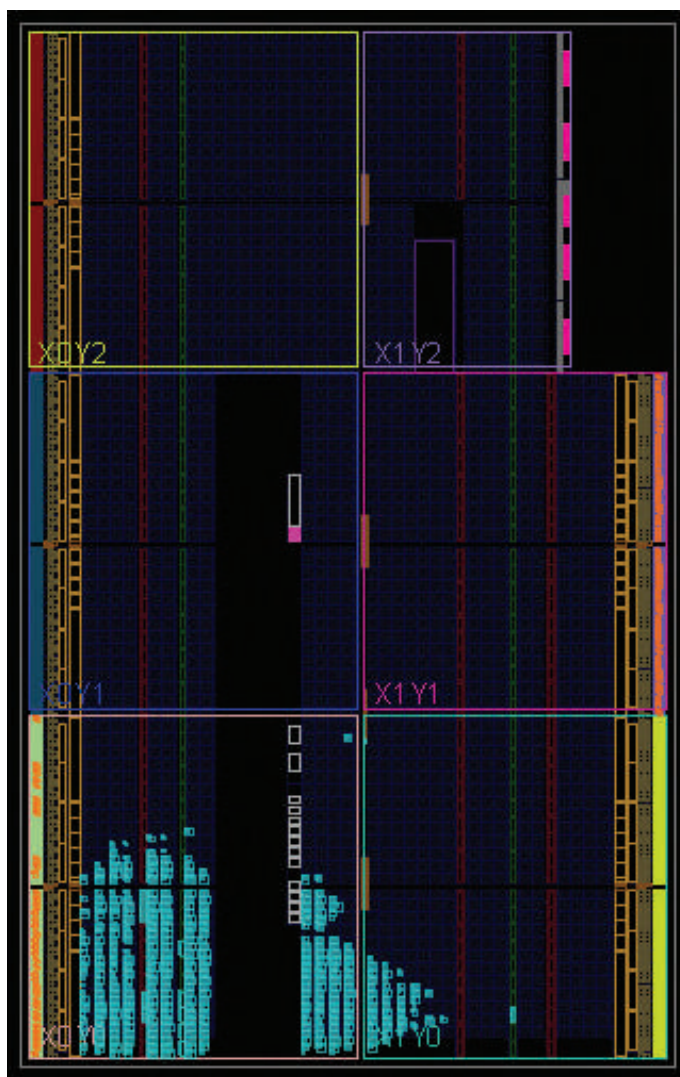


Figure 2: Xilinx XCA35T Floorplanner View (Alchrity Au).

Labs software available from Alchrity. **Figure 4** shows the user interface for this program.

Programming the Flash memory device on the development board that is used to load the FPGA is as simple as specifying the bitstream file, selecting the target board, and clicking the **Program** button.

By default, the program looks for a bitstream file of type **.bin**, which is the default used by the iCEcube2 software. The Vivado software generates a bitstream file of type **.bit**, so keep that in mind when loading the Xilinx bitstream file. ◀

210347-01

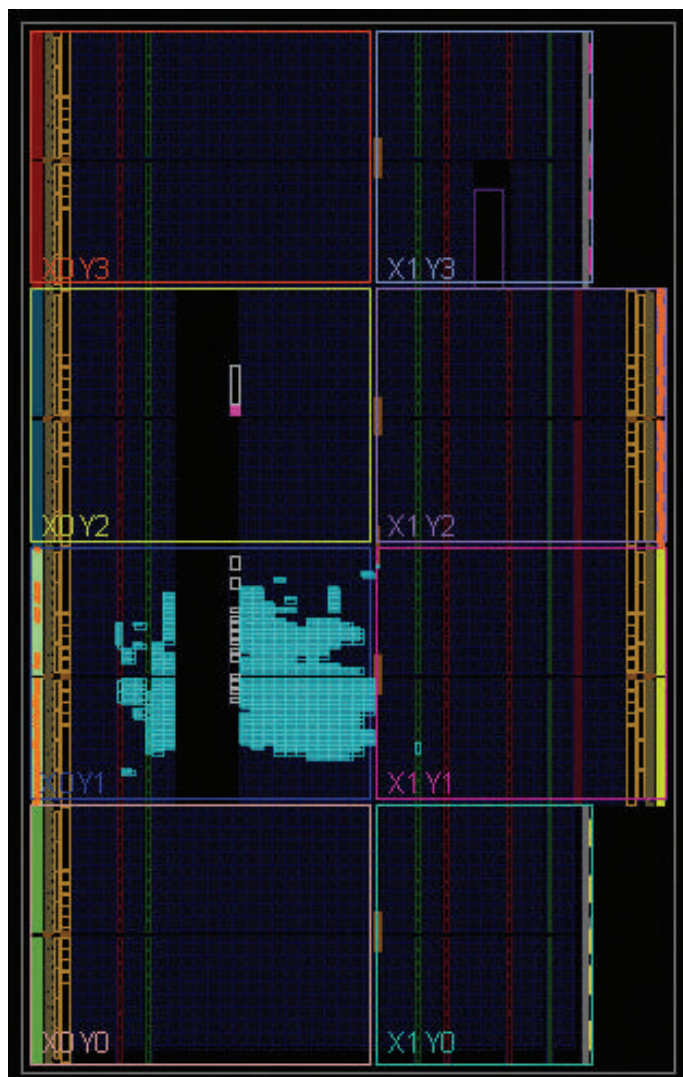


Figure 3: Xilinx XCA100T Floorplanner View (Alchitry Au+).

Questions or Comments?

Do you have any technical questions or comments related to this article? Then email the author at monted@systemyde.com or Elektor at editor@elektor.com.

Contributors

Text: Monte Dalrymple
Editor: Jan Buiting
Layout: Giel Dols



Listing 3: Xilinx XCA100T Report Extract.

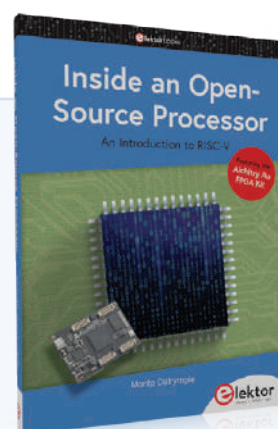
1. Slice Logic

Site Type	Used	Fixed	Available	Util%
Slice LUTs	2185	0	63400	3.45
LUT as Logic	2185	0	63400	3.45
LUT as Memory	0	0	19000	0.00
Slice Registers	1377	0	126800	1.09
Register as Flip Flop	1377	0	126800	1.09
Register as Latch	0	0	126800	0.00
F7 Muxes	3	0	31700	<0.01
F8 Muxes	0	0	15850	0.00



RELATED PRODUCTS

- > **Book:** M. Dalrymple, *Inside an Open-Source Processor* (Elektor 2021) (SKU 19826) www.elektor.com/19826
- > **E-Book:** M. Dalrymple, *Inside an Open-Source Processor* (Elektor 2021) (SKU 19827) www.elektor.com/19827
- > **Alchitry Au FPGA Kit** (SKU 19638) www.elektor.com/19638

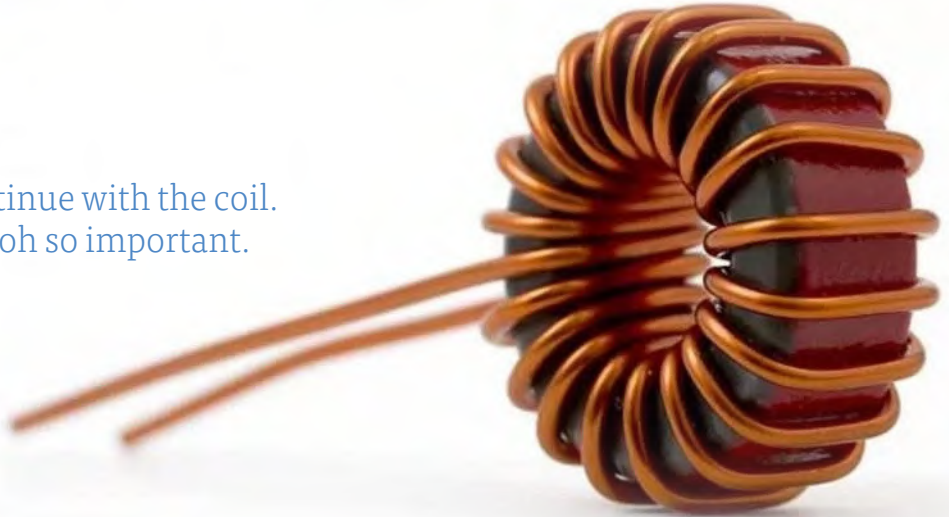


Starting Out in Electronics

We Are Not Yet Done with the Coil

By Eric Bogers (Elektor)

In this installment we continue with the coil. Unloved by hobbyists, but oh so important.



Series and Parallel Circuits

Resonant circuits (**Figure 1**) are the frequency-determining circuits in oscillators; at the so-called resonant frequency their impedance is at its minimum or maximum (that is, the value is at its most extreme). At this resonant frequency the impedance of the capacitor is equal to that of the inductor; the following equation holds:

$$X_C = \frac{1}{2 \cdot \pi \cdot f_R \cdot C} = X_L = 2 \cdot \pi \cdot f_R \cdot L$$

$$\Rightarrow f_R = \frac{1}{2 \cdot \pi \cdot \sqrt{L \cdot C}}$$

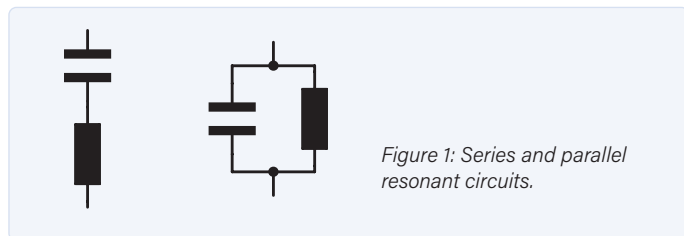


Figure 1: Series and parallel resonant circuits.

This formula is valid for both series and parallel resonant circuits. In **Figure 2** you can see the impedance characteristics as a function of frequency. A series circuit can be used in (among other things!) a loudspeaker box to attenuate a small frequency range in order to linearise the overall characteristic of the box. For this purpose, in most cases, a resistor has to be connected in series with this series-resonant circuit.

Crossover Filters

Figure 3 shows the impedance and phase characteristics of a bass speaker (which, incidentally, is mounted in a front-loaded horn enclosure, but there is no need to remember that detail). **Figure 4** shows the frequency characteristic of that system and **Figure 5** gives an impression of the amount of distortion that is present.

We see that we can divide the entire frequency range into four regions. The first region is below the resonance frequency — in

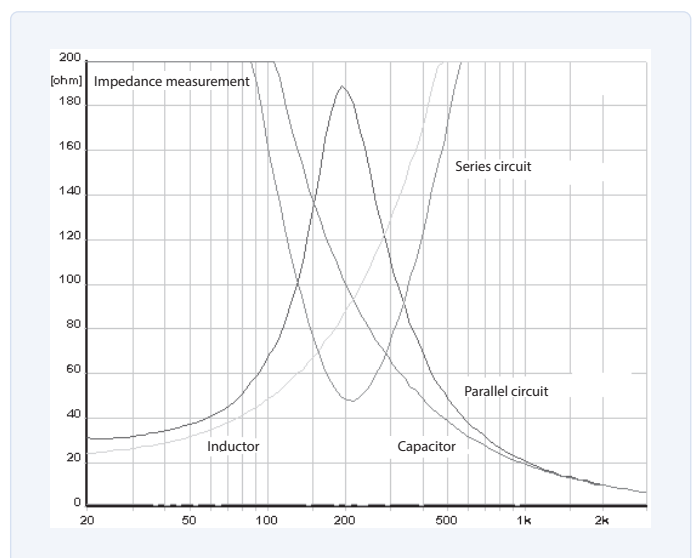
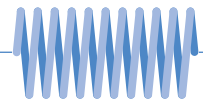


Figure 2: Impedance characteristics of series and parallel resonant circuits.



the example of this loudspeaker this region is from 0 Hz to about 30 Hz. The efficiency of the loudspeaker could be called fairly mediocre here, the amount of distortion is very large, while the voltage and the current are some 60° to 90° displaced in phase with respect to each other. This also means that the power loss in the amplifier is at its highest. For these reasons it is pointless to use this loudspeaker in this frequency range: in the first place, it doesn't sound good, in the second place, because of its poor efficiency, it requires way too much electrical power, and in the third place, the amplifier will overheat (with all the detrimental consequences that entails).

For this loudspeaker, a high-pass filter with a corner frequency of at least 50 Hz should be used. For the reproduction of lower frequencies we would need to use another type of loudspeaker and/or another type of enclosure.

The second region is around the resonant frequency; here the impedance (and therefore also the efficiency) increase dramatically. In most cases, loudspeaker boxes work quite well in this frequency range (as it happens, however, that is not the case in this example...).

In the third region the impedance has a reasonably horizontal (constant) characteristic and the efficiency is acceptable with a small distortion factor; the phase characteristic too, gives no reason for concern. In the example this region is from about 100 Hz to about 1 kHz — and this is the frequency range where this loudspeaker should be used. The distortion characteristic illustrates the relatively large second-order distortion (even harmonics) that are a characteristic of horn systems.

Finally, in the fourth region the impedance progressively increases as a consequence of the self-inductance of the voice coil, the output level clearly reduces and the distortion also increases again. This point is the upper frequency limit of the signal that may be applied to this loudspeaker. If with passive crossover filters (see later on) the corner frequency falls in this fourth region, then impedance compensation has to be used (this will be a topic in the next instalment).

Why Crossover Filters?

As we have seen in the example above, loudspeakers can only reproduce a limited frequency range well. That is why the audio spectrum has to be split into a number of regions, each of which is then applied to a loudspeaker that was designed specifically for the corresponding range. This is the purpose of the crossover filter, which ensures that each loudspeaker is served with the correct frequencies.

This can, in principle, be done in two different ways: the signal can be split after the power amplifier with the aid of a passive filter network (that comprises inductors and capacitors), or the signal can be separated earlier into the different frequency ranges, after which each signal range then has its own power amplifier and

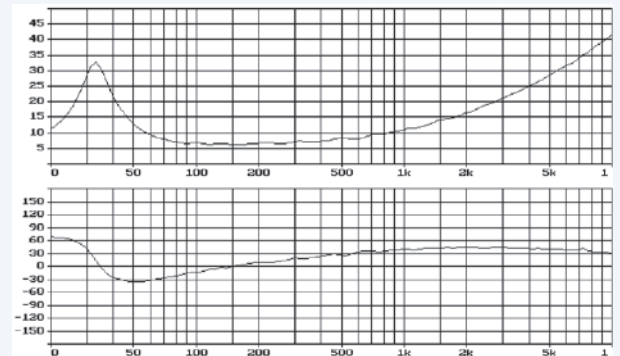


Figure 3: Impedance and phase characteristics.

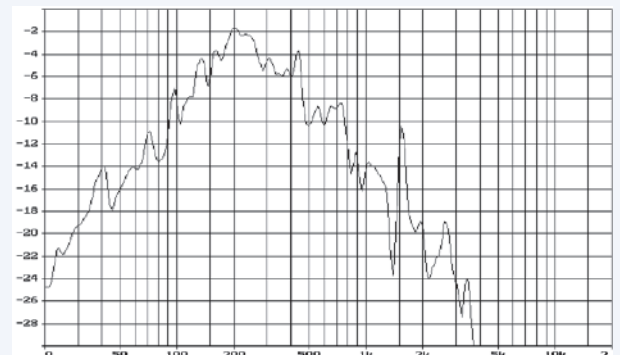


Figure 4: Frequency characteristic.

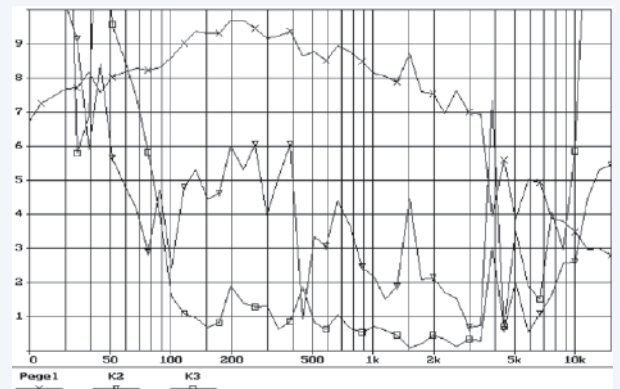


Figure 5: Distortion.

loudspeaker. In this case we speak of an active crossover filter.

As an aside, there is also a hybrid form possible: here the low frequencies are separated from the remainder using an active filter, and for this remainder a passive filter is used. This is sometimes called bi-amping.

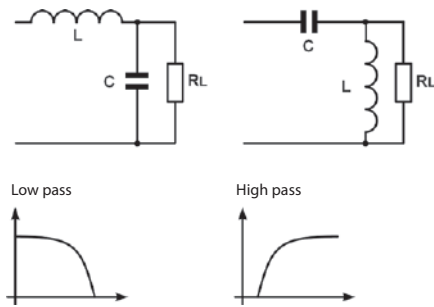


Figure 6: Passive low-pass and high-pass filters with a slope of 12 dB/octave.

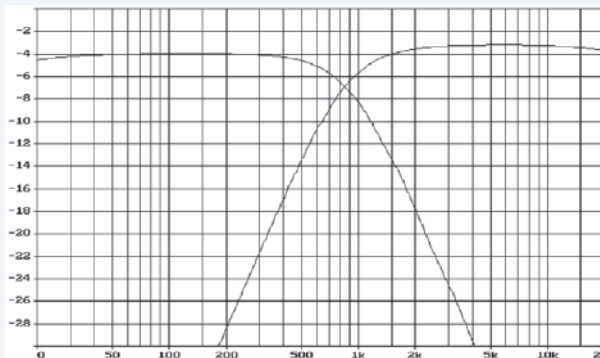


Figure 7: Frequency characteristics of a high-pass and a low-pass filter.

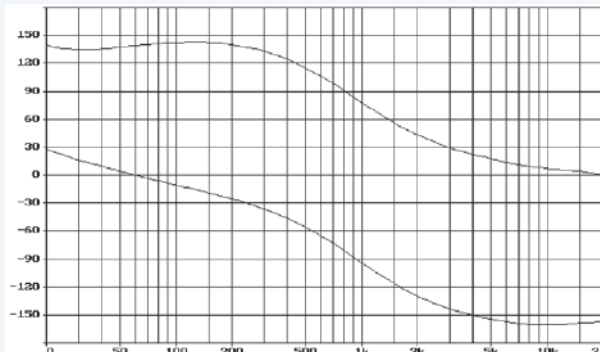


Figure 8: Phase characteristics of a high-pass and a low-pass filter.

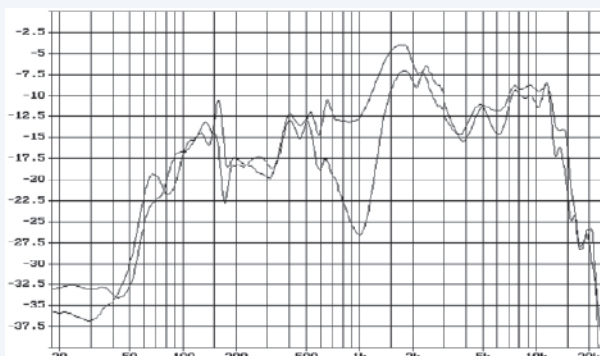


Figure 9: Correctly and incorrectly connected crossover filter.

Below we will continue with passive crossover filters; active filters will be dealt with in a later instalment.

Passive Crossover Filters

Figure 6 shows the schematics of both a low-pass and a high-pass filter with a slope of 12 dB/octave. Slopes of 6 dB/octave, which can be realised with a series inductor or capacitor respectively, must be avoided in audio applications. **Figure 7** illustrates the frequency characteristics of these filters.

The corner or crossover frequency of a filter is the frequency at which the level of the output signal has been reduced by 3 dB compared to the input signal. **Figure 8** demonstrates that at the corner frequency the phase of the output signal is displaced $+90^\circ$ (high-pass filter) or -90° (low-pass filter), compared to the phase of the signal in the pass-band.

When the loudspeakers are connected with equal phases, their signals in the region of the crossover frequency are displaced by 180° with respect to each other, and these signals will therefore then cancel each other. The result would be a 'dip' in the characteristic of the system, as can be seen in the lower curve of **Figure 9**.

In principle it doesn't matter which of the two speakers needs to have its terminals reversed, but to guarantee that any two loudspeaker boxes that are positioned near each other have the same phase, it is recommended that we stick to the international 'standard': the low-frequency speaker is connected normally and the high-frequency speaker is connected with reversed polarity.

Although having stated that, in practice we cannot blindly trust the labelling that the manufacturer places next to the terminals of their loudspeakers – firstly because any manufacturer can do whatever they please, and secondly, as a consequence of a difference in delays, the phase can actually be the opposite at the crossover frequency. Both with active and with passive filters the polarity of the loudspeakers must always be verified in practice.

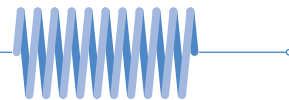
For the values of the components the following holds (for a Butterworth characteristic):

$$C = \frac{1}{2 \cdot \sqrt{2} \cdot \pi \cdot f \cdot Z} = \frac{0.1125}{f \cdot Z}$$

$$L = \frac{Z}{\sqrt{2} \cdot \pi \cdot f} = \frac{0.2251 \cdot Z}{f}$$

Various characteristics are popular for the design of crossover filters, each of which has its advantages and disadvantages with regards to the slope at the crossover frequency and impulse behaviour; for further details, we refer you to the relevant professional literature.

Figure 10 shows the basic schematic of a band-pass filter, which is really just a series connection of a low-pass and a high-pass filter.



The upper and lower corner frequencies are then also calculated using the familiar method. ◀

210564-01

The series of articles "Starting out in electronics" is based on the book, Basic Electronics Course, by Michael Ebner, published by Elektor.

Questions or Comments?

Do you have any technical questions or comments prompted by this article? Send an email to the author or to the editor of Elektor via editor@elektor.com.

Contributors

Idea and illustrations: Michel Ebner
Text and editing: Eric Bogers
Translation: Arthur de Beun
Layout: Giel Dols

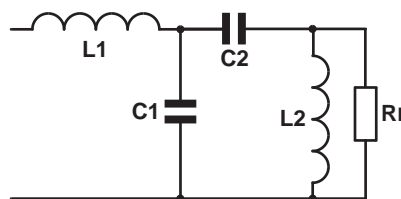


Figure 10: Band-pass filter.



RELATED PRODUCTS

- **B. Kainka, Basic Electronics for Beginners, Elektor 2020. (SKU 19212)**
www.elektor.com/13950
- **B. Kainka, Basic Electronics for Beginners (E-Book), Elektor 2020. (SKU 19213)**
www.elektor.com/18232

Post your ideas and electronics projects

all sizes / all levels / all sorts

at www.elektor-labs.com
and become famous!



Create a project now at:
www.elektor-labs.com

design > share > sell



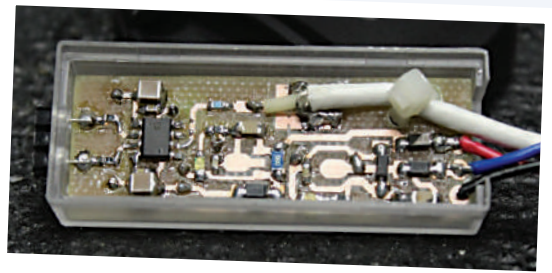


Differential Oscilloscope Current Probe 2.0

Elektor 11-12/2020, P. 68 (180665)

The INA849 from TI is a better pin-compatible alternative to the AD8421 used in this probe design. It does not require any frequency response compensation and has a significantly higher bandwidth. It is only necessary to adjust the value of the resistor which sets the gain.

Alfred Rosenkränzer



Joy-IT LCR-T7 Multifunction Tester

Elektor 11-12/2021, P. 22 (210365)

The article gives the impression that Joy-IT developed the described device itself. This is definitely not the case. The LCR-T7 is a copy of a community project based on an open source project (www.mikrocontroller.net/articles/AVR_Transistortester). The idea for an automatic transistor tester originally came from Markus Frejek, it was taken up by several other enthusiastic electronics hobbyists such as Markus Reschke, Karl-Heinz Kübbeler, Pieter-Tjerk de Boer and many others who improved the design using a lot of their own effort and expertise. There is also a special forum for the tester (www.mikrocontroller.net/topic/transistortester-avr), where development of the design is documented all the back to 2012. No reference is made to this resource in the *Elektor* article. I assume this omission was not deliberate but as a result of incomplete knowledge on your part. There is a lot of discussion these days about IP ownership, plagiarism and copyright. There is no suggestion that any law has been violated, as far as I know the original authors of this copied design have made no claims to copyright or sought any financial compensation for the commercial exploitation of their ideas. Fairness dictates that honor be given where honor is due.

Marcel "Derri" Derrmann

Dear Mr Derrmann,

Thank you for your detailed information. You are of course right that we did not set out to reinvent the wheel. The original developers of this project deserve great respect for their fantastic idea and the work they put into its design. Our company not only develops its own original equipment, but also takes great ideas and tries to build them into a quality finished product for developers and makers at the most attractive price possible. Our development department is often responsible for adding the 'finishing touch' here.

Volker Bode, SiMAC/Joy-IT



Balcony Power Plant

Elektor 09-10/2021, P. 44 (210326)

Want to use SmartHome sockets to measure solar power feed in? Our reader Jörg Trautmann wanted to do this and wrote us about his experiences:

In my own PV installation I was a bit disappointed that the first SmartHome sockets I tested could only measure power consumed by an appliance and would not show power fed into the socket from a grid-tie inverter and PV system. After a long period of searching, I discovered the LogiSmart PA0066 SmartHome socket from LogiLink, which can actually measure power fed into the mains. As far as I know, the simplest method for measuring current is to use a current clamp; the direction of current flow is irrelevant for the measurement. Here the units should use a different method of measurement, which is a very interesting aspect of their design.

Jörg Trautmann

N.B. The smart socket referred to accepts standard German domestic mains plugs, in the UK a grid-tie inverter should only be connected to the mains wiring via its own isolator.

Questions or Comments?

If you have any questions or comments, please let us know. You can reach us by email at editor@elektor.com.

Color to Sound

How to Read Out a Color Sensor via I²C

By **Annika Schlechter** (Germany) and
Volker Ziemann (Sweden)

Electronics can be helpful for handicapped people, as this educational project demonstrates. For those who cannot see colors themselves, a color sensor can do the job; a buzzer is used as acoustical feedback, for color and brightness information. Here we report on two versions of the project, based on an inexpensive Arduino Nano.

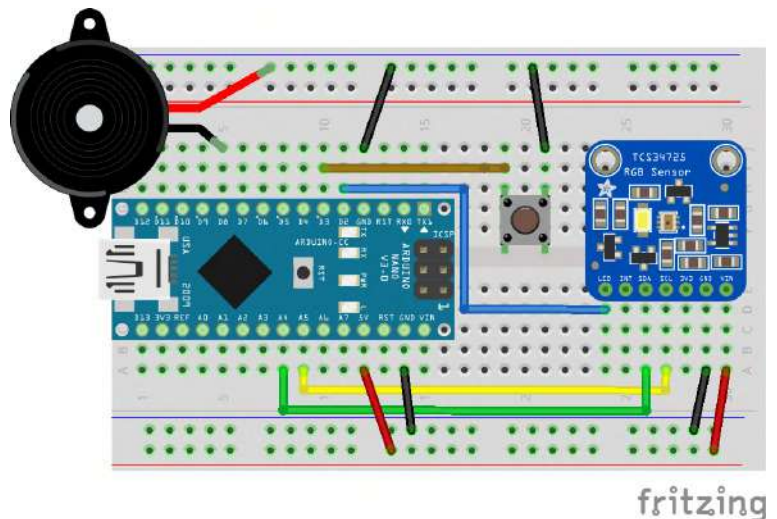


Figure 1: Schematics of the prototype. A piezo buzzer and a TCS34725 color sensor, mounted on a small breakout board, are connected to an Arduino Nano.

How can a seeing-impaired person select clothes with matching colors in the morning? Well, by converting the color information into a sound signal, which is the idea behind this project. The color is converted to the pitch of a sound and the brightness to its duration.

Figure 1 shows the schematics of a prototype, with a piezo buzzer and an ams TCS34725 color sensor mounted on a small breakout board [1]. Both are connected to an Arduino Nano. This prototype system was developed as part of the course “From Sensor to Report” at Uppsala University [2], which teaches basic data-acquisition skills including interfacing of sensors and microcontrollers to the outside world. Therefore, we do not use ready-made libraries to interface the sensor, but code the basic functions to read out the sensors ourselves. Moreover, the logic to assign the sounds to the colors initially resides in a Python script on a laptop, rather than on the microcontroller. The PC and microcontroller are communicating via a serial interface. We developed a communication protocol to send sensor values from the Nano to the PC; commands are going in the other direction to activate the buzzer with the right frequency. Later, however, we extended the functionality to allow stand-alone operation on the Nano. Still, this is not a production-ready system, but a prototype that shows the basic functionality and invites the reader to experiment further.

Before starting to assemble components, we need to install the Arduino development system (IDE) from [3] in order to develop programs for the Nano. The IDE is available for all commonly used operating systems. Please follow the instructions relevant for your system. If you have not used Arduinos before, please follow [4] to familiarize yourself with the

Arduino IDE. As a second prerequisite, we need to install python [5], which is likewise available for all systems. Windows and MAC OS binaries can be found on [5] and practically all Linux distributions have packages in their repositories. Again, follow the instructions for your system. But, enough of the preliminaries, let's get started!

The Color Sensor

The heart of this project is the TCS34725 color sensor [6], which comprises a 3x4 photo-diode array (**Figure 2**). Color-sensitive filters mounted in front of the diodes make them sensitive to red, green, and blue light. Moreover, a clear diode without a filter provides information

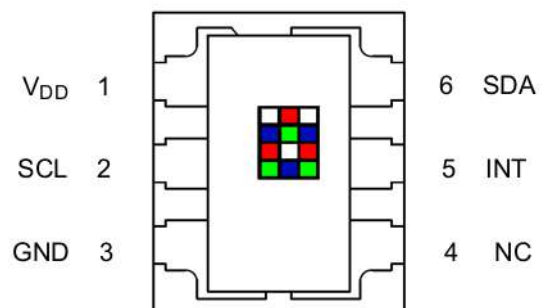


Figure 2: The TCS34725 color sensor integrates a 3x4 photo-diode array. Color-sensitive filters mounted in front of the diodes make them sensitive to red, green, and blue light. Source: Datasheet / ams [6].

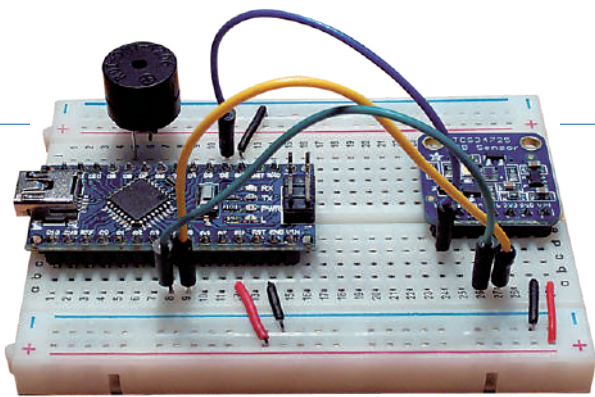


Figure 3: The hardware (see Figure 1) is assembled on a solderless breadboard (here without the button).

about the brightness. The analog signal from the diodes is converted to digital form by on-board analog-to-digital converters. The color information is made available via an I²C interface that connects to a microcontroller using only two wires for the communication, one for the clock signal (SCL) and one for data (SDA). The color sensor is then controlled by writing and reading registers on the device, but more on that later on where we describe the software.

The hardware is assembled on a solderless breadboard (**Figure 3**), pretty much following Figure 1 with the Nano shown on the left-hand side and the TCS34725 color sensor on the right. The latter is connected by black and red wires to Ground and the 5 V rail at the bottom. Likewise, the corresponding pins of the Nano are connected to the power rails. The green and yellow wires connect the data (SDA) and clock (SCL) pins to the corresponding pins on the Nano, which are situated on A4 and A5, respectively. The blue wire that connects the digital output pin D2 to the pin labeled **LED** on the color sensor allows to control a white LED that is mounted on the breakout board. The buzzer is connected to pin D8 on the Nano. The button that connects pin D3 to ground with the brown wires triggers a conversion when operating the Nano in the standalone mode of operation, discussed below.

This circuit is brought to life by an Arduino sketch that first includes the *wire.h* library to provide low-level functions to communicate over the I²C lines. These functions are used right away in our own two functions `I2Cread()` and `I2Cwrite()`, which encapsulate all communication with the sensor.

`I2Cread()` is used to readout a sensor value or parameter, stored in so-called registers. As parameters, the function gets the address of the sensor on the I²C bus, here `0x29`, and the register address (it took us a while to find out that `0x80` has always to be added to the latter). The function returns the contents of that register.

The function `I2Cwrite()` is used to send commands from the controller to the sensor, for example to set configuration values. This function gets the sensor and register addresses and, additionally, the new value that should be written to that register. No value (`void`) is returned from this function.

The Application

Let's return to our specific sketch. In the `setup()` function, we initialize the serial communication, the basic I²C functionality and the sensor (via function `color_begin()`). Inside the `loop()` function, which is executed indefinitely, we first check whether any command from the PC has arrived on the serial line. If that is the case, we read the register contents from the sensor, for all three colors. For one color, we have



RELATED PRODUCTS

- **Arduino Nano (SKU 17002)**
www.elektor.com/17002
- **Breadboard (830 Tie Points) (SKU 17092)**
www.elektor.com/17092

to read out two 8-bit registers and assemble the received bytes to a 16-bit word - following the instructions from the data sheet. This is shown in the following code snippet for the red color:

```
b3=I2Cread(TCS34725,0x16); // raw data, red
b4=I2Cread(TCS34725,0x17);
red=b4*256+b3;
```

Here `b3` contains the least-significant byte and `b4` the most-significant byte, which we multiply by 256 and add to `b3` to obtain the 16-bit variable `red`. We treat the other colors likewise; only the register addresses are adapted.

Additionally, we use the brightness reported from the un-filtered diode and store it in the variable `clea`. This will determine the duration `d` of the sound. If the value of `clea` is less than 1000, we emit a 0.5 s long tone; and if it is larger, we use 1.5 s.

Now we read the received command and copy the result into a character array `line`, so we can use the standard C function `strstr()` to find out which command was sent. The following code snippet illustrates this:

```
Serial.readStringUntil('\n').toCharArray(line,30);
if (strstr(line,"color?")==line) {
    Serial.print(clea); Serial.print(" ");
    Serial.print(red); Serial.print(" ");
    Serial.print(green); Serial.print(" ");
    Serial.println(blue);
} else if (strstr(line,"tone_red")==line) {
    tone(buzz, 262, d);
    ...
```

We see that the command `color?` causes the Nano to send the values of the four signals `clea`, `red`, `green`, and `blue` back across the Serial line. If `tone_red` is received, the built-in `tone()` command is used to activate the piezo buzzer - in that case with the frequency which is assigned to the red color. Several equally constructed `strstr()` blocks follow and trigger the appropriate actions.

Note that we use a simple protocol, based on sending character strings back and forth; a request from the laptop is terminated with a question mark, for example `"color?"`, and the Nano responds by sending the values back. An instruction is based on just sending the command, for example, `"tone_red"`, which causes the buzzer to sound with 262 Hz for the duration specified by `d`. The communication vaguely mimics the

SCPI command language that is supported by modern oscilloscopes and other test and measurement equipment. This makes interfacing to external programs that support accessing the serial line very simple; python, octave, Matlab, and Labview all support this.

The PC Program

We used Python 3 on the laptop to communicate with the Nano. The very basic code is shown in **Listing 1**.

After importing support for the serial communication and basic timing functionality we open the Serial port to which the Nano is connected at the baud rate that matches that chosen on the Nano. Giving the operating system 3 seconds to establish the connection we then send the command "color?".

Note that Python 3 encodes character strings as Unicode, whereas the Nano expects plain ASCII strings. We therefore have to prepend the letter "b" to the command "color?" in order to send the string as plain ASCII data. After another short wait, we receive the response from the Nano in the variable `reply`, which contains the values of the four colors that we individually retrieve with the `.split()` method and assign to mnemonic names, such as `red`.

Having the values available, we are ready to implement the logic to assign the sounds to the colors. This assignment is based on quite a bit of experimentation with colored paper sheets until a good setup was found. We used paper with a light, medium, and dark tone for each of the three base colors in order to calibrate the sensor. We repeatedly measured the respective color values at a fixed distance and used that information to specify the constants in the python code shown above. For example, the command `tone_red` is sent to the Nano, if the red component, normalized to the intensity, is larger than 0.6. Next we test the green intensity and send `tone_green` if the threshold is exceeded. Finally, the value of the blue component is tested. If even that does not qualify, "no_color" is sent to the Nano, causing it to emit the corresponding tone. Certainly some experimentation with the constants is needed in order to adapt the system to your wardrobe.

Standalone Version

The standalone version of the code is available from this article's website [7]. It closely follows the example above, except that instead of waiting for a command arriving on the Serial line, the Nano now waits for the button to be pressed before reading the sensor and emitting the appropriate sound. Moreover, the color information is directly processed on the Nano as shown in the following snippet:

```
if (red/clea > 0.6) {
    tone(buzz, 262, d);
} else if (green/clea > 0.3) {
    ...
}
```

About the Authors

Volker Ziemann's interest in electronics started with *Elektor*, around the time of the 40W

Edwin amplifier (mid-1970s), but he got sidetracked to studying physics and worked with particle accelerators ever since — SLAC in the US, CERN in Geneva, and now in Uppsala, Sweden. Since electronics plays an important role for control and data acquisition in accelerators, his early interest has been useful throughout his career. He now teaches at Uppsala University and has authored three books, one of them about data acquisition with Arduino boards and the Raspberry Pi.

Annika Schlechter is studying toward an MSc in Physics at the University of Heidelberg, Germany. At the moment, she is continuing her studies during her Bachelor thesis at the German Cancer Research Institut in Heidelberg. This project began during an Erasmus semester at the University of Uppsala, Sweden, where she attended Volker's lecture "Sensor to Report."



Listing 1: Python script for laptop communicating with Arduino Nano.

```
# colortosound_arduino.py
import serial, time

ser=serial.Serial("/dev/ttyUSB0",9600,timeout=1)
time.sleep(3);

ser.write(b"color?\n") # write command to get colors
time.sleep(2);

reply=ser.readlines() # read answer

# process answer to get color values
colors = reply[0].split()
for i, entry in enumerate(colors):
    colors[i]= int(entry.decode('utf-8').replace('\r\n', ''))

intensity = colors [0]; red = colors [1]
green = colors [2]; blue = colors [3]

# write command to control buzzer according to color

if red/intensity > 0.6:
    ser.write(b"tone_red\n")
elif green/intensity > 0.3:
    ser.write(b"tone_green\n")
elif blue/intensity > 0.24:
    ser.write(b"tone_blue\n")
else:
    ser.write(b"no_color\n")

ser.close()
```

If it is desirable to always illuminate the clothes with the built-in LED on the breakout board, we can turn it on with `digitalWrite(led,HIGH)` just before reading the registers from the TCS34725 and turning it off afterwards.

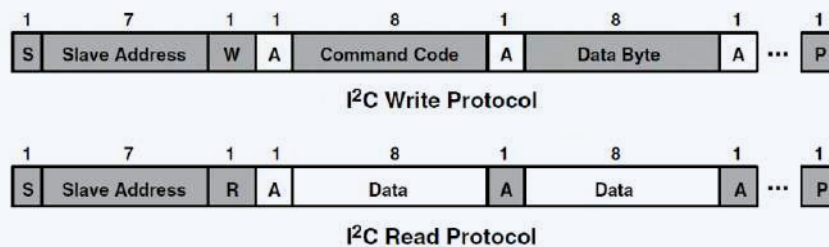
I²C Communication

The communication between two integrated circuits, hence I²C, is based on a synchronous serial protocol that is carried by two wires. It follows the simple rule to detect the voltage level on one of the wires, called SDA, after the voltage on the other wire, called SCL, has changed. One such reading provides a single bit and eight consecutive bits constitute one byte. One of the participants on the I²C bus, usually a microcontroller, orchestrates the communication and always provides the clock signal. The other participants have a chance to enter the communication, if they are addressed and given the right to put their information on SDA. Moreover, after each byte the sensor is given one clock cycle to acknowledge receipt of the previous byte.

In order to initiate the communication with a sensor, the microcontroller sends one byte. The first seven bits of that byte contain the address and the last bit indicates whether the controller or the sensor is in control of SDA to transmit the following byte. If the controller wants to continue sending, for example to change a register on the sensor, it sends two additional bytes, the register address and the new value. If, on the other hand, the controller wants to

read a register from the sensor, it transmits the bit to indicate that the sensor is now in charge of SDA, and reads SDA synchronously with the clock signal that the microcontroller continues to generate.

The upper image in the figure illustrates writing to the sensor. The bits that are shaded dark are controlled by the microcontroller and the un-shaded bits by the sensor. To initiate a transaction, the microcontroller creates a start condition, denoted 'S', followed by the seven bits of the sensor's address and the eighth bit set to indicate that the controller wants to continue sending. One bit 'A' is provided by the sensor to indicate that it understood. The second byte contains the address of the register on the sensor and one byte with the value to be written to the register. Note the bit labeled 'A' where the sensor has to acknowledge each received byte. Reading from the sensor, shown by the lower image in the diagram, works pretty much the same way, only the eighth bit of the first byte now indicates that the sensor is given the right to 'speak' and SDA is subsequently controlled by the sensor, while the microcontroller has to acknowledge every received byte.



Source: TCS34725 data sheet / ams [6].

In a third stage, we can make the system portable by using a bare ATmega328, flashed with an Arduino bootloader, connected to the sensor. Powered from a battery as long as a button is pressed, the system starts up and continuously converts color to sound until the button is released to disconnect the battery. The software is also available from the article's web page. Building the system is left as an exercise for enthusiastic readers.

In the "Sensor to Report" course the color sensor was quite popular. A second student, who wanted to monitor the color of the sky over a longer time to later analyze the data, used essentially the same setup with the Nano communicating with a host computer via the Serial line. She used a python script on a Raspberry Pi to store the data in a MySQL database, also running on the Pi, and later used Octave to prepare plots of the colors that were recorded over about a week. The

plots were recorded just before Christmas and made us painfully aware of how short the days can get in Uppsala. Further data acquisition projects, following the same spirit, are described in [8].

210051-01

Questions or Comments?

Do you have technical questions about this article? Please email the editors at editor@elektor.com.

Contributors

Design and Text: Annika Schlechter, Volker Ziemann
Editor: Jens Nickel
Layout: Sylvia Sopamena

WEB LINKS

- [1] TCS34725 breakout board: <https://www.adafruit.com/product/1334>
- [2] "Sensor to Report" course: <https://ziemann.web.cern.ch/ziemann/teaching/s2r19/>
- [3] Arduino website: <https://www.arduino.cc/>
- [4] Arduino tutorials: <https://www.arduino.cc/en/Tutorial/HomePage>
- [5] Python website: <https://www.python.org/>
- [6] TCS34725 Color Sensor datasheet: <https://ams.com/tcs34725#tab/documents>
- [7] Software Download: <http://www.elektormagazine.de/210051-01>
- [8] V. Ziemann, A Hands-on Course on Sensors using the Arduino and Raspberry Pi, CRC Press, 2018.

BattLab-One

Measure and Optimize the Battery Life of IoT Devices

By **Tam Hanna** (Slovakia)

When you need to choose a suitable battery for your next battery-powered IoT design, you could sit down with a DVM and measure the active and sleep current to estimate roughly how long the device will run from a single charge. The design could then be optimised by perhaps altering the ratio of sleep to active mode and maybe by choosing a different battery type and capacity. The BattLab One is a USB-powered device that can do all this at a click of a mouse button. It both powers the device under test and also shows all data relating to expected battery life.

Obtaining a consistent measurement of the energy consumed by a device running a microcontroller is not a trivial task. As with any computer chip, power consumption depends heavily on the instantaneous processing load and operational status. It is not only necessary to consider normal operation and standby or deep sleep states, in which the current consumption of common microcontrollers can vary between less than 1 μA and several hundred milliamps (mA). Some peripherals in the device may also only be active at certain times so that current consumption will not be constant. A specialized tool like BattLab-One is very useful in this respect and can take this dynamic behaviour into account to make a reasonably realistic estimate of the operating time of the device when powered by different battery types.

What Is BattLab-One?

In the field of power consumption measurements, a source measure unit (SMU) is a useful tool to have on the workbench. This type of power supply provides power to the device under test (DUT) and also

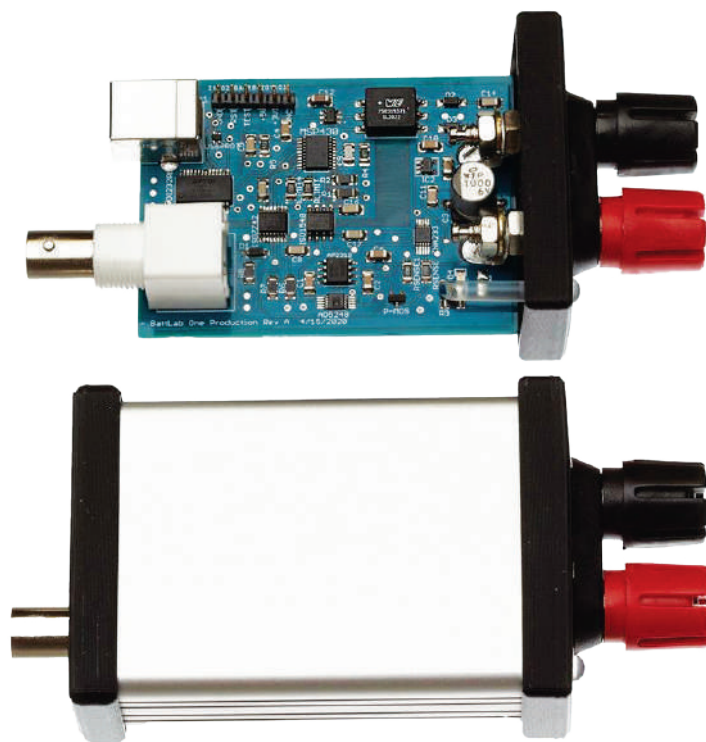


Figure 1: The BattLab-One and its aluminum housing.

makes accurate measurements of power consumed over defined periods of time. Professional examples of SMUs are manufactured by Keithley and Keysight. The BattLab-One from Bluebird Labs aims to fulfill a similar function, but for a price of less than €100. The device (including metal enclosure) is available from the Elektor shop [1].

Refer to **Figure 1** to see the USB type B socket on the left together with a BNC socket that accepts a trigger input signal. On the right are the two 4-mm sockets from which the DUT can be powered. **Figure 2** shows the plastic collars, which can be unscrewed to allow croc clips to be attached to the posts. A closer look at the case end caps shows a surface texture consistent with 3D printing.

BattLab-One is powered from a computer USB port and is able to supply devices with a range of supply voltages from 1.2 to 4.5 V and currents up to a maximum of 450 mA. The discharge profiles of frequently used battery types (such as Li-Ion, LiFePO₄, alkaline, NiMH and NiCd technology) are included in the software and used

to calculate expected battery life. BattLab-One does not take into account subtleties such as load and battery age related voltage drops. According to the manufacturer, the measuring device offers a basic battery simulation and a current measuring range from 10 μ A to 500 mA. A 16-bit ADC sampling at 1 kHz is incorporated into the design to measure current to the DUT.

BattLab-One and Linux

Software for the device was developed by Bluebird Labs using Python. As a first attempt, I was keen to get the system working directly under Ubuntu 20.04 LTS. The software source code is available from GitHub [2] and can be compiled using the usual methods. However, Bluebird Labs has taken a different approach by using *pipenv*. Anyone who, like me, has only installed the usual *venv* variant needs to install *pipenv*. For me the process went as follows:

```
tamhan@TAMHAN18:~/BattLabonespace/
BattLab-One$ sudo apt install pipenv
```

The first problems arose when setting up the Virtual Environment, as Python version 3.7 is assumed. If, like me, you want to use a newer version (i.e., version 3.8 preinstalled under Ubuntu 20.04 LTS), the following statement will be required:

```
tamhan@TAMHAN18:~/BattLabonespace/
BattLab-One$ which python3
/usr/bin/python3
tamhan@TAMHAN18:~/BattLabonespace/BattLab-One$ pipenv
install --dev --python /usr/bin/python3
```

Now let's consider the process of creating a virtualenv for this project. The software issues warnings (**Figure 3**) when the commands are entered. The virtual environment is therefore activated according to the following scheme. It will remain active like a normal *venv* environment until the terminal window is closed:

```
tamhan@TAMHAN18:~/BattLabonespace/BattLab-One$ pipenv
shell
...

```

Execution of the three .py files with the software is now possible using to the following method:

```
(BattLab-One-CK7P-15V) tamhan@TAMHAN18:~/
BattLabonespace/BattLab-One$ python3
BattLab-Release_V1.2.1.py
Traceback (most recent call last):
  File "BattLab-Release_V1.2.1.py", line 59, in
<module>
    import pkg_resources.py2_warn
ModuleNotFoundError: No module named 'pkg_resources.
py2_warn'
```



Figure 2: Croc clip power hook up.

```
tamhan@TAMHAN18:~/battlabonespace/Battlab-One$ pipenv install --dev --python /usr/bin/python3
Creating a virtualenv for this project...
Using /usr/bin/python3 (3.8.10) to create virtualenv...
created virtual environment CPython3.8.10.final.0.64 in 327ms
creator CPython3Posix(dest=/home/tamhan/.local/share/virtualenvs/Battlab-One-CK7P-15V, clear=
seeder FromAppData(download=False, pip=latest, setuptools=latest, wheel=latest, pkg_resources
activators BashActivator,CShellActivator,FishActivator,PowerShellActivator,PythonActivator,Xo

Virtualenv location: /home/tamhan/.local/share/virtualenvs/Battlab-One-CK7P-15V
Warning: Your Pipfile requires python_version 3.7, but you are using 3.8.10 (/home/tamhan/.loca
$ pipenv check will surely fail.
Installing dependencies from Pipfile.lock (4cf014)...
 2 11/11 - 00:00:07
To activate this project's virtualenv, run the following:
$ pipenv shell
tamhan@TAMHAN18:~/battlabonespace/Battlab-One$
```

Figure 3: PipEnv error message.

Unfortunately, Bluebird Labs does not comply with the setup tools specification but explicitly includes a library that is only available in a few versions of the setup tools. The .py file must be edited to remove the following statement:

```
import pkg_resources.py2_warn
```

Now the software has a problem with the local paths:

```
(BattLab-One-CK7P-15V) tamhan@TAMHAN18:~/
BattLabonespace/BattLab-One$ python
BattLab-Release_V1.2.1.py
...
_tkinter.TclError: error reading bitmap file "icons\
bbirdlogo.xbm"
```

The solution is to modify the TK-Inter-GUI by removing the line:

```
root.iconbitmap(bitmap=GetIconPath())
```

to remove the program icon. On this occasion you can adjust the file path using:

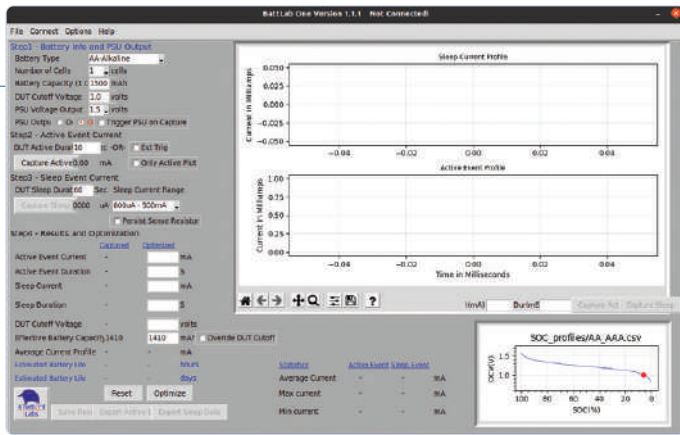


Figure 4: The BattLab Linux GUI.

```
img = PhotoImage(file='/home/tamhan/BattLabonespace/
BattLab-One/icons/bbirdlogo_png1.png')
```

The next attempt to execute it shows that the TK-Inter environment is looking for the *bblogo.gif* file in the wrong folder

```
_tkinter.TclError: couldn't open "/home/tamhan/.
local/share/virtualenvs/BattLab-One-CK7P-15V/lib/
python3.8/site-packages/matplotlib/mpl-data/images/
bblogo.gif": no such file or directory
```

You can therefore copy any GIF file into this directory and give it the appropriate name. If you now comment out the following line, you can start the program with a slightly limited range of functions (Figure 4):

```
#toolbar.children['!button8'].
config(command=select_range)
```

Deactivation of the measuring range button turns out to be uncritical in practice, because the Matplotlib diagram is problematic in terms of usability anyway. Many labels appear cut off and the size of the window cannot be adjusted.

As we will see, the program has fewer problems running in Windows 10. Before changing the operating system, we can tidy up to remove the created virtual environment and thereby free up mass storage space:

```
(BattLab-One-CK7P-15V) tamhan@TAMHAN18:~/
BattLabonespace/BattLab-One$ exit
exit
tamhan@TAMHAN18:~/BattLabonespace/BattLab-
One$ cd /home/tamhan/.local/share/virtualenvs/
BattLab-One-CK7P-15V
tamhan@TAMHAN18:~/BattLabonespace/BattLab-
One-CK7P-15V$ rm * -rf
tamhan@TAMHAN18:~/BattLabonespace/BattLab-
One-CK7P-15V$ ls
```

BattLab-One Running in Windows 10

The situation here is simpler in that ready-made .exe files are

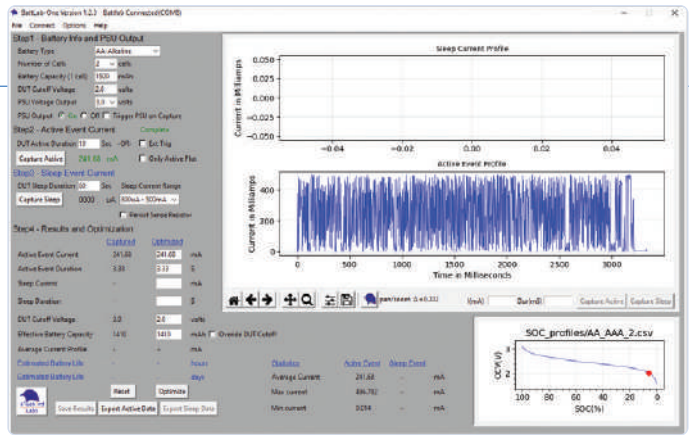


Figure 5: BattLab-One records and displays the DUT current consumption.

provided under [3]. The ZIP archive includes, among other things, the *SOC_profiles* subfolder, which contains the various battery profiles in tabular form. You will need to unzip the archive.

When *BattLab-Release_V1.2.3.exe* file is executed, the user interface display looks similar to the Linux version in Figure 4. The Windows version behaves largely in the same way but has fewer rendering errors. First, we can select the battery type up in the top left area. The associated battery data is used to create the characteristic discharge curve shown at the bottom right. To the right of the *PSU Output* field, *On* and *Off* buttons switch the integrated voltage converter to power the DUT connected at the two output sockets.

I have been working on my own IoT sensor project called *HygroSage* for some time now. With its colour display and powerful processor, it would be an ideal candidate to test out BattLab-One. I connected the sensor. By clicking on the option *Capture Active*, the software shows a flickering green progress bar labeled *Active Event Current* and then does nothing visible for 10 seconds. During this time, the system measures the information and finally displays it as shown in Figure 5.

The value of any point of the waveform can be displayed using the cursor. In practice, however, this does not work quite as convincingly because the value indicated in the text box below does not show the measured value, but the position of the mouse pointer on the graph. This means you need to move the mouse pointer to a point on the curve. In addition, there is no really convenient way to zoom into the diagram (generated by Matplotlib). Despite the full HD resolution of my screen, I wasn't able to expand the user interface screen but it is possible to select an area of the waveform using the *Zoom to Rectangle* option.

For testing purposes, I connected a calibrated, high-quality multi-meter in series with the DUT, which measured a current of around 32 mA. BattLab-One did not agree with this; it would seem the type MCP1640 switching regulator in BattLab-One generates noise that actually interferes with BattLab-One. As a second test, I used a 1-kΩ resistor as a load for BattLab One and specified output voltage of 3 V. The result is the trace showing measurement quantisation noise in Figure 6. Connecting a 100-μF electrolytic capacitor in parallel to the *HygroSage* did not result in any significant improvement of the quantised current consumption waveform.

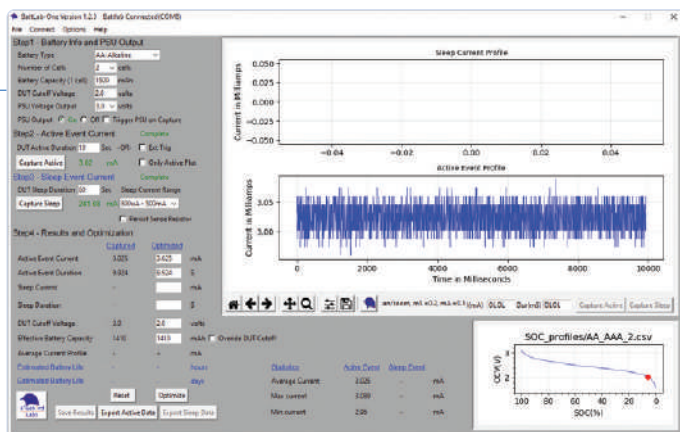


Figure 6: Trace showing current through a 1-k Ω -resistor at 3 V with some noise evident.

Sleep Mode Capture and Optimization

Once the active current consumption has been captured in Step 2, we can move on to Step 3 to measure sleep current. Here we need to ensure the DUT will be in sleep mode when the measurement is taken. We now enter parameters as we did in Step 2 for the active mode capture. Unless things went badly wrong with your design the device will always consume significantly less power in sleep mode. BattLab-One gives you the option to switch the current range in sleep mode between 10 μ A – 800 μ A or 800 μ A – 500 mA to improve measurement resolution. Be aware that the device shouldn't draw too much current in the lower measuring range. According to the documentation, a load current exceeding 250 mA can damage the hardware. With the DUT running in sleep mode we can now press *Capture* to record the current.

The results after a test with *Hygrosage* are shown in **Figure 7**. In Step 4, the system displays the calculated values. Using this data we can play around with the battery characteristics and capacity together with the active/sleep times and then click on *Optimize* to easily see what sort of battery life we can expect for the device when it is deployed in the field.

The battery life prediction will be reasonably accurate for devices which operate with a defined active and sleep period. Some devices rely on an external event interrupt to wake from a sleep mode; in this case, we can only work out the battery life by using an estimate of how often we can expect the interrupts to occur.

To Sum Up

The BattLab-One hardware is quite usable. This software however has scope for improvement. The design is open-source so we can expect its evolution to benefit from community support. This also

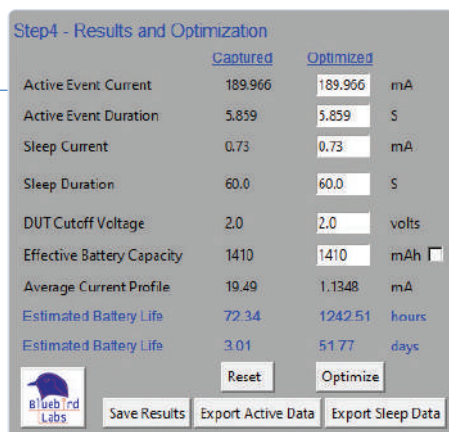


Figure 7: Battery life can be optimised using sleep and active mode current values and battery capacity.

gives you the opportunity to explore BattLab's inner workings and even to adapt and improve the software as required so that BattLab-One fits your own requirements. When it comes to hardware, BattLab-One does a lot of things right and very little wrong. Its low price and high performance makes it a good addition to the test bench for those developing IoT devices. ◀

210473-01

Questions or Comments?

Do you have any technical questions or comments about this article? Contact the author at tamhan@tamoggemon.com or Elektor at editor@elektor.com.

Contributors

Text and illustrations: **Tam Hanna**
 Editors: **Dr. Thomas Scherer, C. J. Abate**
 Translation: **Martin Cooke**
 Layout: **Giel Dols**



RELATED PRODUCTS

- ▶ **BattLab-One – Battery Life Optimizer, with Enclosure (SKU 19757)**
www.elektor.com/19757

WEB LINKS

- [1] BattLab-One: www.elektor.com/battlab-one-battery-life-optimizer-with-enclosure
- [2] Software at Github: <http://github.com/petersdw1/BattLab-One.git>
- [3] Software for Windows 10: <https://github.com/petersdw1/BattLab-One/releases/tag/V1.2.3>

Simple Earth-Leakage Tracer

Testing Isolation of
Mains Supply



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

Residual-current circuit breakers are essential safety devices in our domestic mains supply installation, but sometimes they can switch off (trip) for no apparent reason. A professional earth-resistance tester (or megohmmeter) will help to locate the leak, but the easy-to-build earth-leakage tracer presented in this article proves to be a safe, affordable and useful alternative, using standard CR2032 lithium batteries for the high-voltage power supply.

WARNING. Working with mains voltage can be fatal. The circuit described here is not for beginners. Do not build or use it unless you are experienced in dealing with mains voltages!

Residual-current circuit breakers have saved many lives from fatal electric shocks. Where the older generation knows examples of people who were electrocuted because of — for example — a broken lamp socket, this is unlikely to happen with more modern electrical installations. A residual-current device (RCD) detects current differences between the Live (L) and Neutral (N) leads (i.e., if current leaks via the earth). If this difference is more than 30 mA, the residual-current circuit breaker will switch off. An alternating current of around 30 mA through the human body is potentially sufficient to cause cardiac arrest or serious harm if it persists for more than a fraction of a second. RCDs are designed to disconnect the conduction wires quickly enough to prevent serious injury [1].

An earth-leakage circuit breaker does not only switch off when a person touches the live wire; moisture for example can also cause leakage currents, like moisture in an outdoor wall socket or lamp. The source of such a problem is often difficult to locate. A "simple" way is to first switch off the mains power, then disconnect the various wires at the junction boxes, switch the power back on to see if the earth-leakage switch still trips and thus locate the problem by trial and error, but this method does have its disadvantages. First of all, the power has to be switched off and on again every time wires are (dis-)connected, which increases the risk of errors and of accidentally working on a live circuit. In addition, some earth-leakage problems occur only sporadically, not all faults are easily reproducible.

Why Design This Circuit?

The designer of this project had problems with the electrical system in the sunroom at his parents' house: the RCD of this circuit occasionally tripped for no apparent reason. When he came to check the circuit during the weekend, the earth-leakage switch remained switched on during testing, but a few days later, it tripped again. Rather than simply testing if and when the RCD switches off, it is much better to actually measure the leakage current. If we can measure leakage current below RCD trip level, it is much more convenient to tackle (potential) problems.

How to Measure

In case of isolation issues in mains circuits, measuring resistance with a standard multimeter simply does not work. Anyone who has measured the resistance of their skin with a multimeter will know the paradox: it shows a resistance of around 1 M Ω (depending on how wet or sweaty your hands are). That would mean that if you touch 230 V mains voltage, the current would be well below 1 mA, which is far from lethal; you cannot even feel a current of this magnitude. However, the resistance is very much dependent on the voltage that is applied to perform the resistance measurement, with most multimeters this will not exceed 9 V. At 230 V, however, the resistance of the human body is in the order of kilo-ohms, which leads to currents of a few hundred milli-amps, which can certainly be harmful or even lethal. This voltage dependency of the resistance is caused, among other things, by electrochemical reactions in the human body.

So in order to make an accurate measurement of the leakage current, we must not measure at a low voltage — as a multimeter does — but ideally with 230 V, which is the mains voltage in most European countries. The challenge for the author was to do this in a cheap, but

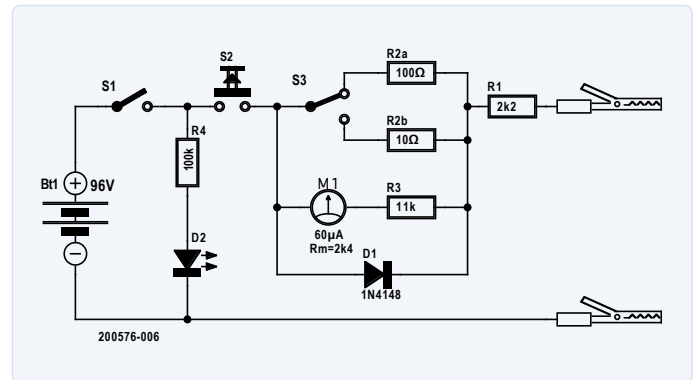


Figure 1: The schematic diagram of the earth-leakage tracer.

also — even more important — in a safe way. In the latter respect, a lower voltage may be advisable and with the solution chosen for the power supply it also cuts the costs: the lower the voltage, the less batteries we need to power this earth-leakage tracer.

Why Batteries?

The CR2032 is a 3 V lithium battery that, as the name implies, has a diameter of 20 mm and is 3.2 mm thick. This makes this battery about the same diameter as a standard C-battery, which is approximately 50 mm long. With 16 CR2032 batteries stacked on top of each other, we have about the same form factor as a C-battery, with a voltage of 48 V. With two such "C-batteries" connected in series, we get 96 V and a standard, dual C-cell battery holder can be used. CR2032s are widely used in portable electrical appliances, toys, and gadgets, making them almost everywhere available at very reasonable prices. Other options, for about the same price, would be to use 64 LR44s, or eight A23 batteries, but there are no standard battery holders for sale that can accommodate these stacks of batteries.

At first, another consideration for the use of these batteries in this design was that, according to some sources (although it is difficult to find conclusive data on this!), CR2032s can deliver only a relatively low current, on the order of 10 mA. A voltage of 96 VDC is potentially dangerous, for direct current (DC) the lethal current for humans is about 100 mA. But, according to the datasheets, the batteries cannot deliver such a high current; so in theory, the batteries would make a safe power supply for the earth-leakage tracer. In practice, however, the author measured (peak) currents of a few hundred milli-amps that could be sustained long enough to be potentially dangerous (see textbox about the CR2032s). Therefore, for safety reasons, this design includes a current limiting resistor.

The Hardware

For the measurement of the leakage current, the author used an old-fashioned panel current meter that he still had lying around. The advantage of an analog meter is that it reacts faster than a (standard) digital meter, so peak currents when switching on are also visible. The resistance of this meter (R_m) is 2.4 k Ω and the full-scale current (I_{FS}) is 60 μ A. With resistors, we must ensure that the meter has the correct

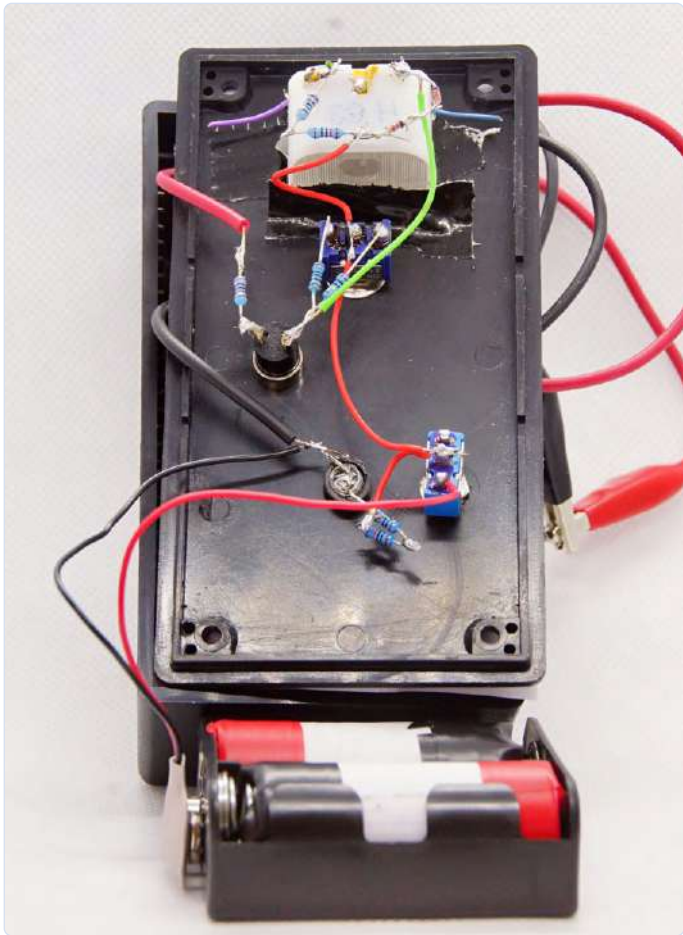


Figure 2: How the author's prototype is constructed.

range. But it is also wise to protect the meter against high currents with a diode. A diode will limit the voltage across the meter and its series resistor R3 to 0.7 V.

Figure 1 shows the schematic of this earth-leakage tracer. R1 is the current limiting safety resistor; to be on the safe side, a resistor of 2.2 kΩ is used, which limits the current to around 45 mA. If this maximum current flows, 4 W will be dissipated in resistor R2. So, either use a power resistor or let the current only flow for a short time.

With the shunt resistor R2, the resistor R3, and the internal resistance R_m of the meter we can, under the condition that R2 is much smaller than $R_3 + R_m$, use Ohm's law to calculate the correct range of the meter:

$$R3 \leq \frac{V_{diode}}{I_{FS}} - R_m \quad R2 = \frac{(R_m + R3) \times I_{FS}}{I_{scale}}$$

where I_{scale} is the actual maximum output current of the circuit.

We can then tweak the resistor values so that standard (E12 range) resistors can be used. The author used the following parts for his meter:

HOW MANY CR2032s ARE THERE?



The popularity of this type of battery has the less favorable side-effect that there are numerous manufacturers producing CR2032s. Apart from the renowned brands, there is an unknown number of manufacturers making lithium batteries with a voltage of 3 V with the same dimensions (20 mm diameter, 3.2 mm thick) from which this button cell takes its type number. When it comes to data like capacity, internal resistance and maximum current, the "CR2032 standard" turns out to be much less of a standard. And when it comes to the cheaper ones — the ones you would prefer to buy when you need 32 batteries — it is impossible to find out from which production line they originate, and the packaging usually provides no more electrical data than "3 V". In retrospect, the general data found by the author indicating that the short-circuit current of these batteries would remain on the order of a few tens of milli-amps turned out to be very optimistic in practice, from a safety point of view. In practice, the CR2032s that he bought turned out to have a short-circuit current of about 300 mA, which gradually dropped to 80 mA after ten seconds. For this reason, it would be a bad idea to omit resistor R1 that limits the current from the earth-leakage tracer. Better safe than sorry!

- R3 is 11 kΩ (10 kΩ and 1 kΩ in series). This limits the meter to value '7' on a full-scale of '8': the diode limits the voltage across R3 and the meter to 0.7 V, resulting in a maximum current of approximately 53 μA, while the full-scale value is 60 μA. The author therefore uses two diodes in series with a total voltage of 1.4 V, so that full-scale range can be obtained, while the current through the meter stays within limits.
- R2a is 100 Ω (for 8 mA range)
- R2b is 10 Ω (for 80 mA range)

Given the simplicity of the circuit and the small number of components, no circuit board is required to build this project, **Figure 2** shows the inside of the earth-leakage tracer built by the author. At the bottom of this picture, you see the two stacks of CR2032 batteries wrapped in tape, in a standard dual C-cell battery holder. The plus and minus terminals of these home-made 46 V batteries are marked with red and black tape, respectively.

Measuring Earth-Leakage Current

Using this earth-leakage tracer is simple. When the device is switched on with S1, the power LED D2 lights up. When button S2 is pressed, the measuring current flows to the terminals, crocodile clips are preferred to connect the test leads to the wires of the circuit that's being tested.

The current can be read from the meter. In this way, we can check all wires in the junction boxes on leakage currents.

With the 96 V test voltage from the batteries, the current measured is of course lower than the current that would actually flow through the earth-leakage at 230 V mains voltage. That is, if we assume that the resistance of the leakage only marginally changes between 96 and 230 V, a proportionally higher current will flow at a higher voltage. In addition, the internal resistance of the battery, the shunt and the safety resistor R_{int} will also reduce the current. The internal resistance can easily be determined by measuring the short-circuit current of the earth-leakage tracer with a multimeter. In the case of the author's prototype with 10 Ω shunt (R2b) selected, this was 38 mA and thus an internal resistance of 2526 Ω . From the current measured I_m , the actual leakage current I_{actual} at mains voltage can then be calculated with:

$$I_{actual} = \frac{V_{mains}}{\frac{V_{batt}}{I_m} - R_{int}}$$

Plotted in a table this is:

I_m [mA]	I_{actual} [mA]
0	0
1	2
2	5
3	8
4	11
5	14
6	17
7	20
8	24
10	31
20	100
30	>200

And what the issue was with the electrical installation of my parents' sunroom? A cable to an outdoor lamp had a isolation leakage that indicated 4 mA during the troubleshooting with this earth-leakage tracer, which would in practice be about 11 mA with the mains voltage switched on. Probably the leakage became worse during the rain. The best way to solve the problem was to replace the cable. ◀

200576-01

WEB LINK

- [1] **Residual-Current Device:**
https://en.wikipedia.org/wiki/residual-current_device

Contributors

Design and text: **Elbert Jan van Veldhuizen**

Editing: **Luc Lemmens**

Illustrations: **Elbert Jan van Veldhuizen, Patrick Wielders**

Layout: **Giel Dols**

Questions or Comments?

Do you have technical questions or comments about his article? Email the editor at luc.lemmens@elektor.com or contact Elektor at editor@elektor.com.



RELATED PRODUCTS

> PeakTech 2710 Digital RCD Tester (SKU 19318)

www.elektor.com/peaktech-2710-digital-rcd-tester

> PeakTech 2715 Digital Loop / PSC Tester (SKU 19078)

www.elektor.com/peaktech-2715-digital-loop-psc-tester



CAUTION! HIGH VOLTAGE!

There are always safety risks associated with working on circuits or systems that are (potentially) connected to mains power. Elektor and the designer of this project accept no liability for any damage (in whatever form) caused by using this earth leakage tracer. As with all projects involving mains power, we offer the following sincere advice: If you don't know what you are doing, don't do it!

POVERTY

and Electronics

Sustainable Development Goal 1

By **Priscilla Haring-Kuipers** (The Netherlands)

Electronics touch poverty in many ways. Technologies can be a resource to help people out of poverty, or they can be the cause of it.

"To end poverty in all its forms everywhere" is the first of the Sustainable Development Goals (SDGs) defined by the United Nation. Electronics often have a supportive role in fighting poverty by providing communication, content or measurements in various projects such as micro-financing, enhancing farming methodology and basic health care. [1]

Poverty Targets

The first target is to end extreme poverty, which is determined as people living on less than \$1.25 a day. If you want to know what that looks like, go to Dollar Street [2] and take a look at families with an income up to \$38 per month. Sadly, because of Covid-19, for the first time in our generation poverty has risen. Effects of the global pandemic have pushed around 120 million people back into this bottom category during 2020.

Other targets include halving the amount of people living below the national poverty line, making sure poor people have access to basic services, can own property and to reduce exposure to climate change related extreme weather.



The Chowdhury family lives in India on \$30 per month. Their favorite household item is this mobile phone. (Source: Zorah Miller, Dollar Street 2015 [2], CC BY 4.0)

Deeper Issues

The worst way our electronics are intertwined with poverty is in the mining of raw materials. Cobalt mining in the Congo is mostly done under very harsh conditions while paying the workers — often children — a fee that keeps them living in (extreme) poverty. Gold and copper mines all over the world have “unhealthy working conditions” while providing very low wages. Mining is booming, but this economic benefit is not passed down to the hands that do the work. Several large manufacturers (Samsung, BASF, Tesla, etc.) have started initiatives to promote ethical mining, but these projects seem to be at the stage of good intentions.

We Decide

Many of the decisions made in High Income Countries determine the economic opportunities for people living in Low Income Countries. What sorts of regulations do we decide on for materials used in electronics? Do we design for repair? How do we process our e-waste? People in Low Income Countries can make use of the electronics we discard by repairing them and selling them as refurbished electronics within their own economies, creating value for themselves. But only if our designs allow. Or they can make a very unhealthy and underpaid living sorting our electronics as waste. Our design decisions determine their options.

Tech to the Rescue

Some believe that technology provides the way out of poverty: “Modern technology



The Jacques family lives in Haiti on \$39 a month. These are the spices they use for cooking. (Source: Zorah Miller, Dollar Street 2015 [2], CC BY 4.0)



The mobile revolution has allowed people living in poverty to access mobile banking and receive vital health information.

is one of the most effective solutions to poverty.” [3] The mobile revolution has allowed people living in poverty to access mobile banking and receive vital health information. Farmers can cooperate on crop prices and receive weather predictions. Mobiles also function as warning systems in case of natural disasters, which is especially important for people living in remote areas. Pump and filtering installations with solar power provide clean water and reliable energy. These technologies, as well as tech used in education and political collaboration, can support people to increase their resources and opportunities and find their way out of (extreme) poverty. ◀

210561-01

WEB LINKS

- [1] UN Department of Economic and Social Affairs, “SDG Good Practices - A Compilation of Success Stories and Lessons Learned in SDG Implementation,” First Edition, 2020.: <https://sdgs.un.org/publications/sdg-good-practices-2020>
- [2] Dollar Street: <https://www.gapminder.org/dollar-street>
- [3] The Borgen Project, “10 Technological Solutions to Poverty,” 2014.: <https://borgenproject.org/10-technological-solutions-poverty/>

The Elektor Store

Never expensive, always surprising

The Elektor Store has developed from the community store for Elektor's own products like books, magazines, kits and modules, into a mature webshop that offers great value for surprising electronics. We offer the products

that we ourselves are enthusiastic about or that we simply want to try out. If you have a nice suggestion, we are here (sale@elektor.com). Our main conditions:
never expensive, always surprising!

Creality HALOT-SKY Resin 3D Printer

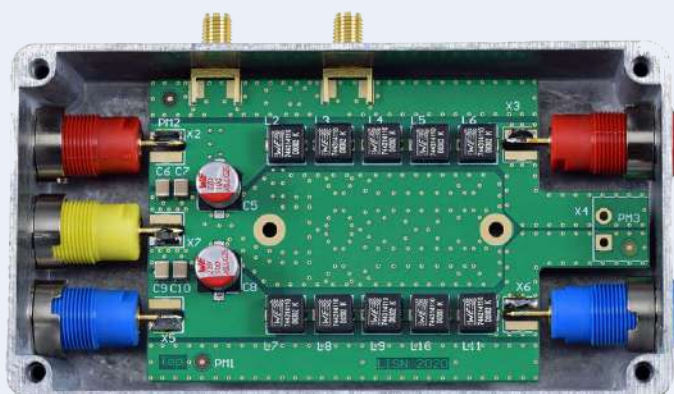
Price: €849.00

Member Price: €764.10

 www.elektor.com/19745



Elektor Dual DC LISN 150 kHz – 200 MHz



Price: €109.95

Member Price: €98.96

 www.elektor.com/19869



Cytron Maker Pi RP2040 – Robotics with Raspberry Pi RP2040

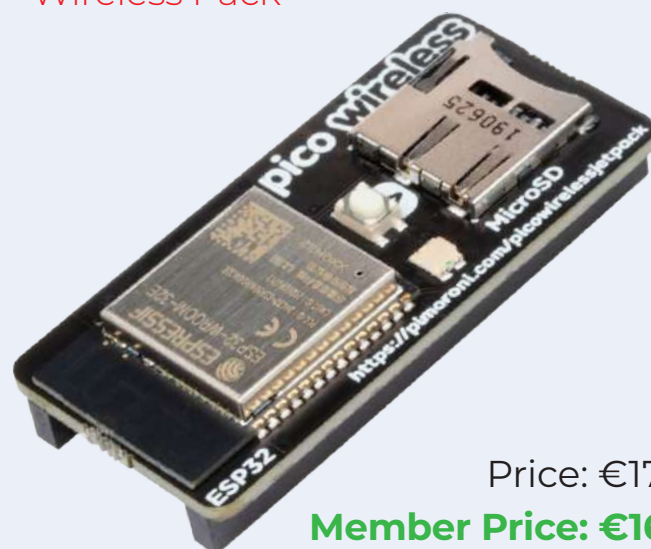


Price: €16.95

Member Price: €15.26

www.elektor.com/19926

Pimoroni Raspberry Pi Pico Wireless Pack



Price: €17.95

Member Price: €16.16

www.elektor.com/19916

MonkMakes Air Quality Kit for Raspberry Pi

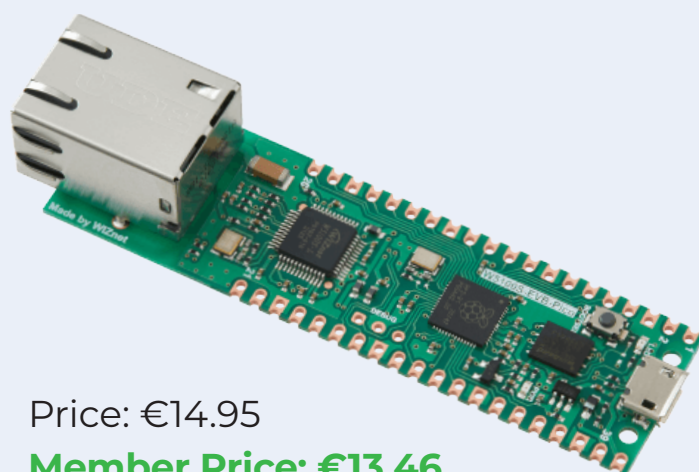


Price: €29.95

Member Price: €26.96

www.elektor.com/19913

WIZnet W5100S-EVB-Pico RP2040-based Evaluation Board



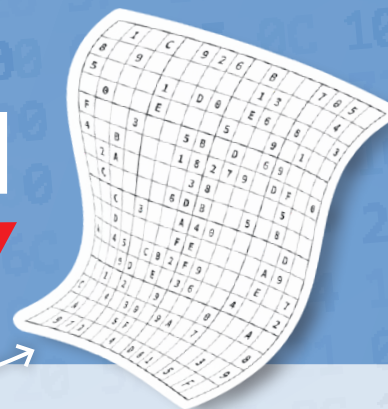
Price: €14.95

Member Price: €13.46

www.elektor.com/19971

Hexadoku

Puzzles with an Electronic Touch



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 store 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 store vouchers worth **€50.00 each**, which should encourage all Elektor readers to participate.

PARTICIPATE!

Ultimately February 15th, 2022, 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 11-12/2021 (November & December) is: **53974**.

Solutions submitted to us before December 17th were entered in a prize draw for 5 Elektor Store Vouchers.

The winners are posted at www.elektormagazine.com/hexadoku.

Congratulations everyone!

	4	7		3		B			5		0		6	8	
	1	D				0	2	B	7				A	4	
B	0		5				6	D				7		E	2
C	A		6		7	8	E	9	1	2		F		D	0
				1	8					5	9				
			A		3	7			C	B		8			
4		8	B				D	F				A	2		C
6	D	E											7	1	9
			3		A	F	9	C	4	0		D			
A						E			2						4
			2	8								A	C		
					5	6			9	D					
5	3	2	D	9	B	4			6	7	C	1	8	F	E
	B	A		0		5			3		E		D	C	
0			9	E							5	2			A
7	E	6		F	1	C			D	A	2		9	5	B

4	9	D	C	3	E	5	F	6	A	8	0	B	2	7	1
E	5	6	3	2	B	0	7	1	9	C	D	F	4	8	A
B	F	A	8	C	1	4	6	E	2	7	3	5	D	0	9
0	1	2	7	8	D	9	A	B	5	F	4	6	3	C	E
C	B	E	9	6	7	D	5	A	8	3	F	2	1	4	0
F	A	0	1	E	8	B	4	2	D	9	7	3	5	6	C
6	7	3	4	9	F	1	2	C	E	0	5	D	8	A	B
D	2	8	5	A	3	C	0	4	B	1	6	E	F	9	7
1	E	B	2	F	9	7	8	D	6	A	C	4	0	3	5
3	6	5	D	4	C	E	1	F	0	B	9	A	7	2	8
7	4	9	A	5	0	6	D	8	3	2	1	C	E	B	F
8	C	F	0	B	2	A	3	5	7	4	E	9	6	1	D
2	8	1	6	7	4	F	E	9	C	5	B	0	A	D	3
5	3	4	B	D	6	8	C	0	1	E	A	7	9	F	2
9	D	7	E	0	A	2	B	3	F	6	8	1	C	5	4
A	0	C	F	1	5	3	9	7	4	D	2	8	B	E	6

The competition is not open to employees of Elektor International Media, its subsidiaries, licensees and/or associated publishing houses.



PROTEUS DESIGN SUITE

Design Quality Assurance

Constraint Driven Design

Flexible and scalable
rule system

Full support for design
rule rooms

Manufacturing
solder mask rules

Live display of
violation areas

Zone Inspector

Analyze plane coverage and
stitching

Grid view of plane
configurations

Edit plane settings and
draw order

Pre-Production Checklist

Set of board tests
before Gerber Output

Includes placement,
connectivity and
clearance testing

Completely independent
code for clearance checks

Dedicated Reporting Module

Tables automatically
populate with design
data

Compliance status for
diff pairs and length
matched routes

Make custom
reports with data
object tables

Generate reports
from templates



Join the Elektor Community

Take out a



membership!



- ✓ The Elektor web archive from 1974!
- ✓ 6x Elektor magazine (Print)
- ✓ 9x Digital (PDF) including Elektor Industry (EN) magazine
- ✓ A 10% discount in our web shop and exclusive offers
- ✓ Elektor's annual DVD-ROM

- ✓ An online Elektor LABs account, with access to more than 1000 Gerber files and a direct line of communication with our experts!
- ✓ Bring a project to publication or even sell it in our shop

Also available

The Digital
membership!



- ✓ Access to Elektor's web archive
- ✓ 10% discount in our web shop
- ✓ 6x Elektor magazine (PDF)
- ✓ Exclusive offers
- ✓ Access to more than 1000 Gerber files



www.elektor.com/member