£9.95 | €10.95 | $12.95

# elektor MAG

design > share > earn

513
SINCE 1961

MAY & JUNE 2022
ELEKTORMAGAZINE.COM

## ESP32-C3 IoT Project

A Wi-Fi Button and Relay

p. 6

FOCUS ON

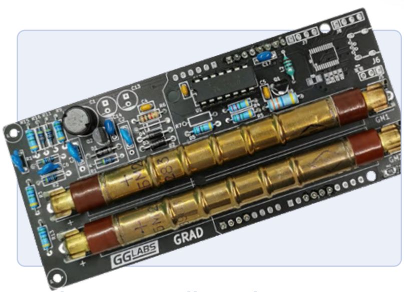
## Internet of Things

p. 54 **The Challenges in Bringing IoT Solutions to Market**
Worries Around Security, Scalability, and Competition

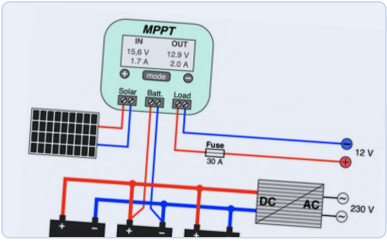
p. 12 **IoT Cloud à la Arduino**
Convenient Solution for Application Developers

p. 74 **Dragino LPS8 Indoor Gateway**
Speedy LoRaWAN Gateway Setup

**DIY $CO_2$ Guard**
Air Quality Monitoring Made Simple p. 24



**Dual Geiger-Müller Tube Arduino Shield** A High-Sensitivity Radiation Sensor p. 18



**Off-Grid Solar Systems**
Design Considerations for DIYers p. 104

# COLOPHON

Elektor Magazine,
English edition

Edition 3/2022
Volume 48, No. 513
May & June 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

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

**Memberships:**
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.

# EDITORIAL

## Jens Nickel

*International Editor-in-Chief, Elektor Magazine*

## Learning From Mistakes

Writing an editorial means always being a few weeks ahead of your publication date. By the time you read these lines, I very much hope you have already seen the first signs of peace in Ukraine. And perhaps the discussions about COVID-19 measures and high incidence rates will have died down.

What does the news (much of it bad) from around the world have to do with electronics? In the last few months, we have all had to realize that our world is highly interconnected and globalized. Prior to the recent geopolitical issues and the COVID-19 crisis, things seemed to be going well for a few years in row! Even large corporations like the international carmakers had become so accustomed to the comfortable situation that few had a Plan B in the drawer just in case a global catastrophe emerged.

Once the current crises have been overcome, many companies will certainly not make the same mistakes again. Increased product availability and shorter transport routes — such things will take on a higher priority than simply securing the lowest price. This will also be good for the electronics industry.

In many areas of electronics, the steady upswing has weakened, but things are still moving forward — for example, in the Internet of Things, which is the main topic for this issue. My colleague Robert van der Zwan has put together some interesting infographics about this on page 60. Of course, we have a variety of background articles and interesting projects for you as well!

We'll stay tuned.

### Farewell, Not Forgetting

The fact that our long-time employee Ralf Schmiedel passed away in March 2022 was a shock for me and my colleagues. Ralf had been in charge of Elektor's reader service since 2014. With great patience and many ideas of his own, the engineer took care of a wide variety of concerns from readers and authors, compiled corrections of projects, and helped our editorial team with other tasks such as the annual DVDs. Many thank-you emails from readers showed how valuable his support was. Ralf, we will all remember you fondly!

## 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 |

FIPP Connecting Global Media

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."

DEUTSCHE FACHPRESSE

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

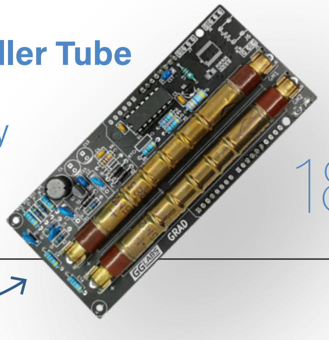
# THIS EDITION

Volume 48, No. 513
May & June 2022

## Regulars

## Features

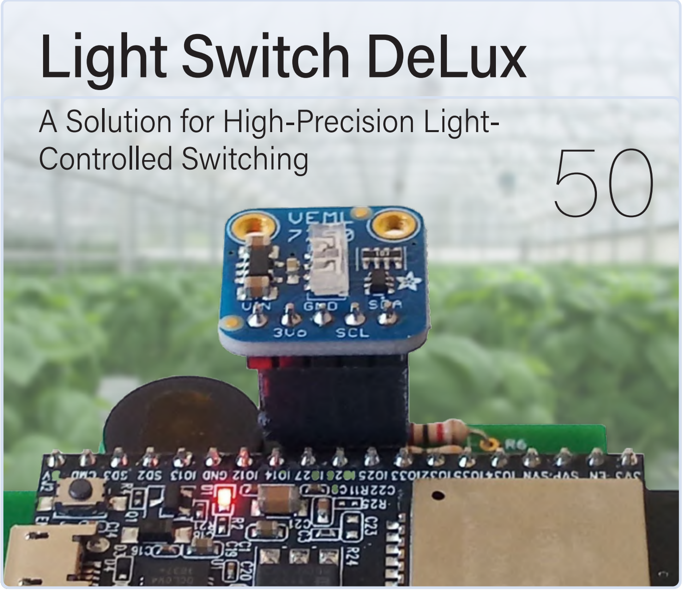
## Industry



Tips & Tricks for Testing Components

No Expensive Equipment Required

40



Light Switch DeLux

A Solution for High-Precision Light-Controlled Switching

50

## Projects

## Next Edition

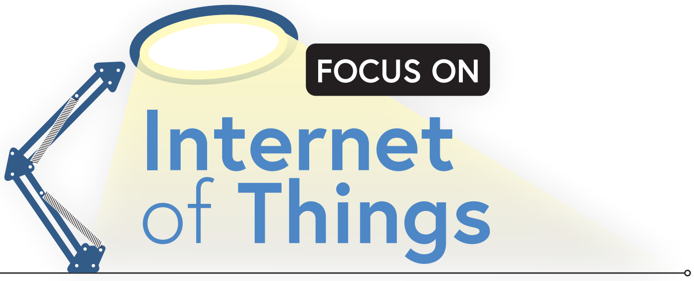
**Elektor Magazine Edition 7-8/2022 (July & August 2022)**
As usual, we'll have an exciting mix of projects, circuits, fundamentals and tips and tricks for electronics engineers and makers. We will focus on Test & Measurement.

**From the contents:**

- Autonomous Inductance Meter
- $CO_2$ Meter with Sigfox
- Design Tools for Analog Filters
- Smart Plugs: A Look Inside and Hacked
- Simple Analog ESR Meter
- Get Started With Your Oscilloscope
- Raspberry Pi Pico Makes an MSF-SDR
- AC Grid Frequency Meter

**And much more!**

Elektor Magazine edition 7-8/2022 covering July & August 2022 will be published around July 7, 2022. Arrival of printed copies with Elektor Gold Members is subject to transport. Contents and article titles subject to change.



# Your First Steps with an ESP32-C3 and the IoT

## A Wi-Fi Button and Relay

**By Mathias Claußen (Elektor)**

The IoT is not a closed book of hidden secrets. Powerful controllers like the new ESP32-C3 and newbie-friendly development environments like the Arduino IDE make developing small projects a piece of cake.

When we speak of the Internet of Things (IoT), we acknowledge that more and more things in our daily lives are becoming connected to the Internet. It starts with lights, heaters and sensors in the home and continues with cars, traffic lights, shipping containers and much more. Small network-capable components are installed in each of the connected things, which enable exchange of information.

The best way to get to know how to connect your own applications to the IoT is to start out with a simple practical example. In this article we create a link between a Wi-Fi enabled push button and a Wi-Fi enabled relay; the relay can be activated remotely by the button and reports its status back to the push button.

### Select the Components

As with all projects, the right components must first be selected. This is where the ESP-C3-12F kit (**Figure 1**), which is available in the Elektor Store, comes into play. This board features a Wi-Fi-enabled ESP32-C3 microcontroller from Espressif. The ESP32-C3 is a replacement for the proven ESP8266. In addition to a modern CPU core, the chip offers a good mix of integrated peripherals that are both beginner-friendly and powerful. (Refer to our review about the ESP32-C3 [1].) An overview of the integrated hardware blocks are depicted in **Figure 2**. In addition to the ESP32-C3, a RGB LED and a USB serial converter are also integrated on the board. We will need two ESP-C3-12F kits for our project.

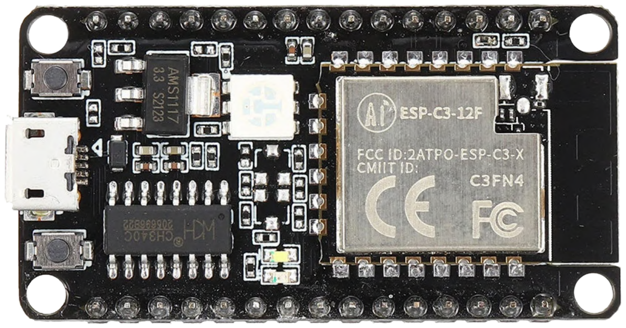


*Figure 1: The ESP32-C3-12F Kit.*

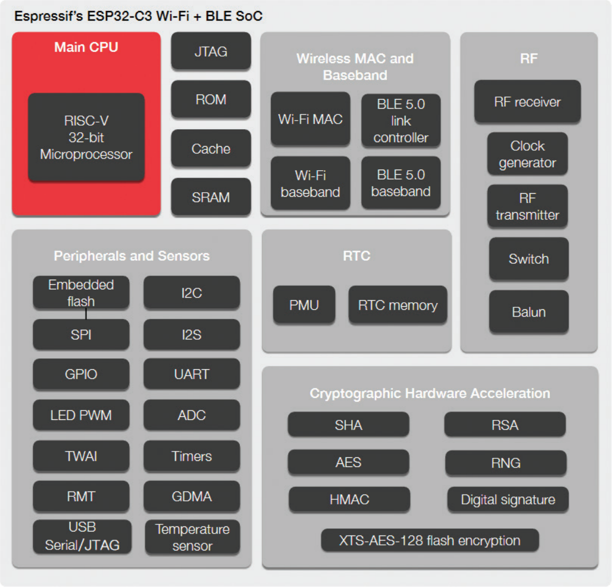


*Figure 2: Functional blocks of the ESP32-C3 (Source: ESP32-C3 datasheet).*

In addition to the ESP-C3-12F kits, a sensor and an actuator are also required. That's where the Elektor 37-in-1 sensor kit comes in handy, this kit includes 35 sensors (the original version had 37, but two of them contained mercury and have since been omitted on safety grounds). An overview of the sensors of the kit (**Figure 3**) can be found in **Figure 4** and the information document [2]. First of all, we will take the Joystick module and use its push button feature to provide control input to the system. The relay module can now connected to the other microcontroller board and acts as the actuator in the system. A few (female/female) flying leads are required to connect the modules. These are included in the Pimoroni "Mini Breadboards & Jumpers" maker kit. (See the **Related Products** box.)

We also need a computer, such as a Raspberry Pi, which will act as a local server for the IoT devices to exchange their data. An original Raspberry Pi version 1 would in fact do the job, but we recommend at least a Raspberry Pi 2 for this application. On-board Wi-Fi was not included until Raspberry Pi model 3B, so to use earlier versions, a simple Wi-Fi dongle or Ethernet cable will also be required. If you need to buy a small but powerful Raspberry Pi, take a look at the Raspberry Pi Zero 2 W bundle (box). For our purposes, it doesn't however need to be a Raspberry Pi. Any PC running a Linux distribution such as Ubuntu [3] will also prove perfectly adequate.

Before we get into the project, let's take a closer look at how the control and exchange of data takes place in this setup.

## MQTT

Any IoT device, whether it's a sensor or actuator, needs to transfer data. For this purpose, we can either go the long way around by developing our own proprietary communication protocol or we can use standard, established protocols. One such system which has become widespread is MQTT. Originally, it was an acronym for "Message Queuing Telemetry Transport," but as the system developed, the title no longer accurately described its function. In 2013, it was officially decided that MQTT would be a label [4].

The MQTT protocol takes care of the exchange of messages using a broker (server) without specifying what the messages look like. You can compare this to sending a letter: the logistics and the format of

*Figure 3: The Elektor 37-in-1 sensor kit.*

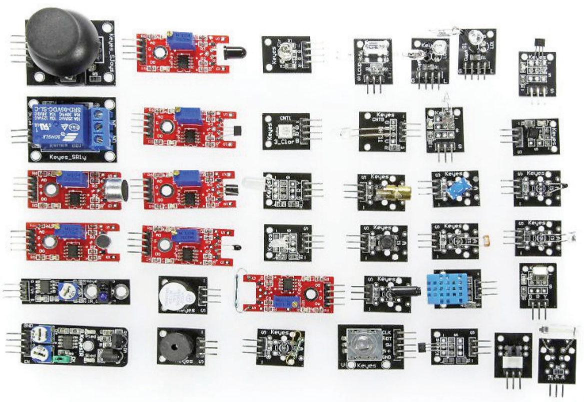
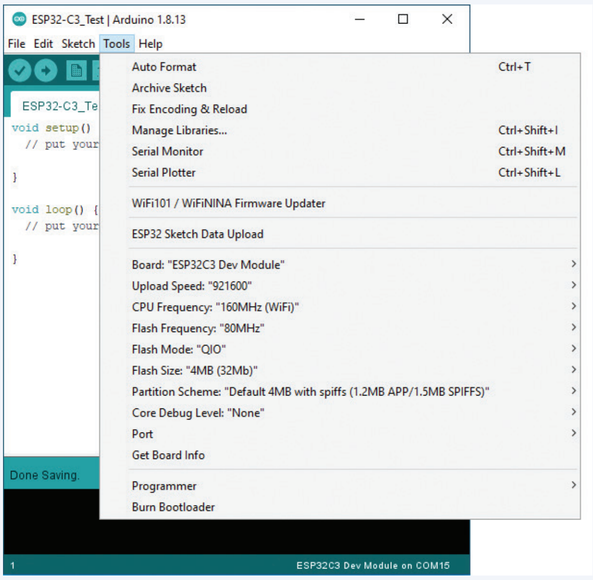


*Figure 4: A whole bunch of sensors and actuators are included.*


ESP32-C3_Test | Arduino 1.8.13
File Edit Sketch Tools Help
ESP32-C3_Te
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
ESP32 Sketch Data Upload
Port
Get Board Info
Programmer
Burn Bootloader
Done Saving.
1
ESP32C3 Dev Module on COM15

```
void setup() {
  // put your

}

void loop() {
  // put your

}
```

Board: "ESP32C3 Dev Module"
Upload Speed: "921600"
CPU Frequency: "160MHz (WiFi)"
Flash Frequency: "80MHz"
Flash Mode: "QIO"
Flash Size: "4MB (32Mb)"
Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"
Core Debug Level: "None"

*Figure 5: Configuration of the ESP32-C3 in the Arduino IDE.*

the envelope are specified by the postal company, but the message in the envelope and the language in which it is written is entirely up to the user.

Which "language" should we choose to send our messages? Again, there are several options. One popular choice (not only used for MQTT) is Java Script Object Notation (JSON).

## JSON

JSON is a lightweight data interchange format for message transfer that is easily generated and interpreted even by small microcontrollers. In addition, JSON text is not only easy to understand for humans, but also easy to write. An overview of the JSON specification can be found on the JSON standard website [5]. JSON is not only used in connection with MQTT, but also in many other areas. JavaScript, the programming language from which JSON is derived, is one of the core technologies on which the World Wide Web is based today. A good introduction to JSON with practical examples can be found on the Mozilla [6] site.

## Setting Up the IoT Environment: The MQTT Broker

As with all projects, proper preparation helps avoid unexpected surprises later. To handle MQTT messages, we require a broker device that can either exist on the Internet or we can install one locally on our network. A local broker means we won't need to rely on cloud services for IoT functionality that for our purposes will only be used to transfer messages between locally connected devices. The broker can be built using a "retired" PC or Raspberry Pi, for example. Using Node-RED, we get a complete toolbox for developing network applications that is not just limited to processing MQTT messages. Node-RED [7] has already been used frequently at Elektor to process MQTT data and can be installed quickly on a Raspberry Pi [8] or a PC [9] thanks to the detailed instructions.

## The Arduino IDE

The Arduino IDE is used here as the development environment. The IDE's editor is not the best in its class, but currently offers the most stable support for the ESP32-C3. The Arduino IDE [10] can be downloaded and installed free of charge from the Arduino homepage. The Arduino ESP32 support (as described in the Espressif documentation [11]) must now be installed. The settings for the board are made as shown in **Figure 5**.

In addition to the Arduino ESP32 support, a few libraries will also be required for our first steps. In our example for the ESP32-C3 to be able to send data via MQTT/JSON, we will need to install Nick O'Leary's *PubSubClient* and Benoit Blanchon's *ArduinoJson* libraries. These can be installed using the Arduino IDE Library Manager (**Figures 6** and **7**).

## Assembling the Hardware

The two ESP32-C3 based modules are assembled according to the circuit diagrams in **Figure 8** and **Figure 9**. The joystick and relay modules require just three wires connected to the respective ESP32-C3

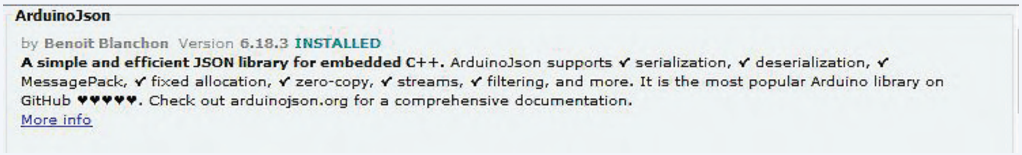

**ArduinoJson**
by Benoit Blanchon Version 6.18.3 INSTALLED
**A simple and efficient JSON library for embedded C++.** ArduinoJson supports ✓ serialization, ✓ deserialization, ✓ MessagePack, ✓ fixed allocation, ✓ zero-copy, ✓ streams, ✓ filtering, and more. It is the most popular Arduino library on GitHub ♥♥♥♥♥. Check out arduinojson.org for a comprehensive documentation.
More info

*Figure 6: PubSubClient in the Arduino Library Manager.*

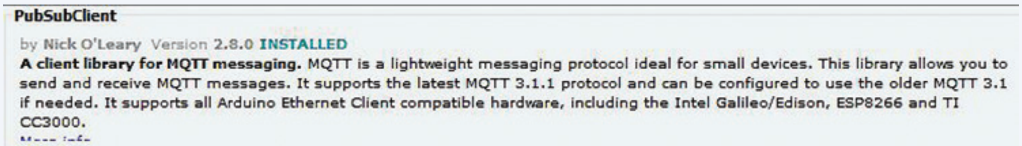

**PubSubClient**
by Nick O'Leary Version 2.8.0 INSTALLED
**A client library for MQTT messaging.** MQTT is a lightweight messaging protocol ideal for small devices. This library allows you to send and receive MQTT messages. It supports the latest MQTT 3.1.1 protocol and can be configured to use the older MQTT 3.1 if needed. It supports all Arduino Ethernet Client compatible hardware, including the Intel Galileo/Edison, ESP8266 and TI CC3000.
More info

*Figure 7: ArduinoJson in the Arduino Library Manager.*

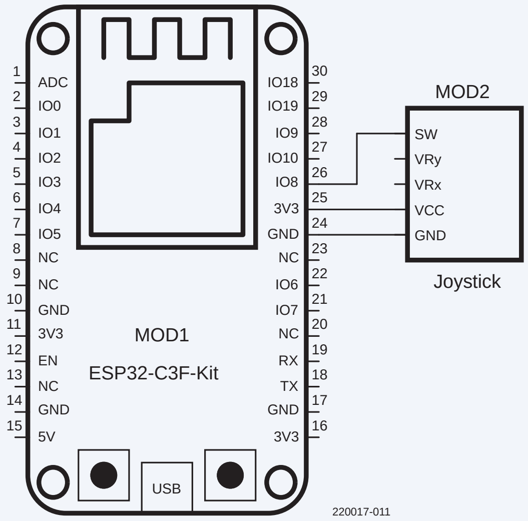


*Figure 8: Circuit diagram of the ESP32-C3 and Joystick.*

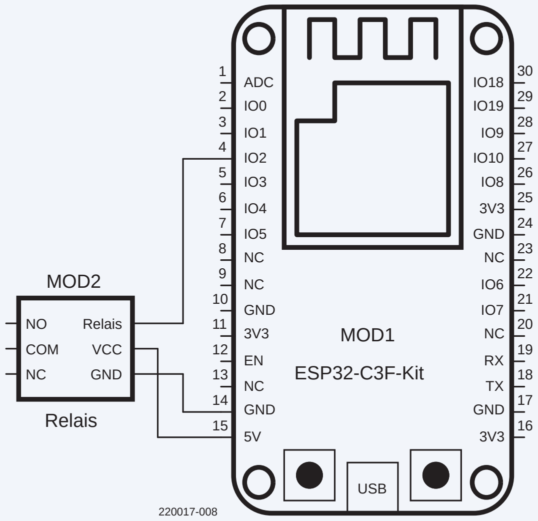


*Figure 9: Circuit diagram of the ESP32-C3 and Relay.*

board. Note the relay board is powered from the 5 V supply pin. **Figure 10** shows all the modules and controllers connected together.

## Software Setup

The source code for this project is also available on GitHub [12]. The sketches for the two ESP32-C3 controllers can be downloaded from there. It will be necessary to enter some information about your local network to these files before we can upload them to the ESP32s. Everything must be set appropriately so that the two controllers can exchange data with the local MQTT broker. For this purpose, `#define` directives are present at the beginning of the two Arduino sketches:

```
#define WIFI_SSID "changeme"
#define WIFI_PASS "changeme"
#define MQTT_SERVER "test.mosquitto.org"
```

These three `#defines` at the beginning of the Arduino sketch must be adjusted for your own network. The SSID and PASSWORD of your network need to be entered in the appropriate position between the quotation marks. The IP address of the Node-RED computer in its own network is specified for the MQTT server. Once both sketches (for the relay and for the button) have been edited they can be uploaded to the respective ESP32-C3. Both ESP32s can then be powered up and the large LED on each board should start flashing white. This indicates that the ESP32-C3 is attempting to connect to the Wi-Fi network. The LED should then light continuously when a board successfully connects. The LED colour will depend on the board's function: the board with the relay connected will light up white. The board with the pushbutton connected will light up red (relay off) or green (relay on) — i.e., according to the relay status.

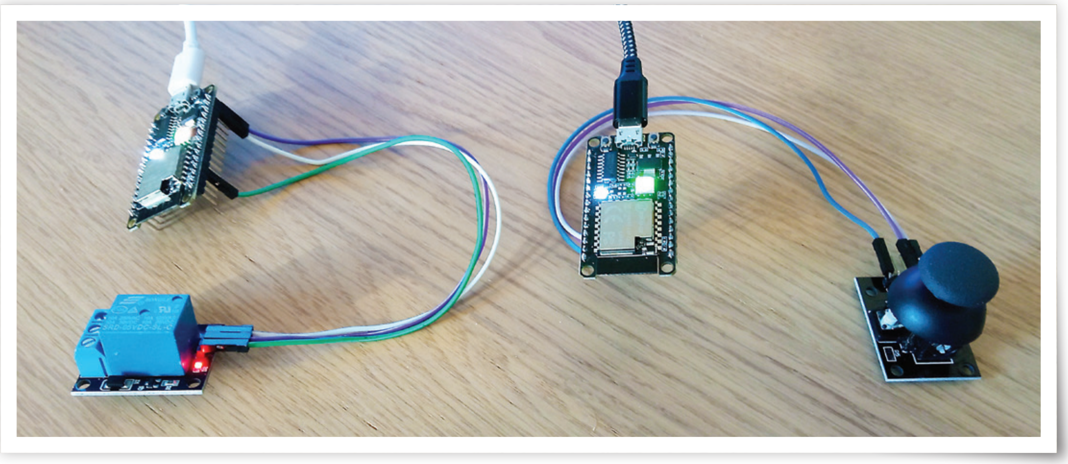


*Figure 10: The complete hardware hook up.*

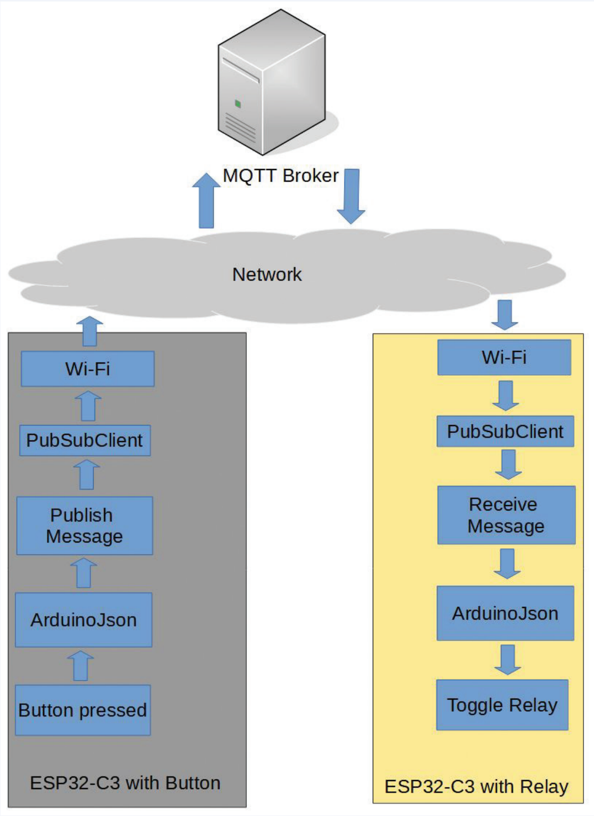


*Figure 11: Data transfer to the Broker and back.*

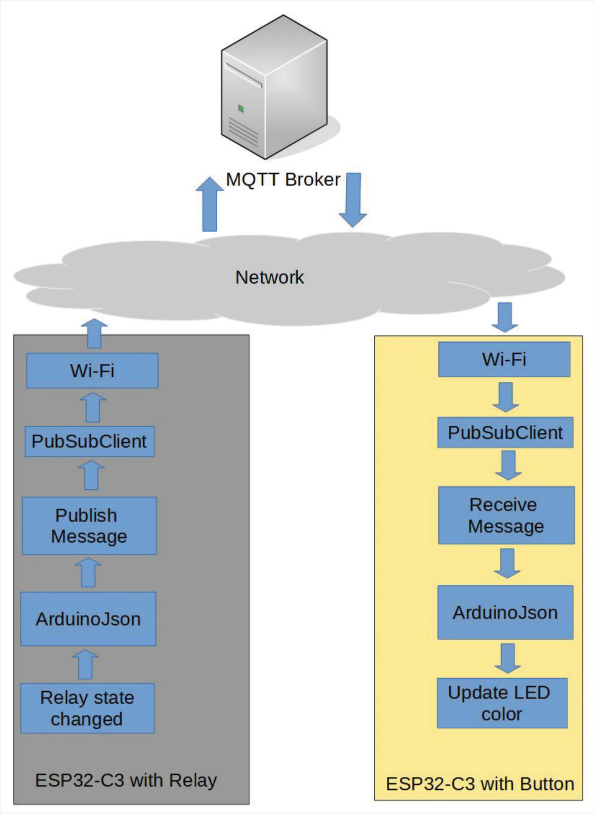


*Figure 12: Feedback from the Relay.*

This now shows that both ESP32-C3s are operating successfully. Pressing the button will change the state of the relay and the colour of the LED will change from red to green or vice versa. The button of one ESP32-C3 can thus successfully control the relay of the other ESP32-C3, and it also receives feedback on the status of the relay. Time to celebrate. Your first IoT application is running! But how exactly does the data exchange work?

## To the Relay and Back Again

First, let's look at the path a button press takes to the relay. **Figure 11** shows how the message is packed layer by layer and then sent to the broker via Wi-Fi. In the source code, this is done using `client.publish(MQTT_TOPIC_OUT, (const uint8_t*)buffer, n, true);`.

Why is this function called `publish` and not simply `send`? This is due to the way the data is later distributed in MQTT. On an MQTT broker, messages are distributed based on a topic; in this case, the topic (`MQTT_TOPIC_OUT`) *"BUTTON"*. When connecting to the MQTT broker, the client (i.e., the ESP32-C3 of our relay) can indicate which topic is of interest (i.e., to "subscribe" to this news channel).

Each participant who has informed the MQTT broker that they are interested in a certain topic will receive the messages sent under this topic. The sender, on the other hand, does not have to worry about distribution. It only sends ("publishes") its messages to the broker.

The ESP32-C3 with the relay subscribes to the *BUTTON* topic in its code using `client.subscribe(MQTT_TOPIC_IN);` , where `MQTT_TOPIC_IN` is "*BUTTON*" here. Every time a message is triggered by the button, it goes to the MQTT broker. It is then delivered to the ESP32-C3 with the relay, causing the relay to toggle.

When the relay changes it state, its controller sends a message to the MQTT broker under the topic *"RELAIS" (RELAY)*, in which the current state (whether on or off) is included. The ESP32-C3 with button, in turn, has subscribed to the *"RELAIS"* topic at the MQTT broker and thereby receives the message with the new status, as shown in **Figure 12**. This causes the LED colour to be set to red or green.

The nice thing about this structure is that a second button and controller can also be integrated into the network to send messages under the topic *"BUTTON"* — just like the first button. The ESP32-C3 with the relay will then respond to the control data from either button and carry out the appropriate switching operation. If you're interested in experimenting further with MQTT, then take a look at Elektor's other projects that use MQTT to send data. Examples are the weather station [13] and the monster LED clock with an external temperature sensor [14]. For more about MQTT and how you can use it to send data to cloud platforms, for example, refer to the series, "My Journey into The Cloud" [15].

## An Ideal Platform for the IoT

IoT does not need to be complicated. As we have shown here, the technology is quite mature, and you can develop IoT devices quickly using simple tools and the proper components. Applications can of course be more complex than just a simple push button and relay. From domestic heating control to the front doorbell, there are many practical applications that can benefit from IoT connectivity. The ESP32-C3 is a powerful and inexpensive board that makes an ideal platform to develop your own ideas!

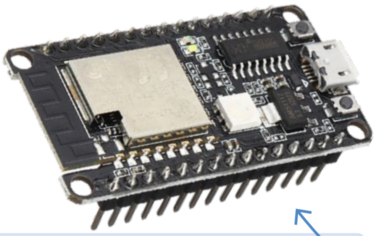
220017-01

### Contributors

Idea and Text: **Mathias Claußen**
Editor: **Jens Nickel**
Translator: **Martin Cooke**
Layout: **Giel Dols**

### 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.



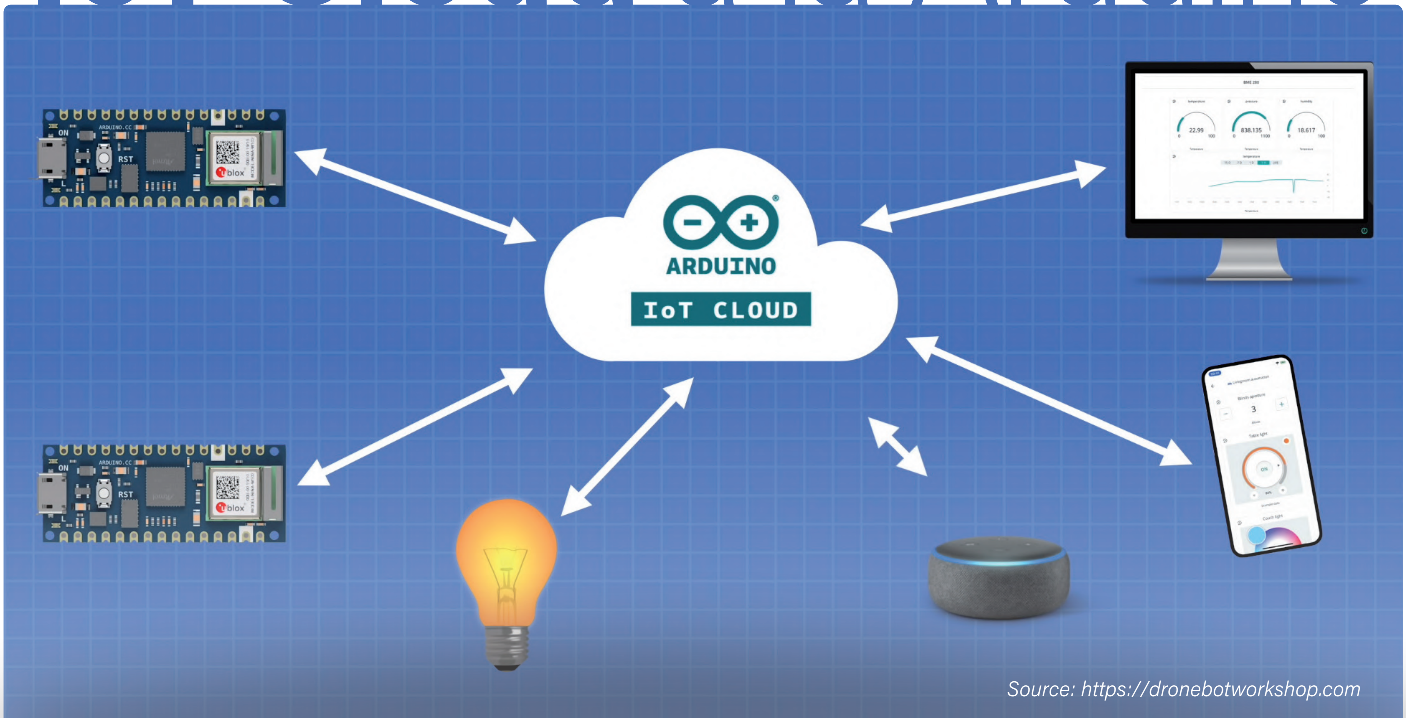
### RELATED PRODUCTS

- **ESP-C3-12F-Kit Development Board with 4 MB Flash (SKU 19855)**
  www.elektor.com/19855
- **Elektor 37-in-1 Sensor kit (SKU 16843)**
  www.elektor.com/16843
- **Raspberry Pi Zero 2 W Bundle (SKU 19952)**
  www.elektor.com/19952
- **Pimoroni Maker Essentials – Mini-Breadboards & Jumper cables (SKU 18430)**
  www.elektor.com/18430

### WEB LINKS

[1] M. Claußen, "Getting Started with the ESP32-C3 RISC-V MCU," Elektor 1-2/2022: www.elektormagazine.com/210466-01
[2] Elektor 37-in-1 Sensor-Kit Documentation: www.elektor.com/amfile/file/download/file/1170/product/6171/
[3] Ubuntu Linux Distribution: https://ubuntu.com/
[4] OASIS MQTT TC Minutes from 25.04.2013: www.oasis-open.org/committees/download.php/49028/OASIS_MQTT_TC_minutes_25042013.pdf
[5] JSON.org: www.json.org/json-de.html
[6] Mozilla Web Docs: Working with JSON: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON
[7] Node-RED: https://nodered.org/
[8] Node-RED Installation on the Raspberry Pi: https://nodered.org/docs/getting-started/raspberrypi
[9] Node-RED Installation on a PC: https://nodered.org/docs/getting-started/local
[10] Arduino IDE download: www.arduino.cc/en/software
[11] Espressif Arduino-ESP32 Installation instructions: https://docs.espressif.com/projects/arduino-esp32/en/latest/installing.html
[12] Elektor GitHub Repository: https://github.com/ElektorLabs/220017-ESP32-C3-and-IoT_First-steps
[13] R. Aarts, "ESP32 Weather Station," Elektor 1-2/2019: www.elektormagazine.com/magazine/elektor-70/42351
[14] M. Claußen, "A Monster LED Clock with Wi-Fi and Temperature Display," Elektor 5-6/2019: www.elektormagazine.com/magazine/elektor-96/42659
[15] J. Nickel, "My Journey into the Cloud," Elektormagazine.com: www.elektormagazine.com/search?query=My+journey+into+the+cloud

# IoT Cloud a la Arduino



Source: https://dronebotworkshop.com

By Tam Hanna (Slovakia)

The Arduino IoT Cloud offers IoT application developers a convenient solution for implementing a cloud back end without having to struggle with MQTT. Curious? Let's take a look.

Lots of microcontroller applications nowadays involve an Internet of Things (IoT) application where information is disseminated via IoT cloud services and an MQTT broker. Creating this type of application using a local development environment like the traditional Arduino IDE can sometimes be awkward. The Arduino Cloud shifts the IDE up into the cloud so that your browser becomes a window into the IDE. We tried it out by sending a variable value to the cloud to make an LED flash on the bench. Then we tried to break it.

The basis of all IoT devices is of course the *Thing*. In the Arduino IoT Cloud development environment, the *Thing* is a virtual object that exists in the cloud. In the real world, it is as an object such as a server, a controller board, or a similarly "intelligent" device [1]. Here your *Thing* is built in the cloud using an online editor to write a Sketch describing how it should behave and respond by using a whole range of *Variables*.

## Who Will Use the Arduino IoT Cloud?

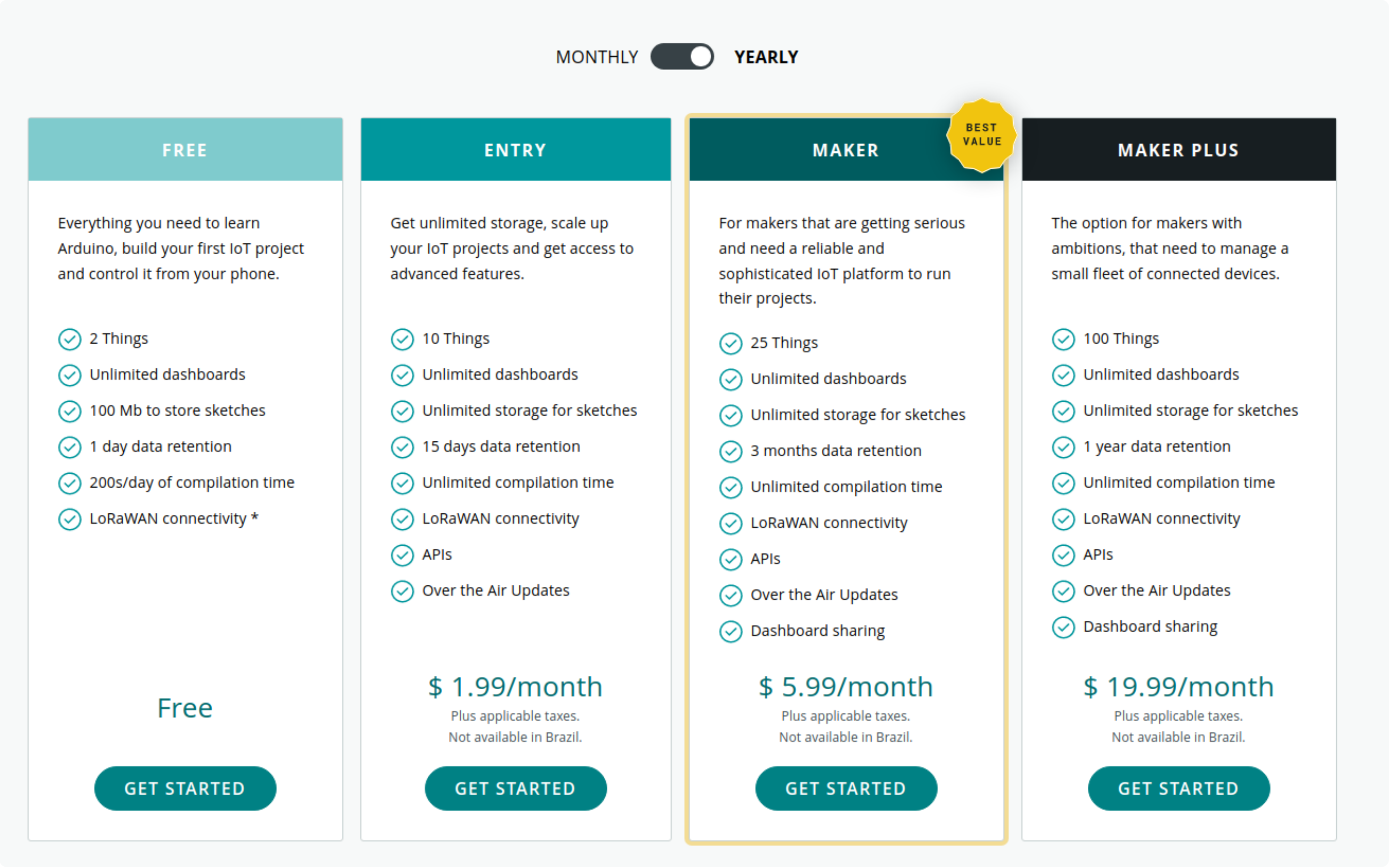
Before we begin, it's important to acknowledge that the Arduino IoT Cloud is not an alternative for other dedicated cloud computing platforms such as Amazon AWS IoT Core, Microsoft IoT Hub, or Yandex IoT Core. If you have large numbers of devices and data to manage, these more established IoT cloud services are the way to go.

At the introduction of the latest incarnation of The Arduino IoT cloud, Massimo Banzi, CTO at Arduino, expressed his ambitions for the platform saying that: "Arduino now offers a complete platform with the MKR family; providing a streamline way to create local IoT nodes and edge devices. These use a range of connectivity options and compatibility with third-party hardware, gateway and cloud systems. The Arduino IoT Cloud allows users to manage, configure and connect, not only Arduino hardware but also the vast majority of Linux-based devices — truly democratising IoT development."

Its support of the MKR IoT-targeted range of Arduino boards and some other popular third-party boards is a welcome addition and would make many IoT system developers give this accessible development platform a second look.

## Setting Up the Hardware

Only the most basic plan of the four possible versions of the Arduino Cloud is free to use. You can check out the various plans and their features in **Figure 1** and decide which package best meets your needs. Of interest to the maker community generally is Arduino Cloud's support of third-party boards, such as the popular ESP8266 and ESP32 family of devices (**Table 1**), and the list of compatible platforms [2]. Driver libraries also allow various Linux-based systems to upload and download information to the Arduino cloud.



MONTHLY ● YEARLY

| FREE | ENTRY | MAKER | MAKER PLUS |
|---|---|---|---|
| | | BEST VALUE | |
| Everything you need to learn Arduino, build your first IoT project and control it from your phone. | Get unlimited storage, scale up your IoT projects and get access to advanced features. | For makers that are getting serious and need a reliable and sophisticated IoT platform to run their projects. | The option for makers with ambitions, that need to manage a small fleet of connected devices. |
| ✓ 2 Things<br>✓ Unlimited dashboards<br>✓ 100 Mb to store sketches<br>✓ 1 day data retention<br>✓ 200s/day of compilation time<br>✓ LoRaWAN connectivity * | ✓ 10 Things<br>✓ Unlimited dashboards<br>✓ Unlimited storage for sketches<br>✓ 15 days data retention<br>✓ Unlimited compilation time<br>✓ LoRaWAN connectivity<br>✓ APIs<br>✓ Over the Air Updates | ✓ 25 Things<br>✓ Unlimited dashboards<br>✓ Unlimited storage for sketches<br>✓ 3 months data retention<br>✓ Unlimited compilation time<br>✓ LoRaWAN connectivity<br>✓ APIs<br>✓ Over the Air Updates<br>✓ Dashboard sharing | ✓ 100 Things<br>✓ Unlimited dashboards<br>✓ Unlimited storage for sketches<br>✓ 1 year data retention<br>✓ Unlimited compilation time<br>✓ LoRaWAN connectivity<br>✓ APIs<br>✓ Over the Air Updates<br>✓ Dashboard sharing |
| Free | $ 1.99/month<br>Plus applicable taxes.<br>Not available in Brazil. | $ 5.99/month<br>Plus applicable taxes.<br>Not available in Brazil. | $ 19.99/month<br>Plus applicable taxes.<br>Not available in Brazil. |
| GET STARTED | GET STARTED | GET STARTED | GET STARTED |

*Figure 1: Which plan is best for you? (Plan pricing as of 1/20/2022.)*

An Arduino Nano RP2040 Connect module is used here as the target board to test the Arduino Cloud functions. This board is based on the RP2040 microcontroller from the Raspberry Pi Foundation and includes a u-blox Wi-Fi module. Before setting up the Arduino cloud, the board is connected to a computer via a USB cable.

First, we need to visit the website [3] and apply for a new Arduino account. For our purposes, we will stick to the basic free plan which is explicitly aimed at newcomers to the system. First, we click on the *Create Thing* button to create a new thing.

The overview in **Figure 2** now shows the configuration of the three basic components of the development environment. I have removed a previous Arduino account and all configuration parameters so that I start with a fresh device setup. The following steps are performed on a machine running Windows 10, but the process is identical on the Ubuntu Linux environment and I have found that hardware detection usually works better under Unix.

Now we can click on the shortcut icon in the *Device* section and choose to *Set Up an Arduino Device*. A few seconds after clicking this option, the back end will point out that the component called *Arduino Create Agent* is missing. Click the Download button to download the software and install it in the normal way.

Note that *Create Agent* is browser-specific: if you install under Chrome, you will need to reinstall. Any firewall warnings that pop up should be acknowledged. Make sure that you allow access to

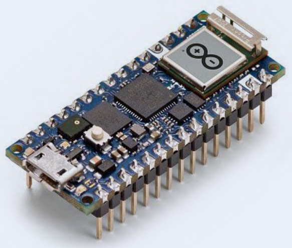
**Table 1: Boards supported by the Arduino Cloud**

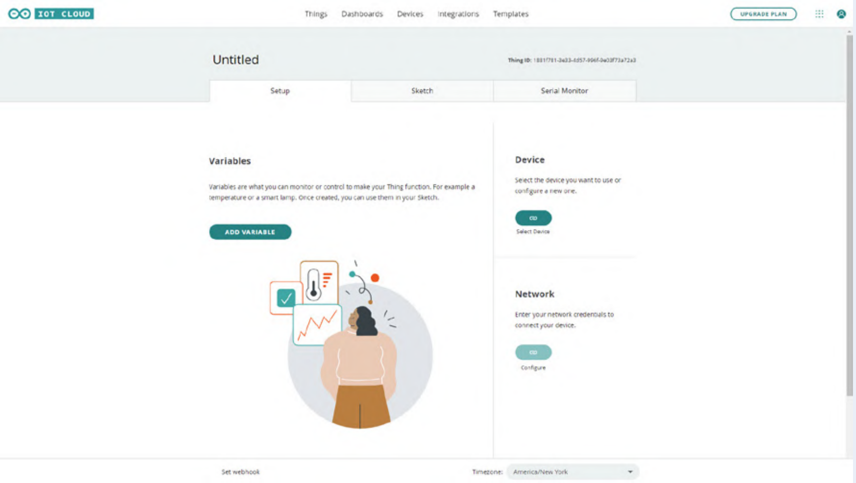
**WLAN**
- MKR 1000 WiFi
- MKR WiFi 1010
- Nano RP2040 Connect
- Nano 33 IoT
- Portenta H7

**LoRaWAN**
- MKR WAN 1300
- MKR WAN 1310

**GSM/NB-IoT**
- MKR GSM 1400
- MKR NB 1500

**ESP32/ESP8266**
- wide range of third-party boards





Untitled

Thing ID: 1881f781-3a33-4d57-996f-0e03f73a72a3

Variables

Variables are what you can monitor or control to make your Thing function. For example a temperature or a smart lamp. Once created, you can use them in your Sketch.

Device

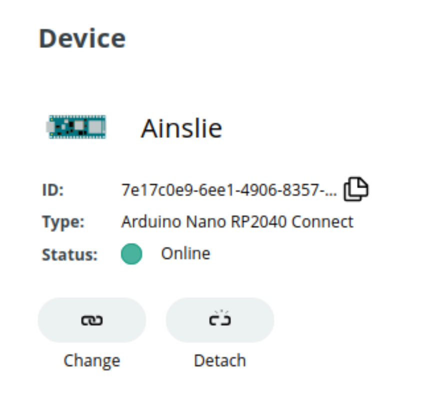
Select the device you want to use or configure a new one.

Network

Enter your network credentials to connect your device.

Timezone: America/New York

Figure 2: The IoT Cloud leads the developer step by step through to the goal.



Device

Ainslie

ID: 7e17c0e9-6ee1-4906-8357-...

Type: Arduino Nano RP2040 Connect

Status: ● Online

Figure 3: This Arduino is connected to the Cloud.

private and public networks alike. As a result, the *Arduino Create Agent* is now resident in your taskbar — in some cases it may need to be called up again from the start menu but that now completes the driver installation.

In the next step, we will refresh the view until the Arduino cloud informs us that our Nano RP2040 Connect board has been recognized. Next, click the *Configure* button to launch the configuration wizard — it will ask you to enter a 'friendly" name and then initialize the target system's secure element with the basic communications software.

Problems sometimes arise during network provisioning under Windows. The more reliable and successful method is to do this using Linux instead. Incidentally, the *Network* section is not automatically enabled by the Arduino cloud. It is only available when you create one of the variables intended for data exchange.

We can now click on *Variables*, to open a dialog and add a new variable. First we assign the name `ledIntenBool` and assign *Boolean* in the data type field. Funnily enough, the Arduino cloud not only supports C programming units, but also implements wrappers around real world variables.

If you want to limit yourself to C only, we recommend selecting the *Basic Types* option. Theoretically, we could then make settings in the *Variable Permission* and *Variable Update Policy* fields, but the default settings will be sufficient for our purposes, which is why we now close the dialog. In the next step, we then create a field of the *Integer Number* type, to which we give the name `ledIntenInt`.

## Preparing the Code

After creating the variables, red dots in the Sketch tab, indicate changes to the program structure. Now it is possible to click on the shortcut icon in the *Network* section to enter the Wi-Fi settings. My preference is to enter the values using a command line tool such as *iwlist* on a Linux machine and then copy them to the clipboard.

In the next step, we switch to the *Sketch* tab and click on the *Verify and Upload* button. The Arduino Cloud then starts compiling the code and sends it to the connected RP2040 using the *Arduino Cloud Agent*. If the compiled code is delivered successfully, the message *"Untitled_dec25a uploaded successfully on board Arduino Nano RP2040 Connect (/dev/ttyACM0)"* is displayed.

Now after the obligatory Reset, the RP2040 will start phoning home using its Wi-Fi transmitter. After a while and pressing *F5* several times the device will appear with "*Status: online*" as shown in **Figure 3**.

If you chose to subscribe to one of the more comprehensive Arduino Cloud plans, you can receive software updates directly via Wi-Fi; but for our simple experiments using the free plan, a wired connection is necessary.

## A Closer Look

The basic editor tool of the Sketch tab is not so useful, but by clicking on the *Open full editor* button, we get a fully-fledged cloud-based IDE that allows us to edit smaller projects more comfortably. First, let's look at the contents of the *thingProperties.h* file, which contains structural elements of the sketch.

First, we see the following declarations that provide the elements required for Wi-Fi access:

```
const char SSID[] = SECRET_SSID; // Network SSID (name)
const char PASS[] = SECRET_PASS; // Network password
// (use for WPA, or use as key for WEP)
```

The Arduino cloud takes care of entering the name and password using the settings already entered in the *Network* section. The next part contains the following two variables the names of which should be familiar from their declaration in the *Variables* section:

```
int ledIntenInt;
bool ledIntenBool;
```

The Arduino cloud implements the variables "in the back end" as standard C variables equipped with additional properties. These properties can be found, among other things, in the `initProperties`

Method, which takes care of setting up the primitives and structures required for cloud communication according to the following scheme:

```
void initProperties() {
ArduinoCloud.setThingId(THING_ID);
ArduinoCloud.addProperty(ledIntenInt, READWRITE,
ON_CHANGE, onLedIntenIntChange);
ArduinoCloud.addProperty(ledIntenBool, READWRITE,
  ON_CHANGE, onLedIntenBoolChange);
}
```

It is interesting here that the addProperty method, takes care of "registering" the attribute. Note the passing of the onLedIntenIntChange and onLedIntenBoolChange function pointers - they will play an important role later on.

The application control function is described in the sketch, which begins with the inclusion of the header (not shown here). This is then followed by the sketch initialization, which runs according to the following scheme:

```
void setup() {
 Serial.begin(9600);
 delay(1500);

 initProperties();

 ArduinoCloud.begin(ArduinoIoTPreferredConnection);
 setDebugMessageLevel(2);
 ArduinoCloud.printDebugInfo();
}
```

From the point of view of the Arduino programming environment, the Arduino cloud is a hardware driver like any other. The ArduinoCloud global object exposes a set of functions that your code uses to communicate with the cloud driver. Of particular importance here is the call to setDebugMessageLevel, which sets the "verbosity" of the driver — the higher the value, the more debug information the cloud driver outputs over the board's serial port.

In the case of "complicated" drivers, the question always arises as to how the processing power is allocated. The project skeleton created for us by the Arduino Cloud can answer this question in the loop method, which takes care of the allocation of computing power according to the following scheme:

```
void loop() {
 ArduinoCloud.update();
}
```

The Arduino cloud gives us the following three listener methods by default:

```
void onTestScheduleChange() {
}
void onLedIntenBoolChange() {
}
void onLedIntenIntChange() {
}
```

onLedIntenBoolChange and onLedIntenIntChange are responsible for the variables created remotely in the back end, while onTestScheduleChange helps implement "internal" functions of the Arduino cloud.

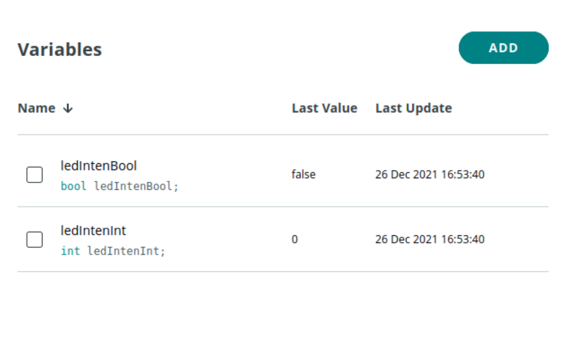
In the next step, we can take care of this by making use of the built-in "intelligent" capabilities of the Cloud. The Arduino board we are using here is fitted with a standard (red) LED on Pin 13 and an RGB LED (channels LEDR, LEDG and LEDB), which can be driven by three PWM signals to change the overall colour output.

Now we can return to the setup function, and initialise the necessary pins:

```
void setup() {
. . .
 ArduinoCloud.printDebugInfo();

 pinMode(LED_BUILTIN, OUTPUT);
 pinMode(LEDB, OUTPUT);
}
```

Changes triggered in the cloud activate the listener, which writes the incoming values to the hardware:

```
void onLedIntenBoolChange() {
 digitalWrite(LED_BUILTIN, ledIntenBool);
}
void onLedIntenIntChange() {
 analogWrite(LEDB, ledIntenInt);
}
```



Variables

| | Name ↓ | Last Value | Last Update |
|---|---|---|---|
| □ | ledIntenBool<br>bool ledIntenBool; | false | 26 Dec 2021 16:53:40 |
| □ | ledIntenInt<br>int ledIntenInt; | 0 | 26 Dec 2021 16:53:40 |

*Figure 4: The Arduino IoT cloud provides detailed information about the current state of the information stored in a variable.*

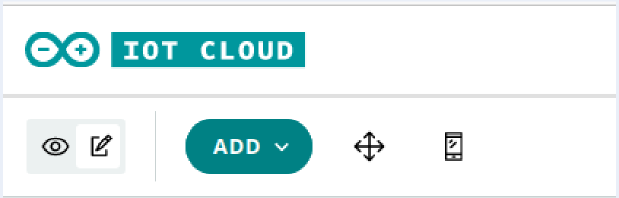


*Figure 5: Switch the dashboard to edit mode by clicking on the pencil icon top left.*

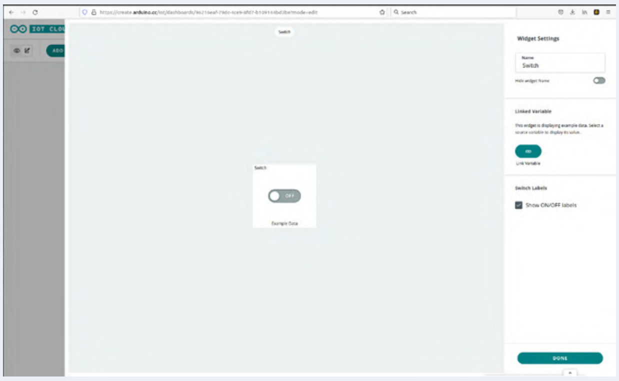
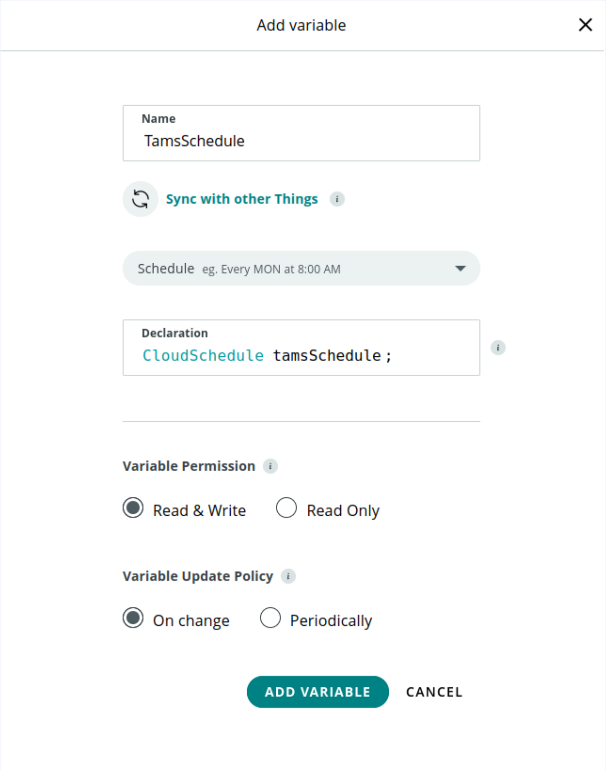


*Figure 6: The editing interface for controls is "modal".*



Add variable ✕

Name
TamsSchedule

Sync with other Things

Schedule eg. Every MON at 8:00 AM

Declaration
CloudSchedule tamsSchedule ;

Variable Permission

● Read & Write ○ Read Only

Variable Update Policy

● On change ○ Periodically

ADD VARIABLE CANCEL

*Figure 7: The Cloud Scheduler is a data type just like all the others.*

At this point, you can transfer the sketch to the circuit board again. The RGB LED uses common anode connections so individual colours are controlled by outputting a 0 to turn on the corresponding LED. The variables are initialised as shown in **Figure 4**, so that the blue diode of the RGB LED lights up after successful initialization.

## Modifing the Variable Contents

If we return to the main back end of the Arduino cloud, we can click on the *Dashboards* tab, which, if this is a new account, will prompt you to create a new dashboard. Now click the *Build Dashboard* button to launch the editor, which will take a moment or two even if you have a fast Internet connection.

To modify and add elements to the dashboard click on the edit (pencil) icon on the top left of the display (**Figure 5).** Once this mode is selected the blue *ADD* button appears which you can use to show the drop down list of available widgets. First, we select a *Switch* widget, which then appears in the editing interface shown in **Figure 6**.

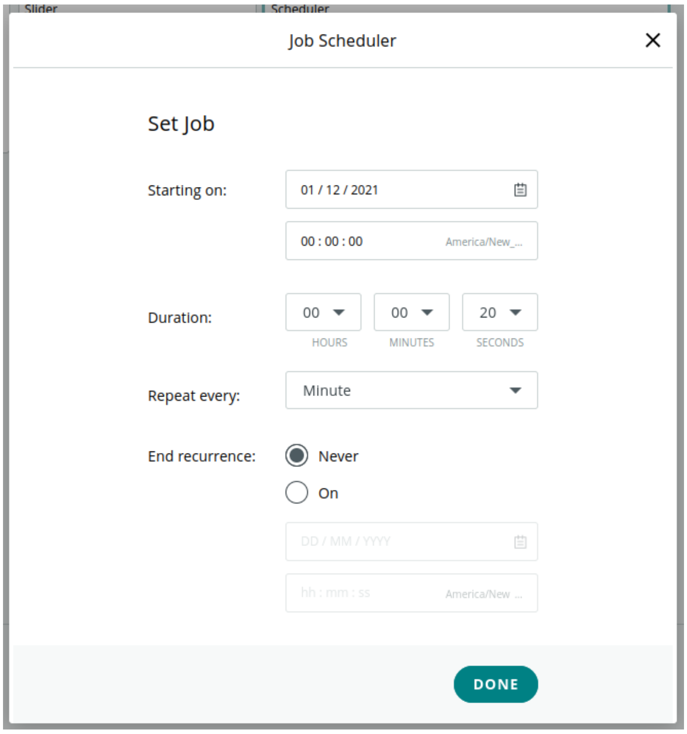
Now over to the right of the display is the *Linked Variable* field with a link button. Click on it to activate a list of all Things and Variables contained in the cloud account. Here we can choose the `ledIntenBool` variable and link it using the *Link Variable* button. The state of `ledIntenBool` will now be controlled by the state of the Switch. A click on the DONE button closes the editing interface and the switch is now incorporated into the dashboard. Now we can click on the eye icon to release the switch to activate. Toggling the switch on and off now controls the red LED next to the microUSB socket.

In order to be able to set the brightness of the blue LED, we have to switch the dashboard editor back to edit mode and add a new control again by using *Add* -> *Widgets*. This time I chose the *Slider* type. In its editing interface, we set its *Value Range* from 0 - 255. The link is made by the variable `letIntenInt`, which represents the RGB LED "brightness control". Last but not least, we switch to activation mode here and see that changes to the slider position now affect the brightness of the blue LED.

## Using the Scheduler

As I write this article, a function for setting up scheduled tasks or web cron-jobs is a new feature: it uses a variable type called *CloudSchedule* that you can define to be true or false at a specific time and for a specific time period. It's not necessary to invoke any timer function because this variable is set or reset automatically in the Arduino IoT Cloud, according to how you configure it. Tasks can then be triggered by checking the state of this variable. To demonstrate the possibilities, let's create a new variable of type *Schedule*. **Figure 7** shows the desired configuration. We can now see the new variable type in the code:

```
CloudSchedule tamsSchedule;
```


Slider Scheduler

Job Scheduler

Set Job

Starting on: 01 / 12 / 2021

00 : 00 : 00 America/New_...

Duration: 00 HOURS 00 MINUTES 20 SECONDS

Repeat every: Minute

End recurrence: ● Never ○ On

DD / MM / YYYY

hh : mm : ss America/New ...

*Figure 8: The Scheduler is configured using the Dashboard.*

The time parameters for this variable are configured via the dashboard. The settings page shown in **Figure 8** show the control elements provided for this purpose.

In the following step, we need to take care of the "local" processing of the values contained in `tamsSchedule`:

```
void setup() {
. . .

 pinMode(LED_BUILTIN, OUTPUT);
 pinMode(LEDB, OUTPUT);
 pinMode(LEDR, OUTPUT);
}

void loop() {
 ArduinoCloud.update();
 // Your code here
 if(tamsSchedule.isActive()){
  digitalWrite(LEDR, HIGH);
 }
 else{
  digitalWrite(LEDR, LOW);
 }
}
```

It is important here that the "writing out" of the information supplied in `tamsSchedule` is the exclusive task of the developer. The cloud is just limited to periodically updating the value contained in `tamsSchedule`. The continual polling procedure shown here in the `loop` structure may not be optimal from a resource point of view, but it works without problem. The program can now be sent to the Arduino where you can watch for the periodic red flashes coming from the RGB LED.

At this point, I could not resist another small system test to see what happens when the RF link goes down. The "lab Wi-Fi" was turned off and the Arduino began to act erratically by random switching the blue element of the RGB LED and also the LED on pin 13, eventually after a few seconds it performed a complete restart.

At the time this article went to press, it was not really clear to me how exactly the Arduino cloud recovers from the loss of the radio link between the end device and server.

## A Convenient Option

It's clear that the Arduino IoT Cloud is still a work in progress and is under constant development and improvement. It does however have great potential and offers the IoT application developer a convenient and low-threshold option for implementing a cloud back end without having the need to struggle with MQTT and Co. Despite the odd hiccup I can thoroughly recommend this product!

210550-01

Contributors
Idea, Illustrations and Text: Tam Hanna
Editor: Rolf Gerstendorf
Translation: Martin Cooke
Layout: Harmen Heida

## Questions or Comments?

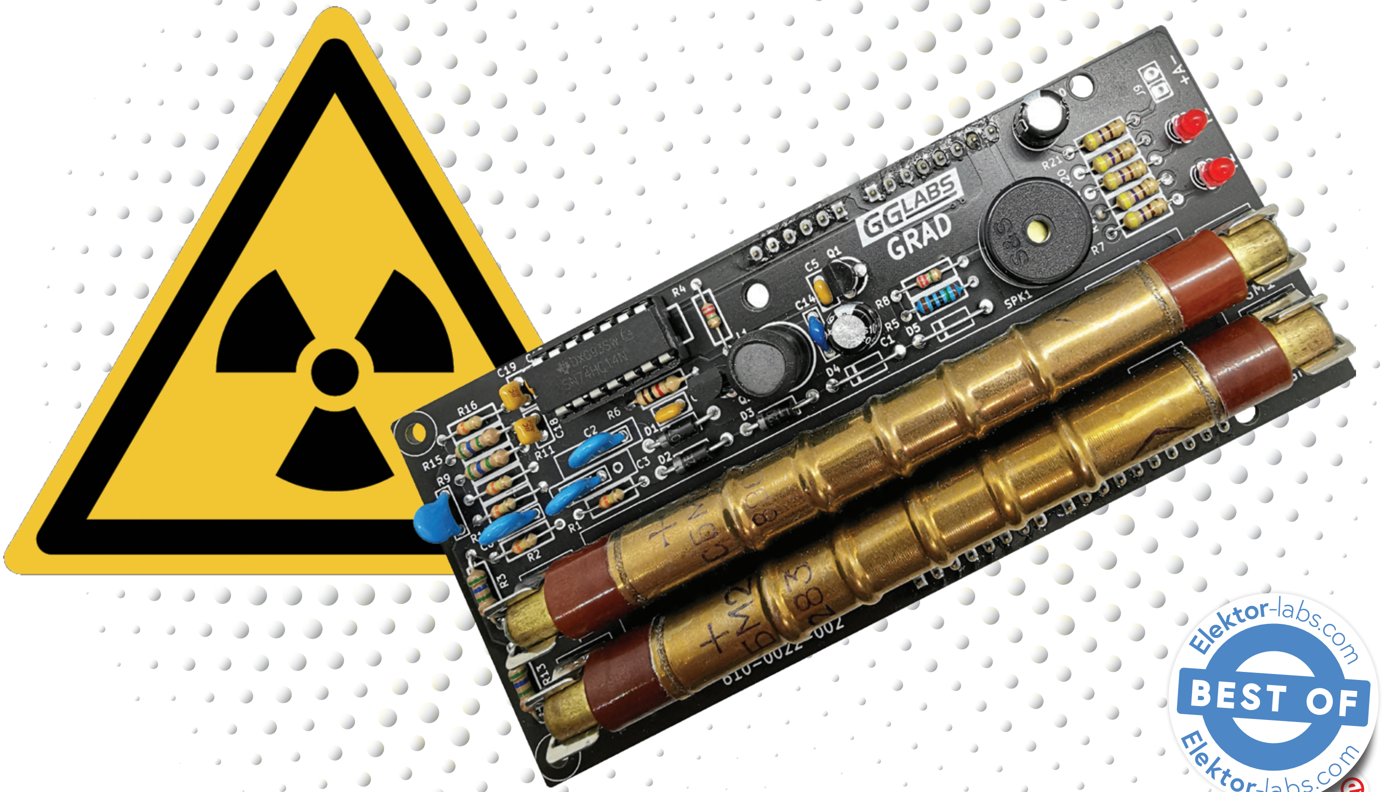
Do you have any technical questions or comments about this article? Contact the author at tamhan@tamoggemon.com or the Elektor team at editor@elektor.com.

## WEB LINKS

[1] Digital twin: https://en.wikipedia.org/wiki/Digital_twin
[2] Boards supported by the Arduino Cloud: https://bit.ly/3t8Vl3W
[3] Arduino Things: https://create.arduino.cc/iot/things

# Dual Geiger-Müller Tube Arduino Shield

## A High Sensitivity, Very Low-Power Radiation Sensor



By Gabriele Gorla (Italy/USA)

With two tubes for increased sensitivity, this Geiger-Müller tube shield can turn your Arduino Uno into an instrument for measuring and recording nuclear radiation. It can even be combined with a Dragino shield for LoRa-connectivity. Collect radiation data in the "field" — and get access to it from all over the world!

Like many other hobbyists, I have always been fascinated by radioactivity and the sensors to detect it. Geiger-Müller tubes [1] are a common and relatively inexpensive way to measure radiation. My "GRAD" project is a complete solution for radiation counting in an Arduino shield form factor. Its main features are dual tube support to increase sensitivity and very low power consumption.

### Introduction to Geiger-Müller Counters

Every Geiger-Müller counter requires four essential functional blocks.

- **Geiger-Müller Tube**. The tube has two terminals and is filled with a low-pressure gas mixture. When biased with the appropriate voltage the gas will ionize and briefly conduct electricity every time it is hit by radiation. Depending on the tube type it is possible to detect alpha, beta and gamma particles. The SBM-20 used in the GRAD is sensitive to gamma and high energy beta radiation.
- **High-Voltage Power Supply.** The tube must be operated in the so-called Geiger Plateau. This is a region where the pulse count is

almost independent of the bias voltage. For common tubes the bias voltage is between 400 V and 500 V. The SBM-20 optimal point is around 400 V.

- **Pulse Detector.** The pulses out of the tube are very short and of variable voltage. The pulse detector conditions the signal so it can be easily counted by the following stage.
- **Pulse counting.** Pulses are counted over a fixed time interval to calculate a CPS (counts per second) or CPM (counts per minute) value. This can be roughly converted to a dose rate by using the parameter on the tube datasheet. An Arduino board is used for pulse counting and its visualization and/or logging.

## Power Supply

There are many circuits on the web for Geiger-Müller tube power supplies. Many are boost converters built around the 555 timer, some open loop, some with feedback. The open-loop designs were not considered as they require tuning per board and do not provide a stable voltage at high pulse counts. Closed-loop designs are more suitable as they provide the required stability. However, to keep overall power consumption low, special attention should be paid to the feedback loop (12 µA of current @ 400 V is ~5 mW).

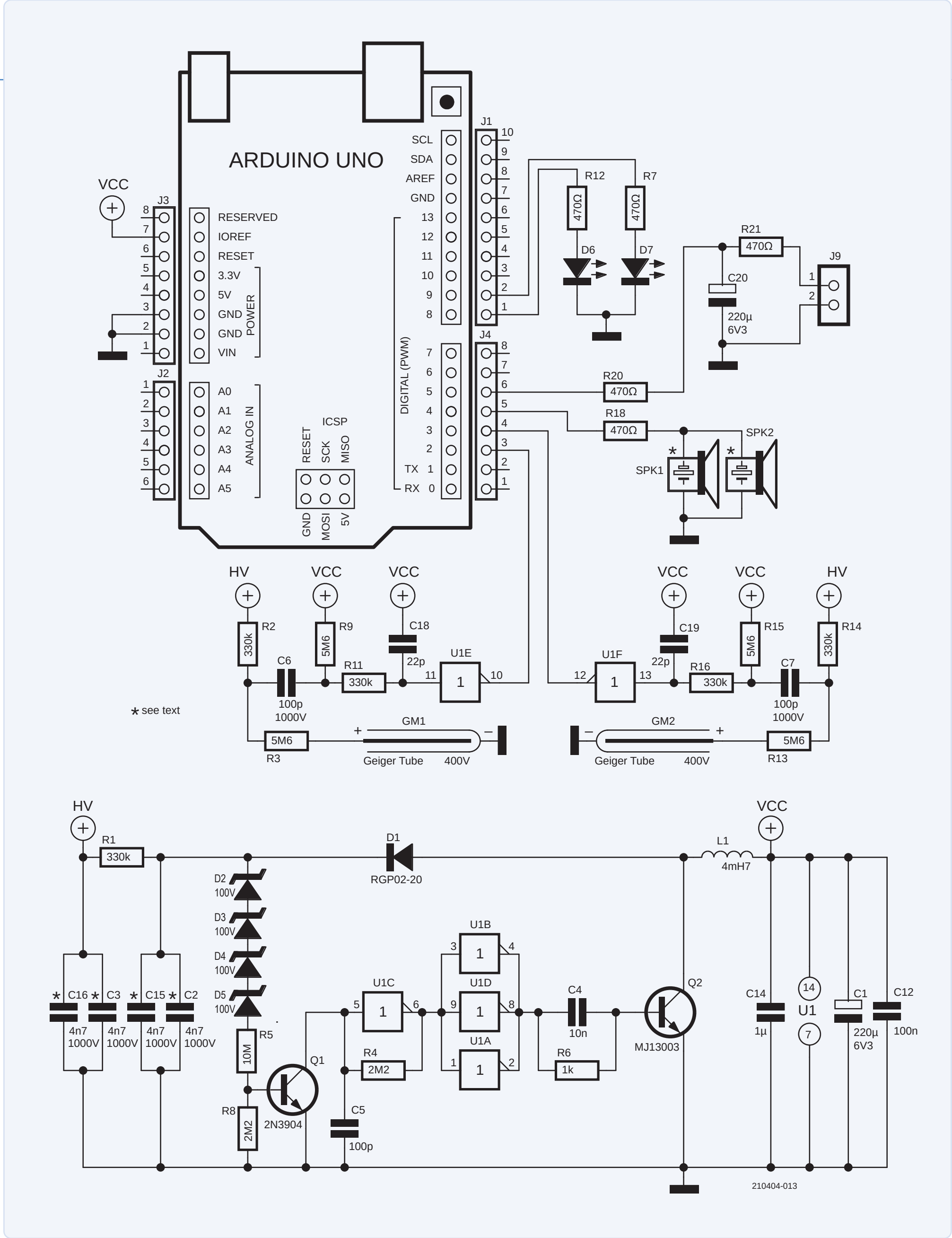


*Figure 1: Schematic diagram of the GRAD03 project [3].*

To avoid the power penalty of high-voltage feedback, the most elegant solutions use a non-isolated step-up transformer like the Analog Devices LT3420. It detects the voltage on the primary side.

We implemented a simple switch mode boost converter with a very low current feedback. Our design — shown in **Figure 1** — is heavily based on the Theremino Geiger adapters [2]. We mixed the SMD, DIY and "Flintstones" versions in a fully PTH design with Zener diode feedback.

The Schmitt trigger inverter U1C, R4 and C5 form an oscillator that generates the pulses to drive the main switch Q2. U1A, U1B and U1D are in parallel to increase Q2 base current.
The feedback is implemented with a series of very low leakage current Zener diodes (D2 to D5). When the output voltage exceeds the total Zener voltage, Q1 turns on and stops the oscillator. When the output voltage drops again Q1 will turn off releasing the oscillator. R1 and C3 further reduce the output ripple. Additional series resistors feed each tube separately (R2 and R3 for GM1, R13 and R14 for GM2).

## Counting

Pulses are picked up from the high voltage side of the tube using a DC blocking capacitor (C6 and C7). The pulse is shaped by the remaining Schmitt trigger inverters (see **Figure 2**).

J1 to J4 are the standard Arduino connectors, but GRAD only uses a few pins. Digital inputs D2 and D3 were chosen as they are interrupt pins on the Arduino Uno. This allow the Arduino to count pulses in the background while it is performing other actions.

Arduino digital outputs D8 and D9 are connected to a pair of LEDs

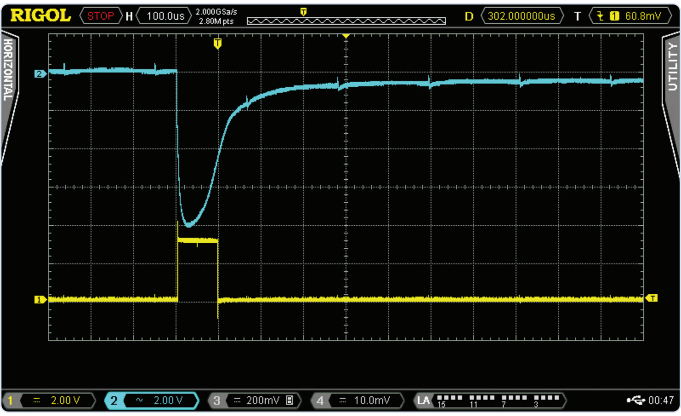


*Figure 2: Scope capture of ionizing radiation hitting tube and pulse shaping.*

to visually show when ionizing radiation hits the respective tube. Pin D4 is connected to piezo speaker SPK1 to provide audible feedback.

Finally, digital output D5 is connected to a low-pass filter formed by R20, R21 and C20 to connect a simple analog 10 mA panel meter (J9).

## PCB Design

The schematic and PCB (**Figure 3**) were designed in KiCad. The design files are available for download at this project's Elektor Labs page [3] in the Project's Elements section. The PCB's gerber and drill files can be found there too. You can use these to order a PCB from your preferred supplier. Last but not least, you'll find an Excel sheet there with a very detailed Bill of Materials, especially documenting the more critical (high-voltage!) components needed for this project, including data on manufacturers, type numbers and even order codes.

**Figure 4** shows the assembled PCB with SBM-20 tubes installed. Note that component pairs SPK1/SPK2, C3/C16, and C2/C15 respectively just provide alternative footprints on the PCB; so only SPK1 or SPK2 will be installed. The same goes for C3 or C16, and C2 or C15. J5 and J6 on the PCB denote two add extra footprints for connecting the negative terminals of shorter Geiger tubes.

## Tube Options

The board is designed to work with 105-mm Soviet SBM-20, STS-5 and the Chinese J305 or the 90-mm J305 and M4011. Any other 400-V tube will also work with some customized tube mounting. In case of custom mounting, to minimize parasitic capacitance, it is important to keep the positive wire as short as possible. For tubes requiring different voltages, the Zener diode(s) should be changed to obtain the required voltage. Any diode with 0.5 µA or less of leakage should work.

## Performance

The power supply consumes 325 µW (65 µA) at 5 V. In battery-operated devices the board can also be powered with 3.3 V. At this lower voltage, the power consumption drops to 150 µW (45 µA).

## Arduino Sample Code

A simple Arduino sketch to drive the counter is available in the Project

## COMPONENT LIST

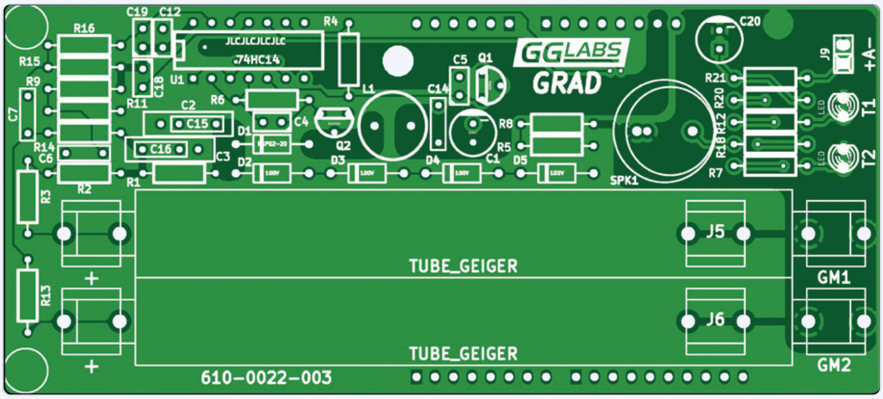


*Figure 3: The PCB layout.*

**Resistors**
R1,R2,R11,R14,R16 = 330 k
R3,R9,R13,R15 = 5.6 M
R4,R8 = 2.2 M
R5 = 10 M
R6 = 1 k
R7,R12,R18,R20,R21 = 470 Ω

**Capacitors**
C1,C20 = 220 µF, 6.3 V
C2,C3 = 4700 pF, 1000 V
C4 = 10 nF, 6.3 V
C5 = 100 pF, 50 V
C6,C7 = 100 pF, 1000 V
C12 = 100 nF, 50 V
C14 = 1 µF, 6.3 V
C15,C16 = 4700 pF, 1000 V
C18,C19 = 22 pF, 50 V

**Inductors**
L1 = 4.7 mH

**Semiconductors**
D1 = diode 1 A, 800 V
D2,D3,D4,D5 = Zener diode 100 V 1.5 W
D6,D7 = red LED 3 mm (Marked T1 and T2 on PCB)
Q1 = 2N3904
Q2 = MJ13003
U1 = 74HC14

**Miscellaneous**
GM1,GM2 = Geiger-Müller tube, e.g. SBM-20
J1,J2,J3,J4 = set of Arduino Uno shield connectors
SPK1 = AC piezo buzzer, e.g. AC-1205G-N1LF

Elements section of the Elektor Labs page. Every 60 seconds, the software will output via the serial port: a comma-separated values (CSV) row with a sequential number, raw count for each tube, a moving average of `cnt1 + cnt2` and a µSv/h dose rate based on the average count. The following output data fragment shows the start of a background radiation measurement made in Santa Clara, CA, using GRAD with two SBM-20 tubes.

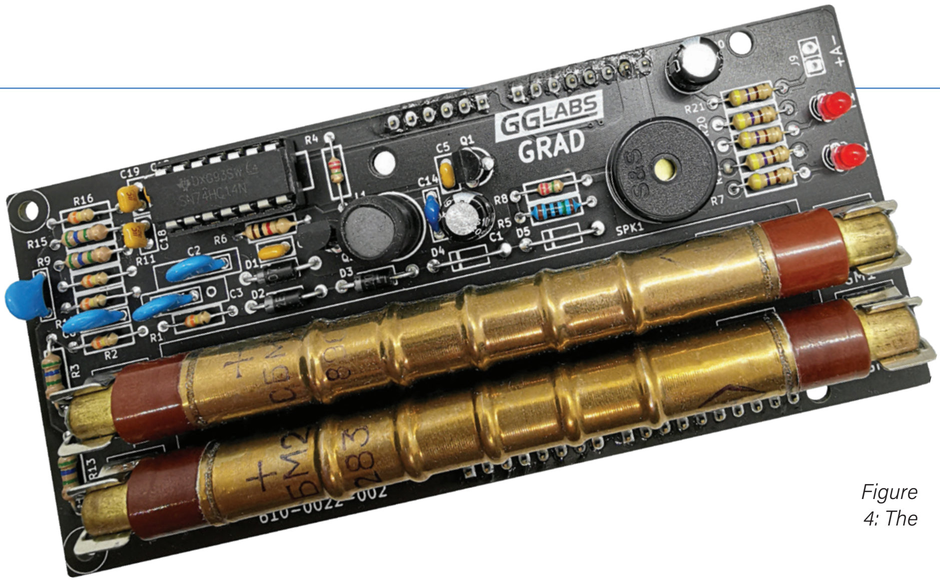


Figure 4: The

```
Seq , cnt1 , cnt2 , avg10 , µSv/h
1   , 14   , 19   , 33.0  , 0.075
2   , 10   , 17   , 32.3  , 0.073
3   , 16   , 12   , 31.8  , 0.072
4   , 16   , 24   , 32.5  , 0.074
5   , 13   , 11   , 31.6  , 0.072
```

The graph in **Figure 5** shows the complete measurement over a longer period of time.

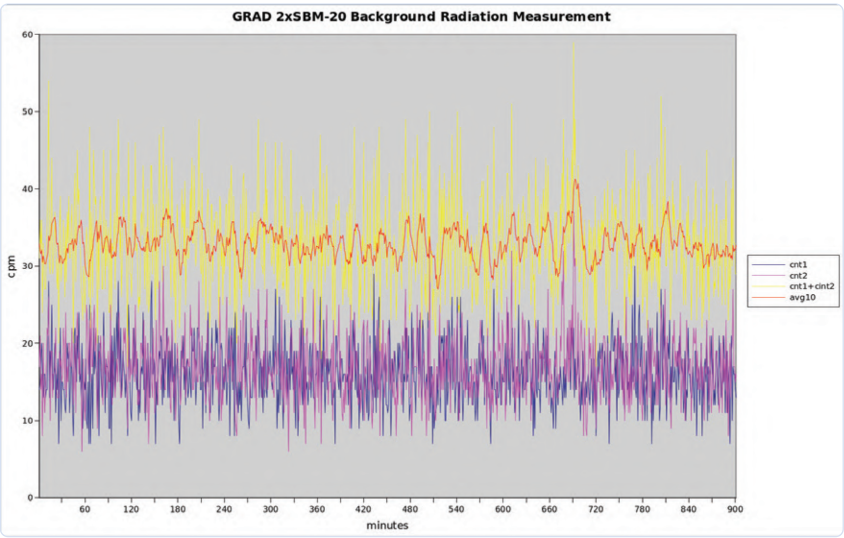


Figure 5: Background radiation in Santa Clara, CA.

There are a few defines at the beginning that are configurable to match the board configuration and/or user preferences. The first set configures the Geiger-Müller tube parameters and the moving window size for the reporting.

```
#define CPM2USV 220 // tube CPM to
                    // µSv/h conversion
                    // factor
#define TUBES 2     // number of tubes
                    // installed
#define WSIZE 10    // moving average
   window (in
                    // minutes)
```

The `CPM2USV` parameter is the count per minute for 1 µSv/h for your specific tube. Unfortunately, there is no published "correct" number for this parameter. For the SBM-20 radiation, enthusiasts on the web use values that range from 130 to 220 (conversion factor of 0.0075 to 0.0045).

The `TUBES` parameter, as the name states, defines the number of tubes installed. Valid values are 1 and 2. Finally, the `WSIZE` define the moving average window size for the counts. The default value of 10 defines a 10-minute window.

The second set of parameters defines how the optional hardware functions are connected:

```
#define LED1_PIN     8  // pin for TUBE1 LED
#define LED2_PIN     9  // pin for TUBE2 LED
#define SPKR_PIN     4  // pin for speaker connection
#define LED_BLINK_MS 20 // duration of LED blink for
                        // each count
```

All three `...PIN` defines select the Arduino pin where the LED or the speaker are connected. The `LED_BLINK_MS` defines the time that the LED will stay on for each count that the tube receives.

## LoRaWAN Networking

The basic sketch requires the unit to be tethered to a computer through USB. Often it is desirable to place the sensor far from the computer, where a USB connection is not available.

A Dragino LoRa shield can be stacked to enable LoRa connectivity. Unfortunately, the Dragino shield also uses Arduino pin D2, so there are two options — a single-tube option and a two-tube option.

Single-tube option: In this case, only tube GM2 can be used and pin 10 of Schmitt trigger U1 must be disconnected to prevent interference with the LoRa communication. The pictures in **Figure 6** show the full stack with a single soviet SBM-19.

Two-tube option: In this case, both the Dragino shield and the GRAD

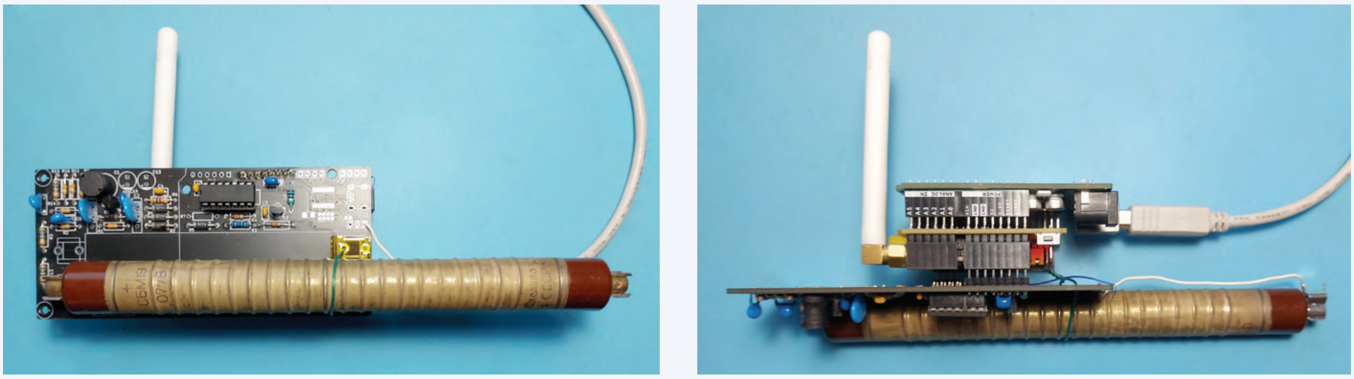


Figure 6: GRAD03 board, Arduino Uno and the Dragino LoRa shield combined.

board must be modified. For the Dragino shield, R5 and J_DIO0 must be removed. A wire should be used to connect the radio DIO0 to Arduino D7.

For the GRAD board R7, R12, C20 and the two LEDs should be removed. R21 should be replaced with a short, and a single LED should be placed at the J9 site (replace the analog meter output).

A sketch that communicates to the The Things Network (TTN) and posts tube counts every 60 seconds is available for download. It uses the LMIC Arduino library to drive a Dragino LoRa shield and it will connect to The Things Network using Over The Air Activation (OTAA). A simple Node-RED flow is used to display the data (see **Figure 7**).

## Measuring Radiation

To verify the correct function of the Geiger counter a radiation source is required. These can be purchased online from special suppliers. Alternatively, specific vintage items that were built using small quantities of radioactive materials can be purchased at a thrift store or Ebay. Common radioactive items include: uranium glass items, thorium lantern mantles and selected colors of vintage fiestaware items.

Lacking any of these, another simple way is to check the decay product of Radon captured by the air filter from an air conditioner or an air purifier (**Figure 8**). These have a relatively short half life (in the order of tens of minutes) so make sure you run the AC for a couple of hours and then measure the filter immediately. If the filter is fine enough to it will emit radiations at a rate many times higher than background in your area.

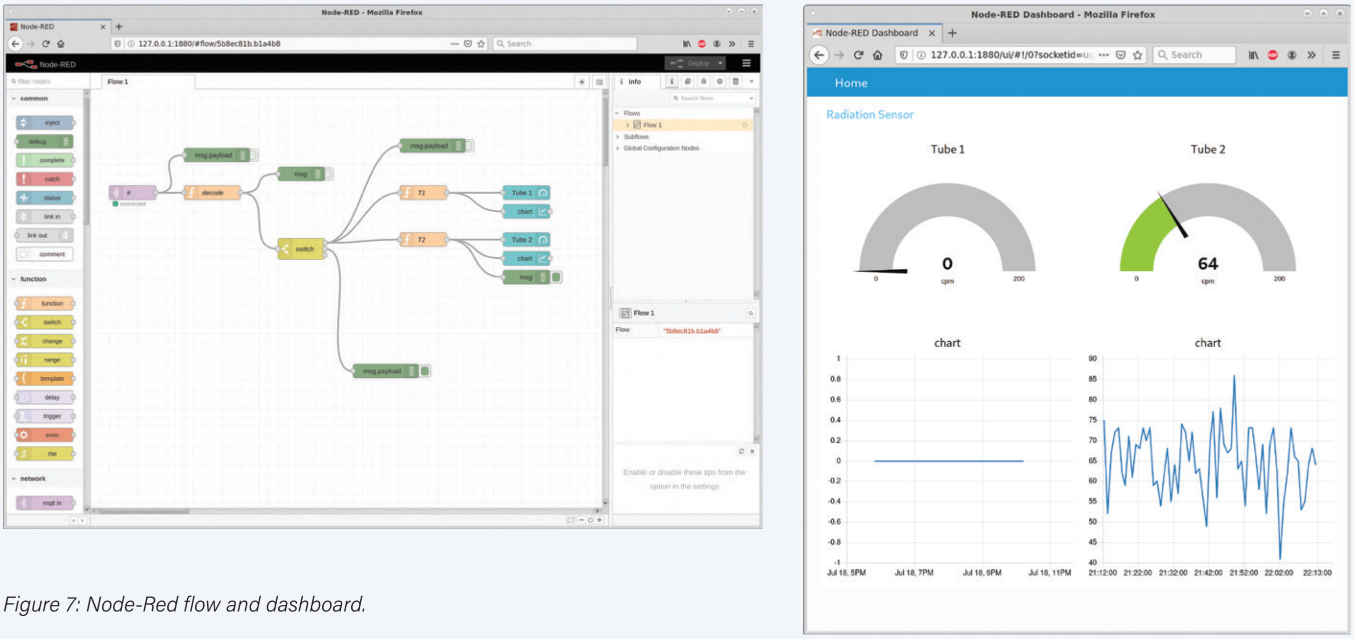


Figure 7: Node-Red flow and dashboard.

The chart below shows the count reaching almost ten times the background and then following the exponential curve typical of radioactive decay. ⏮

210404-01

*This article is based on the material presented on the Elektor Labs page of this project [3]. There you'll find all the downloads for GRAD03, plus discussions and remarks on this subject.*

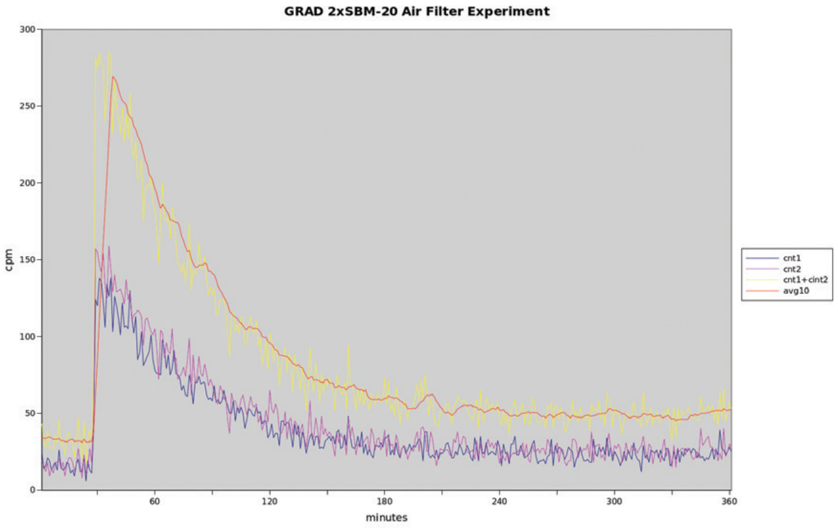


*Figure 8: Radiation from an air conditioner filter due to radon decay.*

## Contributors

Design: **Gabriele Gorla**
Text: **Gabriele Gorla, Luc Lemmens**
Illustrations: **Gabriele Gorla, Patrick Wielders**
Editors: **Jens Nickel, C. J. Abate**
Layout: **Harmen Heida**

## Questions or Comments?

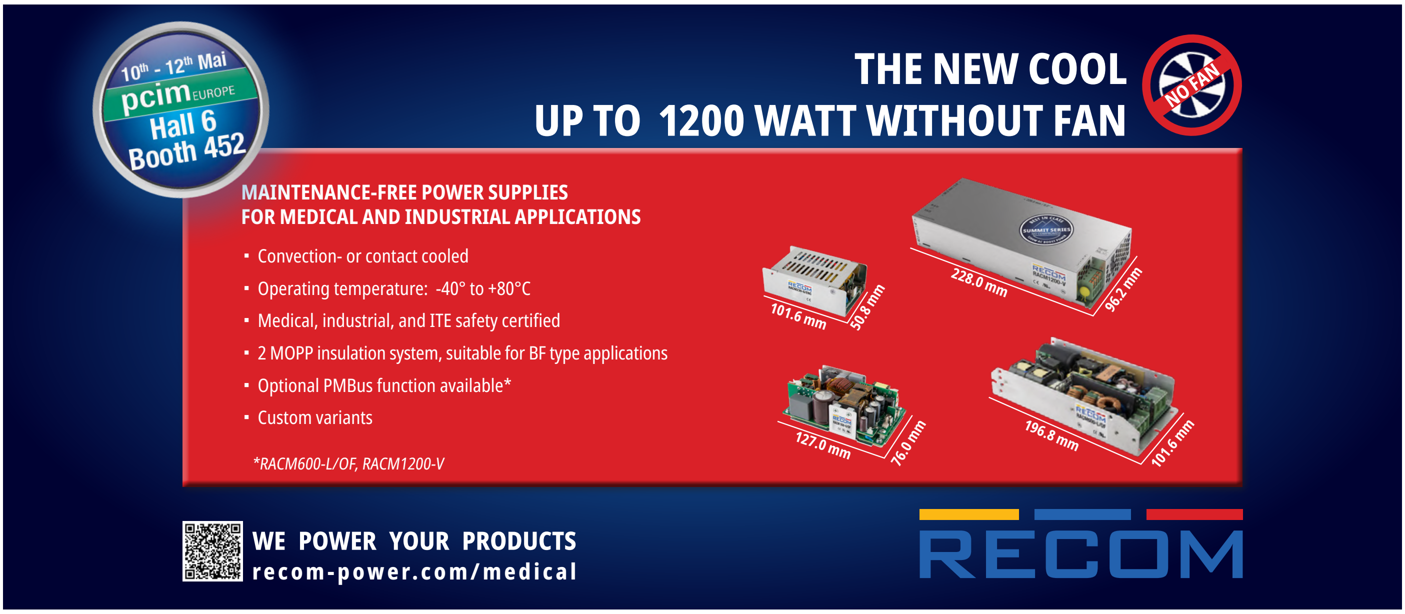
Do you have technical questions or comments about this article? Email the author at gorlik@yahoo.com or Elektor at editor@elektor.com.

## RELATED PRODUCTS

- **Arduino Uno SMD Rev3 (SKU 19938)**
  www.elektor.com/19938
- **MightyOhm Geiger Counter Kit (incl. Case) (SKU 18509)**
  www.elektor.com/18509

## WEB LINKS

[1] Geiger-Müller tubes: https://en.wikipedia.org/wiki/Geiger%E2%80%93M%C3%BCller_tube
[2] Theremino Geiger adapter (pls see three last designs on this page): www.theremino.com/en/technical/schematics
[3] This project on Elektor Labs: https://bit.ly/3giMntz

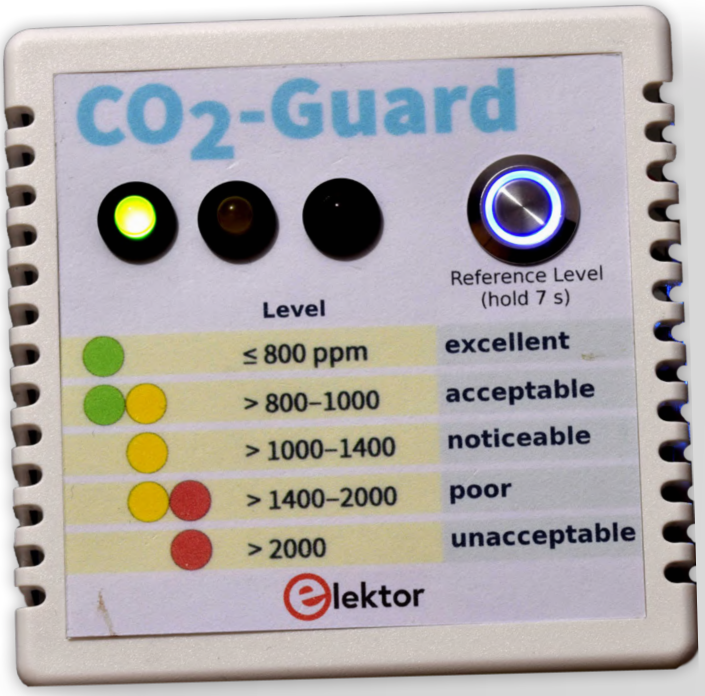


# $CO_2$ Guard

## A DIY Approach to Monitoring Air Quality

By Mathias Claußen, Ton Giesberts, and
Luc Lemmens (Elektor)
Design by Florian Schäffer (Germany)

High $CO_2$ concentrations can cause problems in areas of minimal ventilation. This $CO_2$ guard offers a solution. The design, constructed with through-hole components, warns users with three LEDs and an acoustic alarm when it is time for fresh air. PCB Gerber and enclosure front panel design files are free to download.



This $CO_2$ Guard is a DIY project for indoor air monitoring and the assessment of air quality based on the carbon dioxide concentration (Indoor Air Quality, IAQ). The device uses a non-dispersive infrared (NDIR) sensor and displays the air quality in five ranges on a triple LED scale. In addition, an alarm sounds if the $CO_2$ concentration is unacceptably high. It is possible to transfer the measured values via Wi-Fi ($CO_2$ value and temperature) to the ThingSpeak IoT analytics platform and evaluate the data graphically.

There are many factors that influence air quality, but the $CO_2$ concentration in a room is one that is felt almost immediately. Too high concentration of this gas causes fatigue, loss of concentration, or even nausea, not to mention the lasting negative effects if there is far too much $CO_2$ in your environment. The latter is (hopefully!) not likely to happen in a home or office, but this $CO_2$ Guard can give you a timely indication to ventilate a room.

### The Circuit

The heart of the schematic shown in **Figure 1** is MOD2, a Wemos D1 Mini, which is an ESP8266EX-based Wi-Fi module with 4-MB flash program memory. It collects data from a MH-Z19C $CO_2$ sensor (MOD1) via a software UART, drives three LEDs to indicate $CO_2$ levels (LED1...3), and controls buzzer BZ1, which serves as acoustic alarm when the $CO_2$ level is unacceptably high. The $CO_2$ Guard is powered via the micro-USB connector of the Wemos module, using a standard 5 VDC mains adapter. This supply voltage is routed to pin 9 of MOD2 via an internal diode and fuse, and to an on-board 3.3 V voltage regulator. The output of this LDO is used for the internal power supply of the Wemos module itself and is also connected to pin 8 of the module. This 3.3 V could be used for powering a miniature fan connected to K1, as we will discuss later. The 5 V output of the MOD2 is used for powering the $CO_2$ sensor and for an additional LED inside switch S1 to indicate that the Wi-Fi connection is active. This supply can also be connected to K1 (via R6) as an alternative power source for the fan. Push button S1 can trigger the $CO_2$ sensor to start automatic calibration.

### Powering the Fan

In this project, the fan is not used for cooling, but for airflow past the sensor. Although it seems obvious to power a 5 V fan at its nominal voltage, if maximum airflow is not the most important design consideration — as in this project — it may be a good idea to minimize the noise of this device by reducing the airflow via the supply voltage. With our $CO_2$ Guard, there are some options in this respect.

The 3.3 V supply rail of the Wemos module is regulated by an low drop out regulator (LDO) in a SOT23-5 package. The maximum output current of this LDO is limited to 150 mA due to its package. A significant portion of the output current is used for the ESP8266 of the Wemos D1 Mini module itself. Overloading the LDO can result in instablility of the ESP8266 and therefore random crashes. To avoid overloading the 3.3 V regulator, there's an option on the PCB of the $CO_2$ Guard to connect the fan to 5 V through a resistor (R6, the footprint on the PCB is for a 1 W type) to the 5 V. This makes it possible to reduce the airflow and, more importantly, the noise even further. Be aware that fans runnung at a lower voltage as they are designed for tend to have problems during startup or not starting at all. If the fan should be directly connected to 5 V, a jumper wire can be used for R6 instead.

## Other Hardware

The buzzer is also powered by the 5 V supply. The high output level of the I/Os of the D1 Mini is 3.3 V, so to turn the buzzer off, the voltage divider R4/R7 is connected to the base of PNP transistor T1. Some buzzers are inductive and likely to produce spikes, especially when switching off. Diode D1 prevents these spikes from harming the transistor. The PCB has footprints for both 6.5 mm and 7.62 mm pitch buzzers.

The switch we used for S1 is one with an optical indicator in the form of a blue ring. No external resistor is needed for the LED inside the switch. The pins with + and – markings on the PCB, next to two pins for the switch, are the two connections for the LED. Four wires are connected to four-way pin header S1 (or soldered directly to the PCB). Of course, you can also use a separate switch and LED, R5 then sets the current for the LED and the front panel design needs to be modified to accommodate this optical indicator. With the switch prescribed in the component list, R5 is not needed and must be replaced with a jumper wire.

## PCB

The design and Gerber files for the printed circuit board shown in **Figure 2** are available for download from the GitHub repository [1] of this project. With the Gerber files, you can order the board from your preferred PCB manufacturer. All components are through-hole and the PCB could be single sided; the top layer of the PCB is only filled with copper to avoid the additional costs that single-layer PCBs will have nowadays. So, even for inexperienced makers, soldering will be not too difficult if the guidelines given here are followed. And with all traces on the bottom layer of this PCB, it is relatively easy to desolder components if a mistake is made. Start with mounting the lowest components first, resistors and diode, then the capacitors, transistor, buzzer and headers (sockets for the $CO_2$ sensor MOD1,

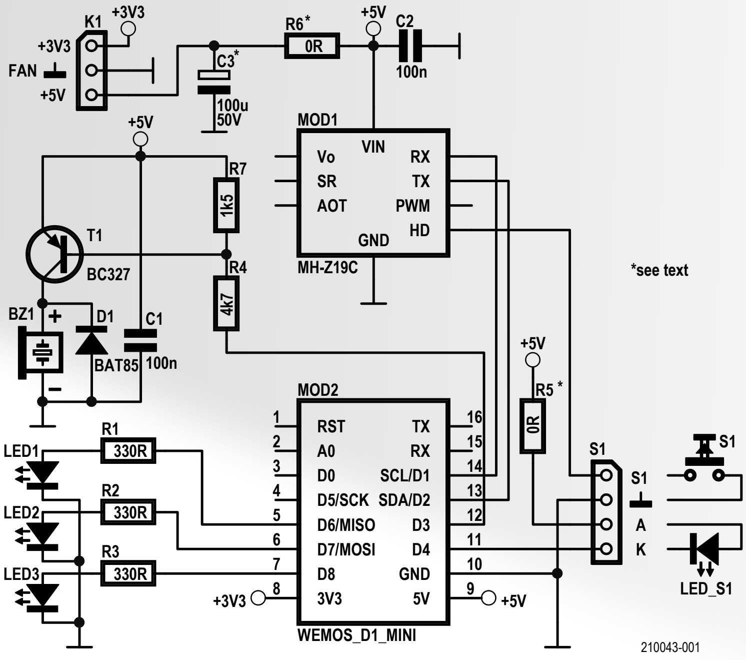


Figure 1: Schematic of the $CO_2$ Guard.

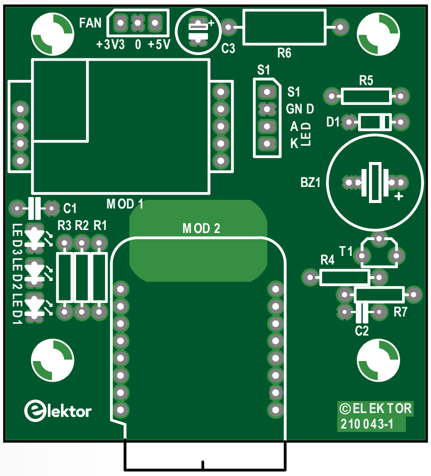


Figure 2: PCB of the $CO_2$ Guard.

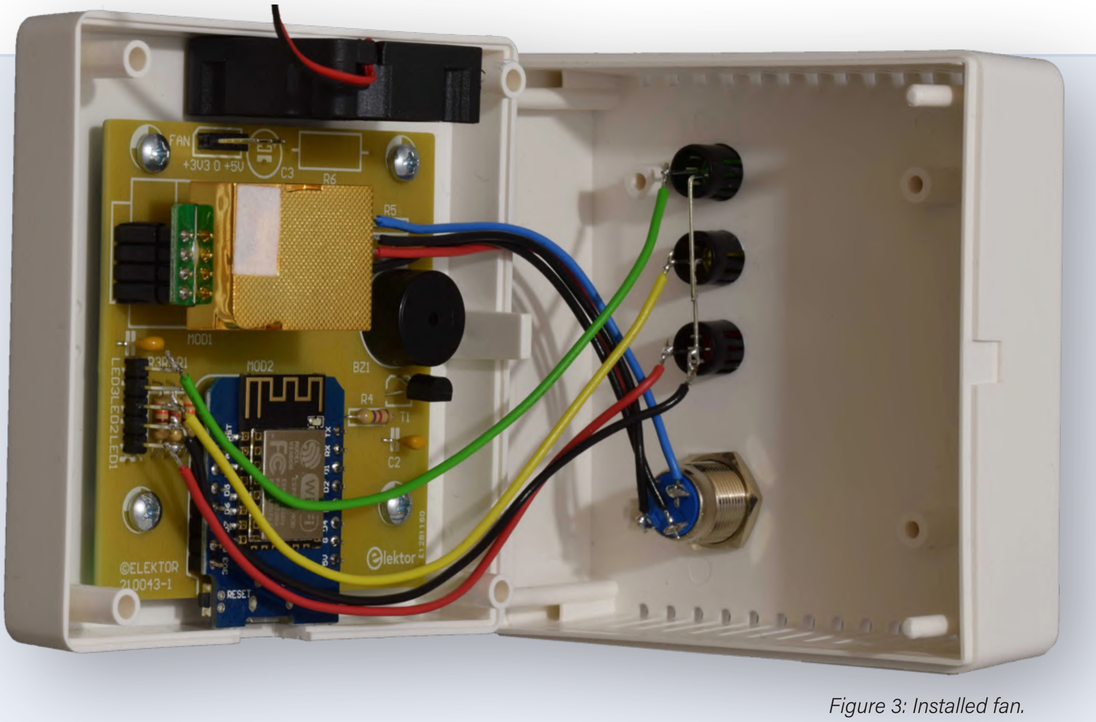


*Figure 3: Installed fan.*

optional pin headers for the switch and LEDs). To mount the D1 Mini module (MOD2), first solder the two eight-pin male headers to the side of the module where the USB connector and most of the other components are located (see **Figure 3**). We advise not using sockets on the PCB for MOD2, since there's some force on the module when plugging and unplugging the USB connector. It's best to solder the D1 Mini onto the PCB. Also, using sockets could limit airflow through the ventilation holes of the enclosure.

When the PCB is fixed with four screws to the bottom half of the enclosure and the fan glued (see below), the sensor module can be plugged into the corresponding sockets. **Never touch the top of the sensor**, as stated in the datasheet [2]: "*Please avoid the pressure of its gilded plastic chamber from any direction, during welding, installation, and use. Especially never touch the air intake (white dust filter).*"

The four and five pins sockets for the sensor will probably be supplied as one 10-pin SIL socket. Cut this socket strip at the location of the fifth pin to get a separate four- and five-pin socket.

## Preparing the Enclosure

Use a copy of the front panel design as a template for the correct positions for the four holes (three LEDs and the switch), and use a punch tool to mark the four centers. On this project's page on Elektor Labs [3], there are some tips and tricks to prevent damage to the case when drilling larger holes in the plastic front panel. With this type of enclosure, pay special attention to the long pins that are used to close it; they can break if not handled properly!

A notch must be made in one side of the bottom half of the enclosure to access the micro-USB connector of the D1 Mini module, which serves as the power supply for the $CO_2$ Guard. Make sure the notch is big enough for the USB-plug of your power supply. Place the D1 mini module on the PCB to mark the correct place for the notch. No need to solder the module with the accompanying male headers to the PCB yet. With the module placed on the PCB and the PCB placed correctly on the screw fixings, maybe use two screws temporarily, the position of the notch can be marked. It's slightly off center.

No holes need to be made for the fan's airflow; it is glued to the bottom side of the enclosure with super glue or epoxy adhesive (**Figure 3**). To avoid resonance the fan should not touch the lid of the enclosure. The fan should blow air out of the enclosure. Before mounting it, check the direction of the airflow by connecting it to a 3.3…5 V power supply or check if there is an arrow on the side of the fan's frame depicting the flow direction. Pay close attention to the polarity of the wires. The wrong polarity will damage the fan. (Modern ones often have some circuitry inside.)

## Connecting the Front Panel Parts

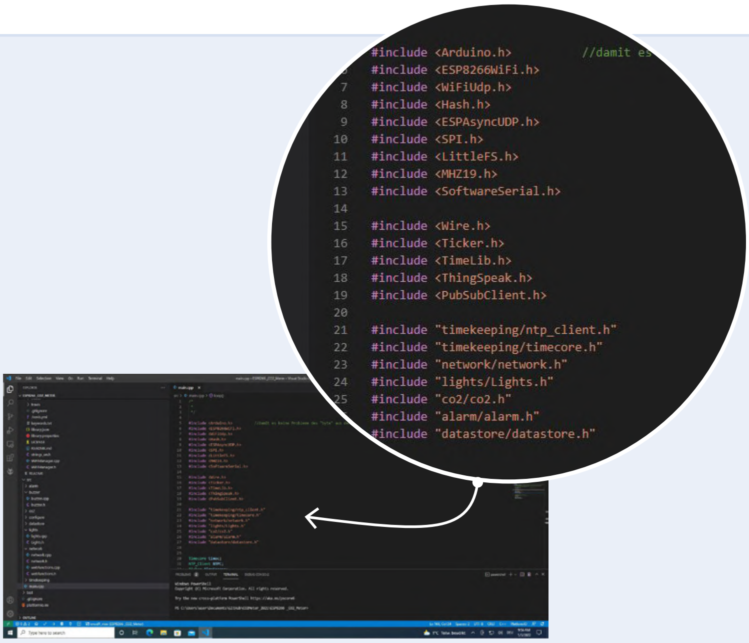
To connect the LEDs, only four wires are needed. The ground connection from the PCB to three LEDs on the front can be one common connection, simply connect three cathodes of the LEDs and use one single wire.

Make sure no wires lie on the antenna of MOD2 when the lid of the enclosure is finally placed on the bottom part. Keep them as far away as possible from the Wemos module, but don't make them longer than necessary. Wires to the switch can be as short as 8.5 cm and the wires to the LEDs about 12 cm. With this length of the wires, you can remove the top of the enclosure and place it on its side next to the bottom with all wires still attached. No screws are needed to close the enclosure; four long plastic pins of the lid slide into four long mounting bosses.

Of course, feel free to create your own front panel design. You can use a separate blue LED and switch for instance. But be aware of the placement of the LEDs and switch; they should not be located too close to the modules.

## Getting Firmware to the Wemos

The firmware for the ESP8266 on the Wemos module is written using



```
#include <Arduino.h>        //damit es
#include <ESP8266WiFi.h>
#include <WiFiUdp.h>
#include <Hash.h>
#include <ESPAsyncUDP.h>
#include <SPI.h>
#include <LittleFS.h>
#include <MHZ19.h>
#include <SoftwareSerial.h>

#include <Wire.h>
#include <Ticker.h>
#include <TimeLib.h>
#include <ThingSpeak.h>
#include <PubSubClient.h>

#include "timekeeping/ntp_client.h"
#include "timekeeping/timecore.h"
#include "network/network.h"
#include "lights/Lights.h"
#include "co2/co2.h"
#include "alarm/alarm.h"
#include "datastore/datastore.h"
```

Figure 4: Code loaded into PlatformIO IDE.

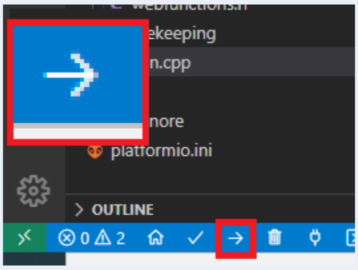


Figure 5: Button for the firmware upload.

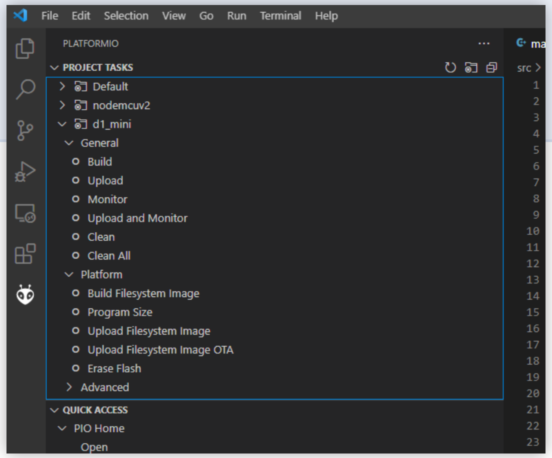


Figure 6: Small ant head shows more options.

the Arduino Framework, so you need the sketch and a set of libraries. For the development of the code of this $CO_2$ Guard, PlatformIO and Visual Studio Code are used. Libraries, project, and upload settings are combined so that compiling the code should not be that much of a problem.

To install Visual Studio Code and PlatformIO, you can follow the guide [4] PlatformIO provides. After the installation, you need to grab the source code from the Elektor GitHub Repository. If you are in doubt how to get the code, have a look at the video that our colleague Clemens Valens [5] created. Also, if you would like to know more about GitHub and the usage of Repositories, take a look at the Elektor Webinar about GitHub [6]. After you have the code and open it with the PlatformIO IDE, it should look like what you see in **Figure 4.** Connect the Wemos to your computer and hit the upload button (**Figure 5**). PlatformIO will download all the required libraries, tools, board support packages and start compiling and afterwards uploading.

PlatformIO now has built and flashed the firmware. As with most ESP8266 projects, there is a second part to it. As the firmware offers configuration through a webserver, we need also to write the webpages to the ESP8266 in a second step. Click in Visual Studio Code on the small ant head and a list of tasks should appear (**Figure 6**). First click on *Build Filesystem Image* and afterwards on *Upload Filesystem Image*. This will write the content for the Webserver to the ESP8266. If you get errors like "can't access COM port," make sure you don't have a serial terminal somewhere open that will block the port. Afterwards, firmware is ready to be configured and used.

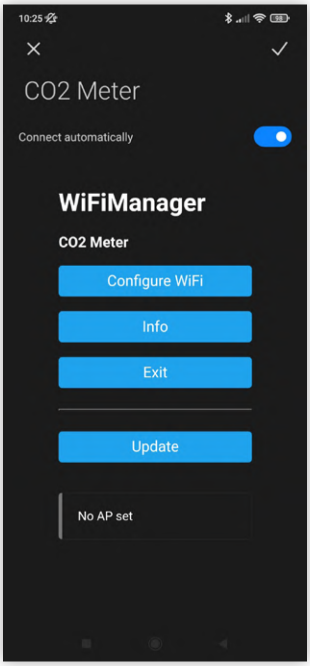
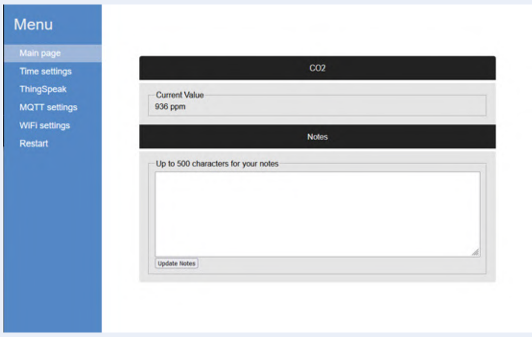


Figure 7: Wi-Fi settings.

## Wi-Fi and Cloud Connectivity

If your ESP8266 on the Wemos module can't connect to a Wi-Fi network, it will start an access point called "CO2 Meter". Connect to it and open a browser pointing to *http://192.168.4.1*. You should see the WiFi Manager configuration interface (**Figure 7**). Hit *Configure WiFi* and select your network and apply the required settings before



CO2

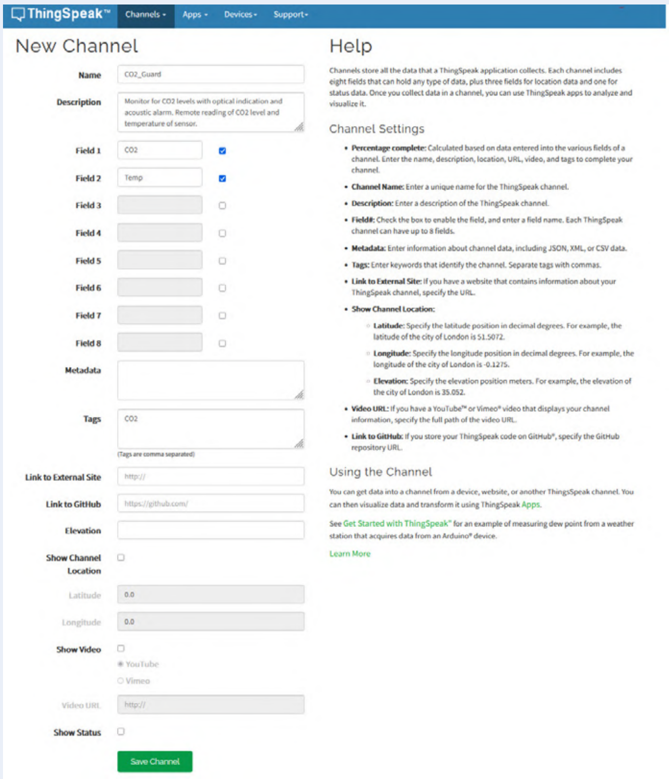
Current Value
936 ppm

Notes

Up to 500 characters for your notes

Update Notes

Figure 8: $CO_2$ Guard web interface.



## New Channel

Name: CO2_Guard

Description: Monitor for CO2 levels with optical indication and acoustic alarm. Remote reading of CO2 level and temperature of sensor.

Field 1: CO2 ☑

Field 2: Temp ☑

Field 3: □

Field 4: □

Field 5: □

Field 6: □

Field 7: □

Field 8: □

Metadata:

Tags: CO2
(Tags are comma separated)

Link to External Site: http://

Link to GitHub: https://github.com/

Elevation:

Show Channel Location: □

Latitude: 0.0

Longitude: 0.0

Show Video: □
● YouTube
○ Vimeo

Video URL: http://

Show Status: □

Save Channel

## Help

Channels store all the data that a ThingSpeak application collects. Each channel includes eight fields that can hold any type of data, plus three fields for location data and one for status data. Once you collect data in a channel, you can use ThingSpeak apps to analyze and visualize it.

### Channel Settings

- **Percentage complete:** Calculated based on data entered into the various fields of a channel. Enter the name, description, location, URL, video, and tags to complete your channel.
- **Channel Name:** Enter a unique name for the ThingSpeak channel.
- **Description:** Enter a description of the ThingSpeak channel.
- **Field#:** Check the box to enable the field, and enter a field name. Each ThingSpeak channel can have up to 8 fields.
- **Metadata:** Enter information about channel data, including JSON, XML, or CSV data.
- **Tags:** Enter keywords that identify the channel. Separate tags with commas.
- **Link to External Site:** If you have a website that contains information about your ThingSpeak channel, specify the URL.
- **Show Channel Location:**
  - **Latitude:** Specify the latitude position in decimal degrees. For example, the latitude of the city of London is 51.5072.
  - **Longitude:** Specify the longitude position in decimal degrees. For example, the longitude of the city of London is -0.1275.
  - **Elevation:** Specify the elevation position meters. For example, the elevation of the city of London is 35.052.
- **Video URL:** If you have a YouTube™ or Vimeo® video that displays your channel information, specify the full path of the video URL.
- **Link to GitHub:** If you store your ThingSpeak code on GitHub®, specify the GitHub repository URL.

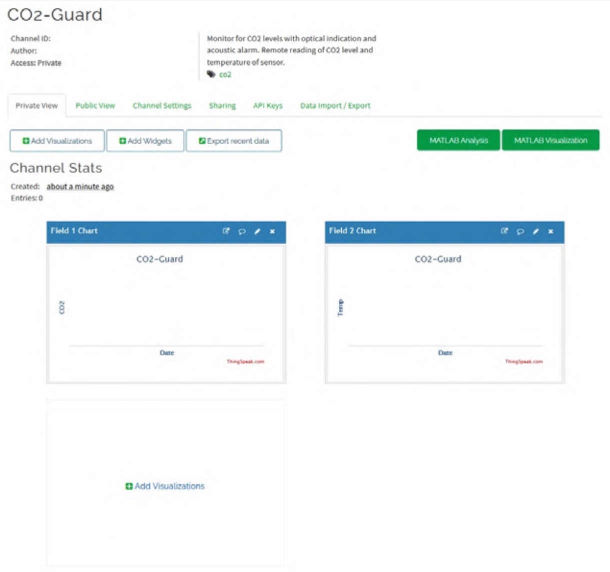
### Using the Channel

You can get data into a channel from a device, website, or another ThingsSpeak channel. You can then visualize data and transform it using ThingSpeak Apps.

See Get Started with ThingSpeak™ for an example of measuring dew point from a weather station that acquires data from an Arduino® device.

Learn More

Figure 9: ThingSpeak settings.



## CO2-Guard

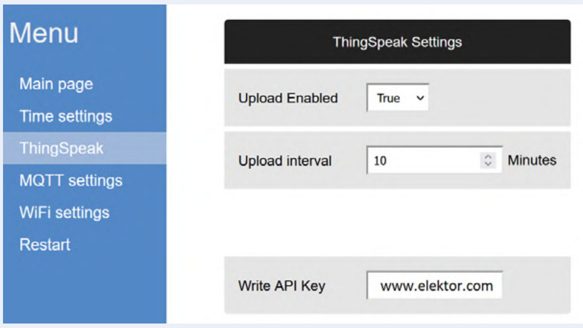
Channel ID:
Author:
Access: Private

Monitor for CO2 levels with optical indication and acoustic alarm. Remote reading of CO2 level and temperature of sensor.
co2

### Channel Stats

Created: about a minute ago
Entries: 0

Figure 10: New generated ThingSpeak channel.



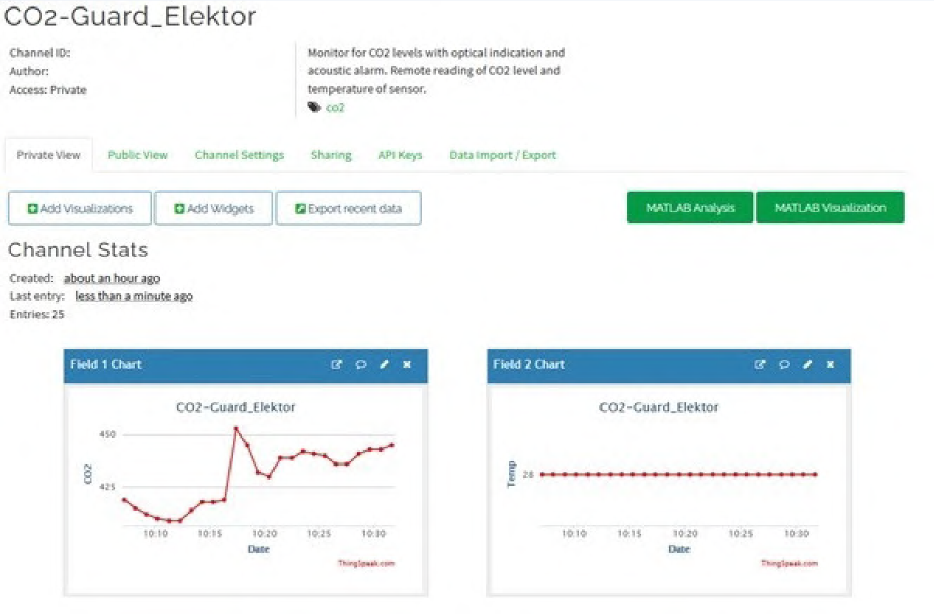
ThingSpeak Settings

Upload Enabled: True

Upload interval: 10 Minutes

Write API Key: www.elektor.com

Figure 11: ThingSpeak settings.



## CO2-Guard_Elektor

Channel ID:
Author:
Access: Private

Monitor for CO2 levels with optical indication and acoustic alarm. Remote reading of CO2 level and temperature of sensor.
co2

### Channel Stats

Created: about an hour ago
Last entry: less than a minute ago
Entries: 25

Figure 12: Arriving data.

hitting save. The ESP8266 will now try to connect to your network. If it can establish a connection, you can use within your network *http://co2guard.local* to access the web interface (**Figure 8**). From here you can start configuring the $CO_2$ guard.

If you like, you can connect the $CO_2$ Guard to a web-based data storage solution like ThingSpeak or an MQTT broker, like the one used in Node-RED. With the latter, you can add even more automation to the sensor — for instance, sending a message to your mobile phone if the $CO_2$ exceeds a certain level. To get started with ThingSpeak, a short tour through the account setup is provided.

## ThingSpeak: Create a MathWorks Account

If you only want to monitor the current concentration of $CO_2$ in the room you are in, you do not need a Wi-Fi connection and certainly not access to a ThingSpeak account, the LEDs (and in extreme cases: the buzzer) then give a clear indication of the air quality, at least: regarding $CO_2$. But if you want, for example, to log the course of these measurements and analyse the data over time, or be able to see from a greater distance what the current concentration is, then connecting to ThingSpeak via Wi-Fi is an obvious choice. You can then also follow the temperature, which is also measured by the sensor module, but not displayed in any way on the $CO_2$ Guard itself.

To create an online channel for your measurements, go to the ThingSpeak website [7] and choose *Get Started For Free*. On the next page you can create a MathWorks account (if you don't already have one). After this, create a new channel and enter the required data as suggested in **Figure 9**. Click on *Save Channel*. An overview of the channel just created is shown as in **Figure 10**.

At the tab *API Keys*, we need to copy the *Write API Key* for later use. Keep it private and don't show it to anyone else. Go to the $CO_2$ Guard's web interface and select the ThingSpeak option on the side menu. A page with the settings for ThingSpeak will show up. Enter the API Key, select an upload interval and enable ThingSpeak (**Figure 11**). You now switch back to the MathWorks ThingSpeak site. Under menu *Channels* choose *My Channels* and then the tab *Private View*. It should show after a short time the first measurements like in **Figure 12**.

To share your data with others, you are able to add individual users with viewing permission, or you can make the channel public so anyone can view your data. Individual users will get an email to the email address you can enter in the tab *Sharing*. If you are using a free (non-paid) plan for ThingSpeak, the sharing of data is limited to only three other users. For more details about sharing data you can consult the MathWorks help center [8].

## MQTT Connection

MQTT is a versatile way to distribute your data to other systems that can store or process it. You don't need an external service to collect data or process it, and you can keep the data within your own network. One of the tools that offers MQTT connectivity is Node-RED (e.g., on a Raspberry Pi). We have covered a lot on installations for Node-RED in the past, and if you want a quick installation guide for your Raspberry Pi, have a look at the Node-RED website [9]. For MQTT within the $CO_2$-Guard's web interface, you can provide a server and a topic where data should be published (**Figure 13**). The data itself is transferred as JSON string, having the objects "*CO2Value*" and "*Temp*", presenting the $CO_2$ value in ppm and the temperature in °C. If you would like to read more about Node-RED using a Raspberry Pi, have a look at Dogan Ibrahim's book, ***Programming with Node-RED*** (Elektor 2020) [10].

## KiCad PCB Design Files

The PCB was originally designed in Altium, and the provided Gerber files are based on this proven design. The design is quite simple as you can see from the schematics. As the schematic is not that complex (see Figure 1), it was recreated in KiCad. Besides the schematic, the PCB needs to be transferred, but as the project started in the last stable version of KiCad 5, there was no native (integrated) import for Altium boards around. One trick is to use the provided Gerber files and import them into KiCad. The Gerber viewer in KiCad is able to generate board files you can use in the PCB layout tool. Not only does this give the outlines of the board but also the footprints and mounting holes we can use for the component placement. This is a

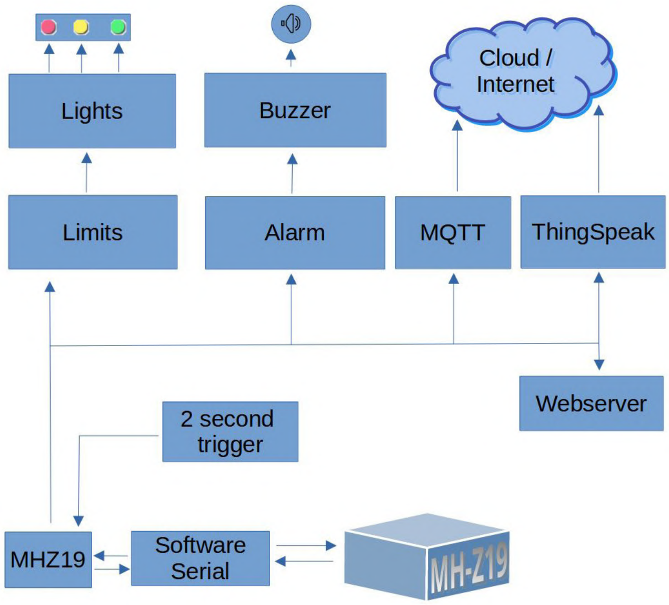


*Figure 13: MQTT Settings.*

way that can be used to import PCBs into KiCad, but it is preferred to use the KiCad version 6 and the native Altium board import if you have Altium files.

But why should someone do this conversion? First, this allows others to directly work with the KiCad files and modify them if wished. Also, with some PCB services it is just drag and drop for a KiCad board file to get your board manufactured (e.g., at Aisler) [11]. You don't have to export to Gerber and drill files to generate a ZIP file. However, if your PCB service needs Gerber and drill files, you are still able to generate them in KiCad to the specifications your service requires.

If you would like to design your own enclosure around the PCB, the combination of KiCad and FreeCad has proven to work very well, as you can see from the Christmas project [12]. And printing an enclosure may be for some more convenient than drilling holes into a premanufactured one. And if you have a 3D printer at home that is collecting dust, why not use it? Also, it means one component less to order. For the 3D-printable case, as at the time of writing this article, it is currently not finished, but it is being worked on. So, in the meantime, if you want to use an enclosure, you need to drill some holes.

## Build Your Own

Building a $CO_2$ sensor for your home or workspace is not that hard, and as you know, many $CO_2$ meters have been developed as both DIY and commercial projects. With the MH-Z19 sensor used in this project, you get decent affordable measurement source. The ESP8266 is a well-proven Wi-Fi microcontroller that not only can be programmed with the Arduino Framework but also offers in the current combination support for Home Assistant.

Want to build your own? You will need to spend around €30 for the sensor (in Europe). The switch, Wemod D1 mini, PCB, LEDs, enclosure, fan and shipping will add to the bill, so you will end up at around €60€ in parts. On the other hand, monitoring the $CO_2$ concentration in your rooms has a lot of advantages. If you know the $CO_2$ concentration, you can ensure that there is enough fresh air.

210043-01

### Questions or Comments?

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

### Contributors

Original hardware design: **Florian Schäffer**
Design and Text: **Mathias Claußen, Ton Giesberts, Luc Lemmens**
Editors: **Jens Nickel, C. J. Abate**
Layout: **Harmen Heida**

## Software

For the software that was originally provided by the developer on the Elektor Labs platform, there were a few areas that could use improvement. For example, hard-coded credentials for your Wi-Fi network work only if you never change the network name or the password the device is connected to. Also having your credentials somewhere in some code files is not the safest way to store them. The same is true for the ThingSpeak API Key. If you need to change this key, it means you need to recompile the whole code and upload it to the device.

What the Elektor lab left hardcoded (as #defines in /configure/configure.h) is the pin mapping, as well as the $CO_2$ tresholds for the three LEDs and the buzzer. In the file /configure/configure.h, you have all hard coded settings in one place and can change them if needed. If you change any settings in this file, you must rebuild the firmware and upload it to the ESP8266, of course.

Inside the software, the dataflow is quite simple, as you can see in the picture. Every two seconds the sensor values are queried and moved back to various code modules. For the local supervising of the $CO_2$ level, we have the limits class, which will control in the end our three LEDs, and we have the alarm class, which will sound the buzzer if a certain level of $CO_2$ is exceeded. Those classes are built in a generic way that makes it easy to change the LED arrangement for a different way of displaying the values. Same goes for the buzzer. If you like to use an active one or even a sound module, this class is flexible enough to accomplish this.

For the cloud connectivity, where possible, pre-built libraries have been used, especially for ThingSpeak. This makes the data exchange with this service a lot simpler than on the ESP32 Weatherstation [13] we published [14] a few years ago.

## COMPONENT LIST

**Resistors**
R1,R2,R3 = 330 Ω, 0.25 W, 5 %
R4 = 4.7 kΩ, 0.25 W, 5 %
R5 = 0 Ω, place wire, see text
R6 = 0 Ω, place wire for 5 V or leave open, see text
R7 = 1.5 kΩ, 0.25 W, 5 %

**Capacitors**
C1,C2 = 100 nF, 50 V, 10 %, X7R, lead spacing 5 mm
C3 = 100 µF, 25 V, optional , not mounted, see text

**Semiconductors**
D1 = BAT85, DO-35
T1 = BC327, PNP, TO-92
LED1 = LED red, 5 mm
LED2 = LED yellow, 5 mm
LED3 = LED green, 5 mm

**Other**
MOD1 = 1x10 SIL header, female, split in 1x4 and 1x5 (see Text)
MOD1 = MH-Z19C, version 400-5000 ppm, pins on bottom
MOD2 = Wemos D1 mini with 2x  1x8 SIL header
S1 = switch with built-in LED (momentary on, self-reset, blue circle)
Enclosure PP73BL, Supertronic
BZ1 = DC buzzer, 5 V, pitch 5 or 7.62 mm, diam. 14 mm max
heat shrink tube, I.D 2.4 mm, 20 cm
LED housing, for 5 mm LED (LED1..3), Black
FAN 5 VDC, 30x30x10.5 mm
12-pin male SIL header
PCB 210043-1 v2.0

**Thin stranded wire to connect LEDs and switch:**
stranded wire, 0.25 mm$^2$, black (2x8.5+12 cm)
stranded wire, 0.25 mm$^2$, red  (8.5+12 cm)
stranded wire, 0.25 mm$^2$, yellow (12 cm)
stranded wire, 0.25 mm$^2$, green (12 cm)
stranded wire, 0.25 mm$^2$, blue (8.5 cm)

## RELATED PRODUCTS

- **WeMos D1 mini Pro – ESP8266 based WiFi Module (SKU 19185)**
  www.elektor.com/19185
- **PCB V2.1 with Components (excluding WeMos D1 and MH-Z19)**
  www.elektormagazine.de/aisler/co2guard
- **H. Henrik Skovgaard, *IoT Home Hacks with ESP8266* (SKU 19158)**
  www.elektor.com/19158
- **D. Ibrahim, *Programming with Node-RED* (E-book) (SKU 19225)**
  www.elektor.com/19225

## WEB LINKS

[1] Elektor GitHub Repository: https://github.com/ElektorLabs/210043-CO2-Guard
[2] Winsen MHZ-19 datasheet: www.winsen-sensor.com/d/files/infrared-gas-sensor/mh-z19b-co2-ver1_0.pdf
[3] “CO2 Guard,” Elektor-Labs.com: www.elektormagazine.de/labs/co2-guard
[4] PlatformIO install guide: https://platformio.org/install/ide?install=vscode
[5] “How to Code (Download) or Clone Something From GitHub,” Elektor TV: www.youtube.com/watch?v=X5e3xQBeqf8
[6] “GitHub Webinar: My Way Into GitHub,” Elektor TV: www.youtube.com/watch?v=j_LgvVhBdwQ
[7] MathWorks ThingSpeak: http://thingspeak.com
[8] MathWorks ThingSpeak Help: https://bit.ly/ChannelPropertiesMATLAB
[9] Node-RED installation guide: https://nodered.org/docs/getting-started/raspberrypi
[10] D. Ibrahim, Programming with Node-RED, Elektor 2020: www.elektor.com/programming-with-node-red-e-book
[11] AISLER B.V: http://aisler.net
[12] M. Claußen, “DIY Christmas Fireplace: A 3D Puzzle with PCBs, LEDs, and Raspberry Pi Pico,” ElektorMagazine.com: www.elektormagazine.com/articles/diy-christmas-fireplace-3d-puzzle-pcbs-leds-raspberry-pi-pico
[13] “ESP32 Weatherstation,” Elektor-Labs.com: www.elektormagazine.com/labs/esp32-weather-station-180468
[14] R. Aarts, “ESP32 Weatherstation,” ElektorMag 1-2/2019: www.elektormagazine.com/magazine/elektor-70/42351

# MonkMakes Air Quality Kit for Raspberry Pi

## Measures Temperature and $eCO_2$

AIR QUALITY
for Raspberry Pi
$eCO_2$
5000
2000
1000
800
500
300
ppm
Download the Instructions by scanning the QR code or visit:
monkmakes.com/downloads/pi_aq

**By Luc Lemmens (Elektor)**

Most of us are now stuck in (private) rooms, so modules for measuring the air quality cheaply are gaining popularity. The MonkMakes Air Quality Kit measures equivalent $CO_2$ and temperature. It is especially designed to be used with a Raspberry Pi 400, but it can also be connected to other Raspberry Pi models using the jumper wires and an included GPIO template.

Thermometers have long been used to monitor room temperature, and in recent years, (e)$CO_2$ meters have become increasingly popular for monitoring air quality. Too much carbon dioxide ($CO_2$) has a negative effect on concentration, and at even higher levels it is bad for your health. This kit measures the quality of the air in a room (how stale the air is) as well as the temperature. It is meant as an add-on for the Raspberry Pi, but it also can be used as a stand-alone device. The board has a buzzer and a bar of six LEDs (two green, two orange, and two red) that displays the air quality. Temperature and air quality readings can be processed by a Raspberry Pi. The buzzer and LED display can be controlled by the host.

The kit comes without printed documentation, but it provides a link to the MonkMakes website where the datasheet and instructions can be downloaded [1]. These documents contain all relevant information; they help the user to connect and use the board. Example applications in Python are available for download on Github [2].

### The Hardware

Besides the six indicator LEDs and the buzzer — the big square component in the middle of the PCB in **Figure 1** — the board contains a power

LED, a temperature sensor, an $eCO_2$ sensor, a microcontroller and of course a 40-pin connector that fits directly on the expansion connector of a Raspberry Pi 400 (**Figure 2**). Perhaps needless to say: other Raspberry Pi boards can not be connected directly, for that the jumper wires are included. The four connections needed (two for the power supply, two for the serial connection) are shown in the print on the MonkMakes board and on the supplied template to match the corresponding pins of the GPIO connector of the Raspberry Pi, as illustrated in **Figure 3.** The power LED lights up as soon as the 3.3 V supply voltage is switched on, and so will one of the $eCO_2$ level LEDs.

The temperature sensor is a Texas Instruments TMP235 [3]. Its output voltage is proportional to the temperature. For $CO_2$ measurement, the MonkMakes board uses a CCS811 TVOC (Total Volatile Organic Compounds) sensor [4]. This does not actually measure the level of $CO_2$, but rather the level of a group of gasses called Volatile Organic Compounds (VOCs). When indoors, the level of these gasses rises at a rate comparable to that of $CO_2$, and can therefore be used to estimate the level of $CO_2$ (called the equivalent $CO_2$ or $eCO_2$).

The on-board ATtiny1614 microcontroller reads both sensors and controls the LED bar display and the buzzer. Via a serial protocol, a host system can request sensor readings or switch the LEDs and buzzer on and off. The kit's datasheet documents this simple protocol, so it will not be too difficult to write your own software to support the Air Quality Kit. As name of this kit indicates, it is designed for Raspberry Pi, but there is no reason why you shouldn't use it with other boards or systems with a 3.3 V UART.

The firmware of the ATtiny also offers an automatic mode (switched on by default) that shows the $eCO_2$ level on the LED bar without any external control; all that is needed is a 3.3 V power supply. So, even without a host system, the Air Quality kit can be used as $eCO_2$ monitor.

## Software

As mentioned earlier, MonkMakes offers downloads of some Python example programs to control this Air Quality Kit to test and demonstrate all its features. In the *Getting Started* section of the documentation, the instructions clearly show how the software can be used on a Raspberry Pi board to implement an $eCO_2$-meter, an $eCO_2$-meter with acoustic alarm (**Figure 4**) and a data logger application. Looking at the examples, like in **Figure 5**, you'll notice that the ATtiny and the API completely relieve you of retrieving and evaluating sensor data: one simple instruction from the host (Raspberry Pi) will trigger the Air Quality board to echo the current ambient temperature (in °C) or $CO_2$ level (in ppm), respectively. Similar commands are used to switch the buzzer on and off, and to control the LEDs of the $eCO_2$ bar display.

## A Nice Design

Only some basic knowledge of the Raspberry Pi is required to get this Air Quality kit working. What is a great advantage for some, will be less attractive for others: knowledge of the sensors and control of buzzer and LEDs is not required. The (source code of the) firmware of the on-board

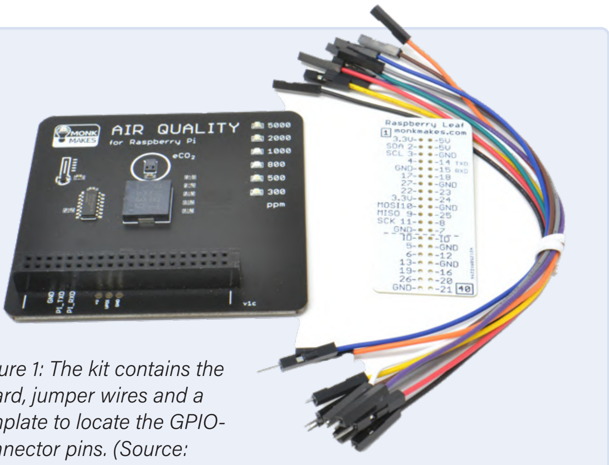


*Figure 1: The kit contains the board, jumper wires and a template to locate the GPIO-connector pins. (Source: MonkMakes)*

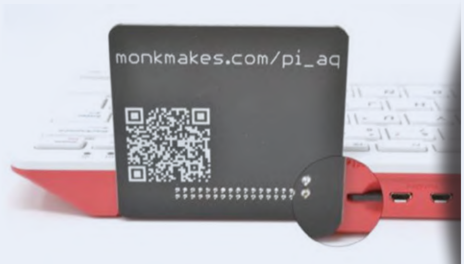


*Figure 2: Air Quality Kit connected to a Raspberry Pi 400. (Source: MonkMakes)*

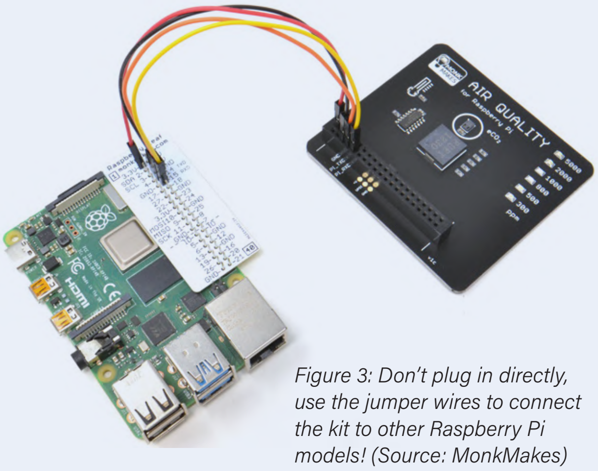
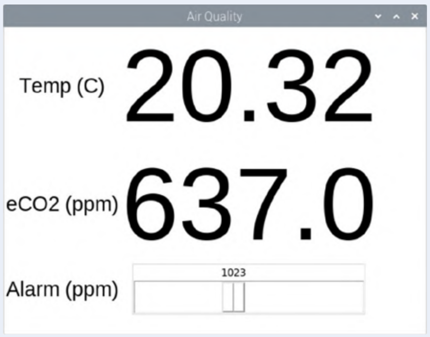


*Figure 3: Don't plug in directly, use the jumper wires to connect the kit to other Raspberry Pi models! (Source: MonkMakes)*



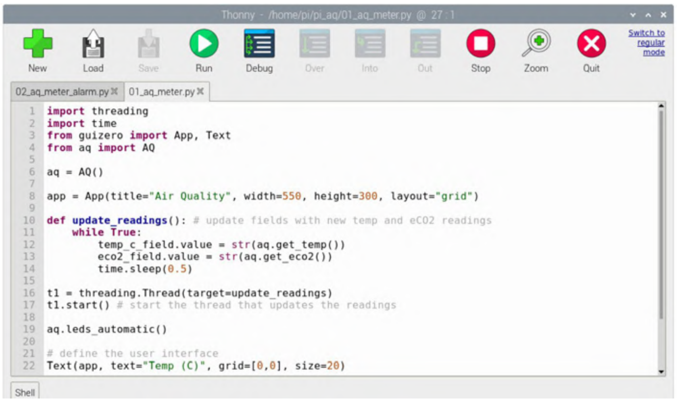
Air Quality

Temp (C) 20.32

eCO2 (ppm) 637.0

Alarm (ppm) 1023

*Figure 4: Raspberry Pi screen output for one of the examples. (Source: MonkMakes)*



```python
import threading
import time
from guizero import App, Text
from aq import AQ

aq = AQ()

app = App(title="Air Quality", width=550, height=300, layout="grid")

def update_readings(): # update fields with new temp and eCO2 readings
    while True:
        temp_c_field.value = str(aq.get_temp())
        eco2_field.value = str(aq.get_eco2())
        time.sleep(0.5)

t1 = threading.Thread(target=update_readings)
t1.start() # start the thread that updates the readings

aq.leds_automatic()

# define the user interface
Text(app, text="Temp (C)", grid=[0,0], size=20)
```

*Figure 5: Python source, showing that only simple, short instructions are needed to communicate with the Air Quality Kit.*

ATtiny1614 is not released (i.e., we don't know what exactly happens inside this microcontroller). However, the protocol to communicate with the board is simple and well documented, and development of your own applications — even for other target systems than Raspberry Pi boards — will be relatively easy. The MonkMakes Air Quality Kit for Raspberry Pi is a nicely designed, well-documented board that, together with the examples, is also suitable for beginners who want to get started with temperature and $eCO_2$ measurements.

210681-01

## Questions or Comments?

Do you have questions or comments about his article? Email the author at luc.lemmens@elektor.com or contact Elektor at editor@elektor.com.

## Contributors

Text: **Luc Lemmens**
Illustrations: **MonkMakes, Luc Lemmens**
Editors: **Jens Nickel, C. J. Abate**
Layout: **Harmen Heida**

## RELATED PRODUCTS

- **MonkMakes Air Quality Kit for Raspberry Pi (SKU 19913)**
  www.elektor.com/19913
- **Raspberry Pi 400 – Raspberry Pi 4-based PC (US) + FREE GPIO Header (SKU 19429)**
  www.elektor.com/19429
- **Raspberry Pi 4 B (1 GB RAM) (SKU 18966)**
  www.elektor.com/18966

## Sensor Characteristics

| | | |
|---|---|---|
| $eCO_2$ minimum reading | 400 | ppm |
| $eCO_2$ maximum reading | 4095 | ppm |
| $eCO_2$ resolution | 1 | ppm |
| $eCO_2$ accuracy | unspecified | |
| Temperature minimum reading | -10 | °C |
| Temperature max reading | 100 | °C |
| Temperature accuracy | +/- 2 | °C |

## About $CO_2$ Concentrations

The level of $CO_2$ in the air we breathe has a direct influence on our well-being. $CO_2$ levels are of particular interest from a public health point of view. To put it simply, they are a measure of how much we are breathing other people's air. We humans breathe out $CO_2$, and if several people are in a poorly ventilated room, the level of $CO_2$ will gradually increase — as well as concentration of aerosols that spread colds, flues and Coronavirus. Another important impact of $CO_2$ levels is in cognitive function — how well you can concentrate and think.

The table below shows the levels at which $CO_2$ can become unhealthy. The $CO_2$ readings are in ppm (parts per million).

### Level of $CO_2$

| | |
|---|---|
| 250-400 | Normal concentration in ambient air. |
| 400-1000 | Concentrations typical of occupied indoor spaces with good air exchange. |
| 1000-2000 | Complaints of drowsiness and poor air. |
| 2000-5000 | Headaches, sleepiness and stagnant, stale, stuffy air. Poor concentration, loss of attention, increased heart rate and slight nausea may also be present. |
| 5000 | Workplace exposure limit in most countries. |
| >40000 | Exposure may lead to serious oxygen deprivation resulting in permanent brain damage, coma, even death |

## WEB LINKS

[1] MonkMakes webpage with instructions: http://monkmakes.com/pi_aq
[2] Software on Github: https://github.com/monkmakes/pi_aq
[3] TMP235 datasheet: https://www.ti.com/product/TMP235
[4] CCS811 datasheet: https://www.sciosense.com/products/environmental-sensors/ccs811-gas-sensor-solution/

# Starting Out in Electronics

## Welcome to the Diode

By Eric Bogers (Elektor)

It has taken a while, but we have finally wrapped up our discussion about the 'passive' components. In this instalment we begin our discussion with a few 'active' components – the semiconductors, to be precise. Now it really becomes interesting!

We commence our discussion of the active components with the diode and related components. Granted, you could argue endlessly about the question whether a diode is an active or a passive component — active components are called that because they are able to amplify a signal, and a diode does not do that. But because they are semiconductors components (or, in the past, electron tubes, but these are outside the scope of this series of articles) we put them in the same heap with transistors and other active components.

If we were writing a 'real' textbook, we would have to fill many pages with information about semiconductor materials (silicon, germanium, selenium and others), doping, PN-junctions and many other things; but this series of articles is targetted at the beginning electronics enthusiast and not at physicists or semiconductor manufacturers. The beginning electronics enthusiast is mostly interested in the question of what function a particular component accomplishes and how they can use it in their designs — and much less in the question of how and why a particular component does what it does.

In **Figure 1** you can see a few common implementations of diodes. At top left are two 'ordinary' diodes: a 1N4148 and a 1N4001. At bottom left are three LEDs (*light emitting diodes*), with diameters of 3 mm, 5 mm and 10 mm. On the right you can spy a set of four bridge rectifiers — these are combinations of four diodes in a single housing. These are used in large numbers in (mains) power supplies. The example at bottom right has a metal housing and is intended to be bolted to a heatsink; this rectifier can handle a current of no less than 25 A.

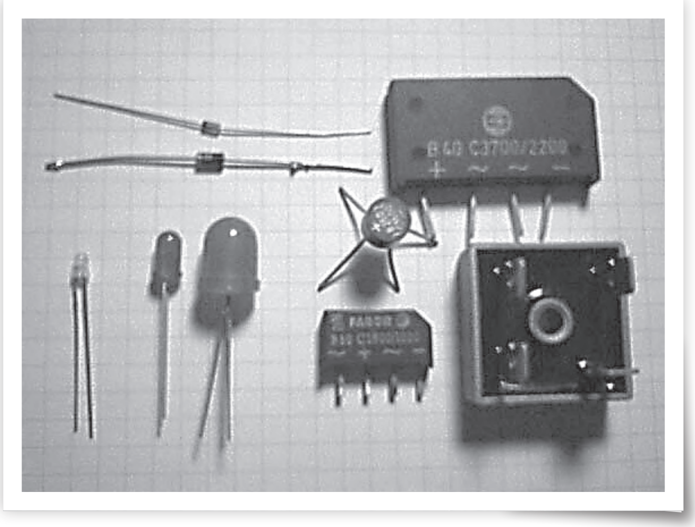


*Figure 1: Several diodes and bridge rectifiers.*

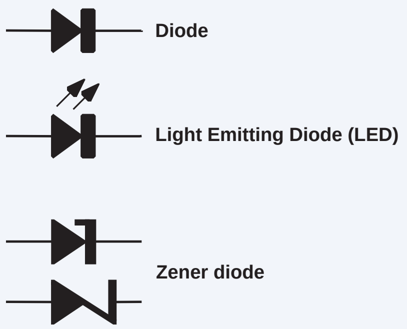


*Figure 2: Schematic symbols for a few common diodes.*

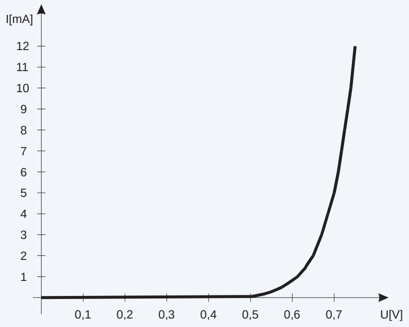


*Figure 3: The V/I-characteristic of a diode.*

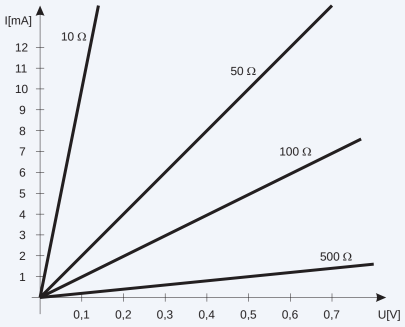


*Figure 4: V/I-characteristics of four resistors.*

## The Diode

A diode can be considered as a kind of valve for electric current: the component conducts in one direction but not in the other. In **Figure 2** you can see the schematic symbols for a few diodes; from top to bottom: an ordinary diode, a light-emitting diode (LED) and below that are two versions of the symbol for a Zener diode (in the bottom symbol the 'Z' for Zener can be recognised). The diode symbol resembles an arrow that points in the direction in which the current can flow. The connection on the left in Figure 2 is called the anode and the one on the right (the bar) is called the cathode.

A (silicon) diode starts to conduct once the voltage at the anode is about 0.6 V more positive than at the cathode — somewhat comparable to a mechanical valve where a certain pressure difference is required before the valve opens. And before you ask where this 0.6 V comes from: this is determined by the semiconductor material, in this case silicon. In a germanium diode this minimum voltage amounts to about 0.3 V. A physicist can undoubtedly explain in fine detail why exactly it is that way, but here it is of no importance to us. We only remember these two voltages: 0.6 V for silicon and 0.3 V for germanium.

It becomes more interesting when we make a graph of the current through a diode as a function of the voltage across the diode. **Figure 3** gives an example. In such a V/I-diagram the voltage is plotted along the horizontal axis and the current along the vertical axis. As a comparison, **Figure 4** shows a V/I-characteristic of a few ordinary resistors. With a resistor that has a small value this line is steep, with a larger value it is much more horizontal.

When we look closely at Figure 3, we see that up to a voltage of 0.5 V barely anything happens — practically no current flows through the diode. Beyond 0.5 V the curve quickly becomes increasingly steep; the slope from this point onwards is mainly determined by the internal resistance of the diode.

For completeness, the curve of Figure 3 was measured using a silicon diode of the type 1N4148, but the V/I-characteristics of other Si-diodes show a very similar curve.

## The Specifications of a Diode

The manufacturer of a diode will at a minimum specify the maximum current capability in the forward direction and the maximum voltage in the reverse (blocking) direction (where the cathode voltage is more positive than the anode). Often the maximum power handling (the maximum dissipation) of a diode is also mentioned.

The maximum current capability of the 1N4148 (a 'typical' small-signal diode) that is used as the example here, amounts to 100 mA, while the maximum dissipation is 500 mW.

On the other hand, a diode is not able to block arbitrarily high voltages either. When the maximum blocking voltage is exceeded, the diode also starts to conduct in the reverse direction. If the current is not severely limited in this case, the diode will overheat and the silicon semiconductor crystal will melt — after which the diode will forever conduct in both directions... This higher voltage also results in a much higher dissipation, so that the diode will be destroyed by a comparatively small current in the reverse direction.

## Regulation of Voltages

Using ordinary diodes a reasonably stable voltage can be obtained in a very simple way (see **Figure 5**). First, the circuit on the left in Figure 5: here a diode is connected via a series resistor of 1 kΩ to a DC voltage source. At a voltage of 7.5 V there flows a current of 6.68 mA, and the voltage drop across the diode amounts to 0.715 V. We now double the input voltage (the power supply voltage) to 15 V. The current will obviously increase, to 14.12 mA, but the voltage across the diode only increases a small amount, to 0.761 V. Conclusion: when doubling the power supply voltage, the voltage across the diode increases by only about 6%.

When we use a 'two-step circuit' (Figure 5, right), the voltage is regulated even better. Using the three diodes in series a voltage of about 2 V is generated. With this already reasonably regulated voltage a second diode is powered.

Here, at an input voltage of 7.5 V we measure a voltage of 0.617 V across the second diode and at an input voltage of 15 V we measure a voltage of 0.662 V. So, a doubling of the input voltage (that is, an increase of 100%) in this circuit results in an increased voltage across the second diode of only 0.8%.

Regulation of voltages using ordinary diodes is only used when

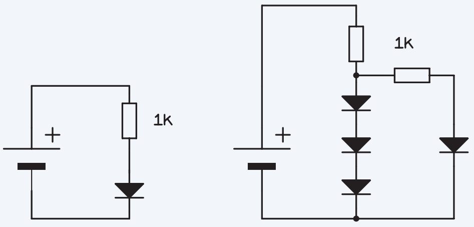


Figure 5: Voltage regulation with ordinary diodes.

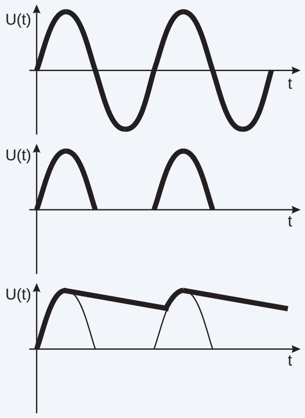


Figure 7: Voltage wave shapes for the single-sided (half-wave) rectifier.

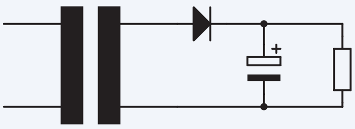


Figure 6: Single-sided (half-wave) rectifier.

we need very low voltages. For higher regulated voltages, we use Zener diodes, which will feature in the next instalment.

## Power Supply Circuits

One of the most important application of diodes is that of rectifier in power supply circuits. Electronic circuits (devices) require a DC voltage for their power supply that, depending on the device, can be anywhere between 1.5 V and 150 V. However, the mains only supplies an AC voltage of 230 V. This voltage therefore has to be transformed to an acceptably low voltage first, and then has to be rectified and smoothed, and in most cases will also need to be regulated.

## One-Sided Rectifiers

The term 'one-sided' (see **Figure 6**) has nothing to do with sides, but means that only one half of the two half periods of the AC input voltage is let through — and only the positive half period — and then only for the relatively short time that the transformer output voltage is higher than the voltage across the electrolytic capacitor plus the forward voltage drop of the diode (which amounts to about 0.7 V). This type of circuit is appropriately called a half-wave rectifier.

We have sketched this in **Figure 7**. The top graph shows the AC voltage as it is supplied by the transformer. If we think away the buffer or smoothing electrolytic capacitor for a moment, then across the resistor (the load) there will be a pulsing DC voltage, as is drawn in the middle graph.

The bottommost graph is actually the most interesting. This graph shows the voltage across the buffer capacitor. During the positive half periods of the AC voltage this capacitor is charged very briefly each cycle; during the remainder of the cycle the capacitor is discharged by the load. In practice this load is generally not a simple resistor but a complete electronic circuit that likes to be powered from a DC voltage.

In the next instalment, we will start to calculate things for these (and other) rectifiers.

220003-01

Editor's Note: The series of articles "Starting Out in Electronics" is based on the book *Basiskurs Elektronik*, by Michael Ebner, which was published in German and Dutch by Elektor.

## Contributors

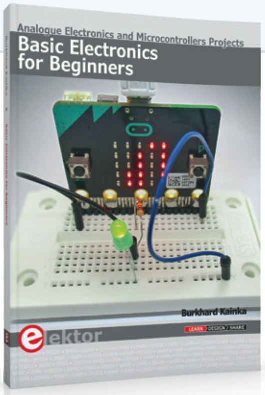
Idea and illustrations: Michael Ebner
Text and editing: Eric Bogers
Translation: Arthur de Beun
Layout: Giel Dols

## Questions or Comments?

Do you have any technical questions or comments prompted by this article? Send an email to the editor via editor@elektor.com.

## 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


Analogue Electronics and Microcontrollers Projects
Basic Electronics for Beginners
Burkhard Kainka
elektor

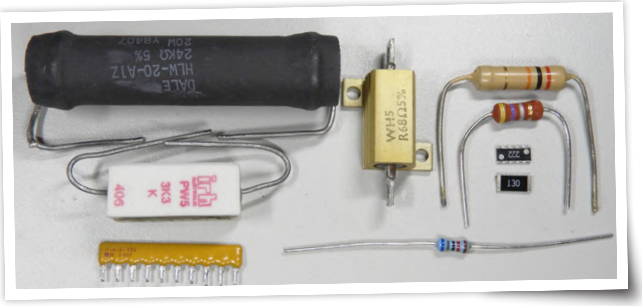
elektorstore
What are you looking for?
NEW
Elektor Products
Measuring Tools
Components
Platforms
Raspberry Pi
Arduino
ESPressif
Books
Sale %
NEW in the Store
143 results found
Sort by Relevance
Brands
Pre-order
PICAN 3 - CAN-Bus Board for Raspberry Pi 4 with SMPS & RTC
JOY-iT JT-RD6006 DC Power Supply (360 W)
Get Started with MicroPython on Raspberry Pi Pico
Raspberry Pi Pico Microcontroller Board
Elektor January/February 2021
DVD Elektor 2020
Elektor Magazine Box 2019 incl. DVD (EN)
Elektor Magazine Box 2018 incl. DVD (EN)

# Tips & Tricks for Testing Components

## No Expensive Equipment Required

By David Ashton (Australia)

In my article on component identification, I touched on the importance of component testing skills [1]. As a hobbyist, you usually just want to verify that a component works before soldering it into your circuit. There are a number of techniques that can be used to test components that have been stripped off boards or of otherwise uncertain provenance, from your spares box, that don't need fancy or expensive test equipment.



### Resistors

Digital multimeters are now so cheap and ubiquitous that there is little point in making your own testers. There are exceptions — very low-value resistors (by which I mean under 1 Ω or so) used for current sensing and other applications, cannot be reliably tested with cheap digital meters, as the resistance of the test leads is in the same ballpark as the resistor to be tested.

A more reliable method is to put a known current through the resistor and measure the voltage across it. There are many designs in past Elektor articles and on the internet for milliohm meters, as they are called, and some are simple and will give acceptable results for a quick test.

Resistors that have got hot to the point of discoloring the body or markings should usually not be kept but sometimes you may want to test and keep them.

## Capacitors and Inductors

When I was about 20 (that was many years ago now), I built myself a capacitance meter. A 555 astable triggered a 74121 monostable with the test capacitor to give a pulse train with duty cycle proportional to the test capacitor. I used a preset pot on each range and used 1% capacitors to calibrate it. It measured from around 1 pf to 100 μF and is one of the most useful gadgets I've ever built.

Many digital multimeters now have a capacitance test function, and I found one in a recent electronics magazine for around €40 ($40/£30) that measures capacitance and inductance, albeit with fairly limited ranges. I had a basic but adequate LCR meter that was about the same price but recently invested in a more versatile one. You very much get what you pay for with this type of test equipment and what you get will depend on your budget and requirements. It's really not worth building your own testers anymore. But in some circumstances, you may need to make more esoteric measurements, such as the Equivalent Series Resistance (ESR) of a capacitor, or the *Q* of an inductor, and for this, you may need more advanced test equipment. Elektor has published many designs over the years for devices to measure all aspects of capacitance and inductance and a search on the magazine site may lead you to a design that fulfils your needs. The Elektor store has some reasonably priced testers as well (see textbox).

Supercapacitors are just a special case of electrolytic capacitors but beyond the range of most testers. They are usually rated at 5.5 V, so be careful not to exceed this. Test them using a resistor of 100 Ω to a 5 V supply, and count the number of seconds for a fully discharged cap to get to 3.5 V. Divide that by 100 and you've got a rough estimate of the capacitance in farads. This uses the RC time constant — the time for the capacitor to reach 0.7 of full charge (5 V x 0.7 = 3.5 V). A 1-F capacitor will take 100 Ω x 1 F = 100 seconds to get to 3.5 V. 100/100 = 1, so 1 farad.

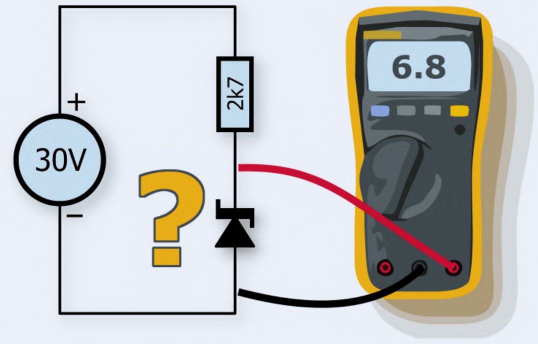
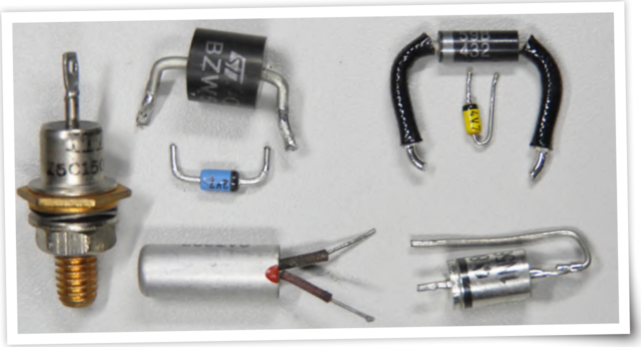


*Figure 1: A way to quickly test Zener diodes.*

## Transformers

Transformers are just a special case of inductors. Identify the windings with a multimeter, then get the inductances. For a small power transformer, any winding with resistance above about 50 Ω and inductance over about 10 H may be a mains winding. And as inductance is proportional to the square of the number of turns, take the square root of the ratio of the windings' inductances to get the transformer ratio.

For example, a transformer with inductances of 10 H and 50 mH has a turns ratio of around $(10 / 0.05)^{0.5}$ = 14, so if it's a 240 VAC mains transformer, you'd expect the secondary to give about 15 V. But many mains windings will have a higher inductance than your LCR meter will measure, so you can't always do this.



## Diodes

Diodes are easy to test with any multimeter — they conduct one way and not the other. Digital multimeters may have a diode test range and this will give you an indication of the voltage across the diode when conducting — around 0.6 to 0.7 means it is a standard silicon diode, 0.3 to 0.5 means it is a Schottky diode, and as low as 0.2 means a germanium diode. The position of the decimal point may vary, but it's the first figure you need to look at.

Zener diodes are a bit more difficult. They will behave as normal diodes when tested with a multimeter, as a multimeter is unlikely to apply enough voltage to make it conduct in reverse (Zener) mode. To test Zeners, use your lab power supply on its maximum voltage, with a very low current limit (say 10 mA), or a series resistor that will pass around 10 mA at the full voltage. Apply this to the Zener and measure the voltage across it with your multimeter (**Figure 1**). This will only work for Zeners with voltages less than your power supply, but a 30 V power supply will cover most Zeners.

LEDs can be tested much like Zeners, but turn the power supply voltage down to 5 V or so — LEDs have quite a low reverse breakdown voltage. Test any LED with a clear case both ways — it may be a bicolor type. Limit the current to 10 mA; it is enough to make any LED light and is safe even for small LEDs.

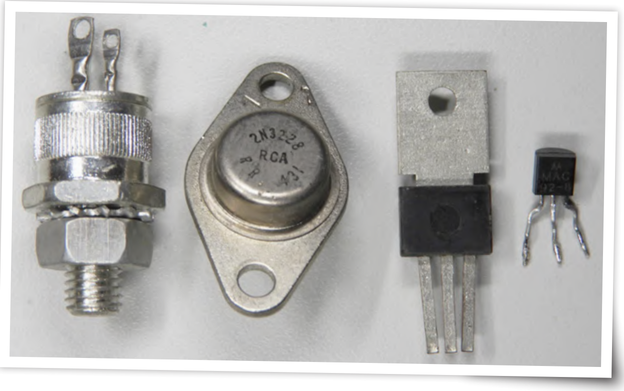
## Transistors

When I was about 13, I saw an article somewhere for a simple transistor tester, checking function and giving an $H_{fe}$ (gain) value. It gave reasonable accuracy on small-signal silicon transistors, but was inaccurate for germanium transistors — hardly a problem these days! It has had a lot of use and I still use it today. But again, most multimeters have a transistor tester built in, and generally, they do a good job.

Bear in mind also that a transistor behaves as two diodes, so identifying the base is fairly easy. Transistors like Darlingtons and power transistors are more difficult to test as they need higher base-emitter voltages or larger collector currents to do a proper test. But for the hobbyist, a basic functional "go/no go" test is all that's needed.

## MOSFETs

MOSFETs have become ubiquitous, and a simple transistor tester will not be able to test them. Most of them require gate-source voltages of 5 V or above to switch them on fully. Fortunately, there is an easy way to do a functional test of a MOSFET using only a current-limited power supply (or a power supply and a resistor, or even better a light bulb) and your fingers or a high-value resistor (a megohm or so). Putting the resistor (or your fingers) gate-to-positive should switch it on and gate-to-source should switch it off. Because MOSFETs have appreciable capacitance on the gate, it may take a second or more to switch off and on. Use 12 V to apply enough gate voltage, and limit the current to 100 mA or less, or you may burn your fingers if the MOSFET is not on a heatsink!



## Thyristors/SCRs/Triacs

For these you can use the same setup as for MOSFETs above, except that you will need a smaller resistor for the gate — around 470 Ω to 1 kΩ should be fine. When you connect the gate, via the resistor, to the anode or the positive supply, the SCR should turn on (draw current) and continue to do so even after the gate drive is removed. You have to interrupt the anode connection to switch it off. Triacs are the same but will work with the power both ways.

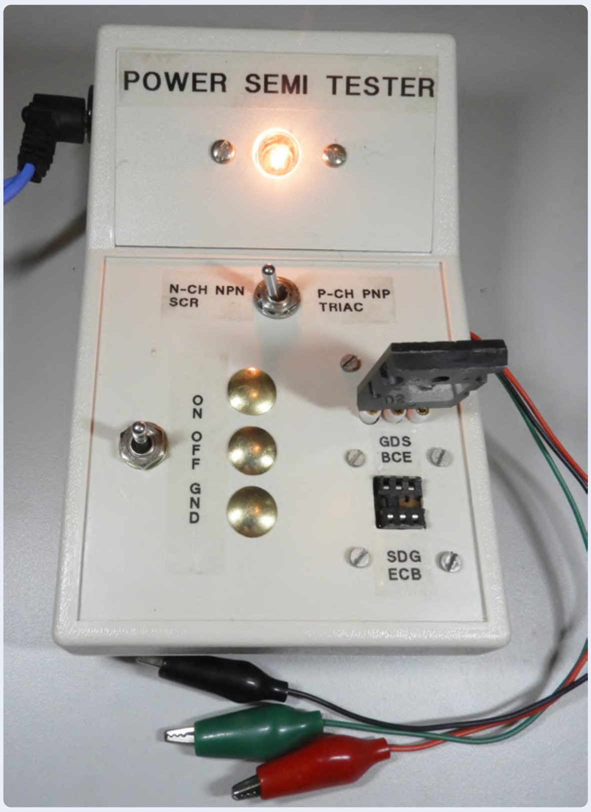


*Figure 2: A simple Power Semiconductor Tester tests MOSFETs, transistors, SCRs and Triacs.*

The test requirements for MOSFETs, transistors, and SCRs are similar, and you can build yourself a basic tester that will test all of them. The one I built is shown in **Figure 2**.

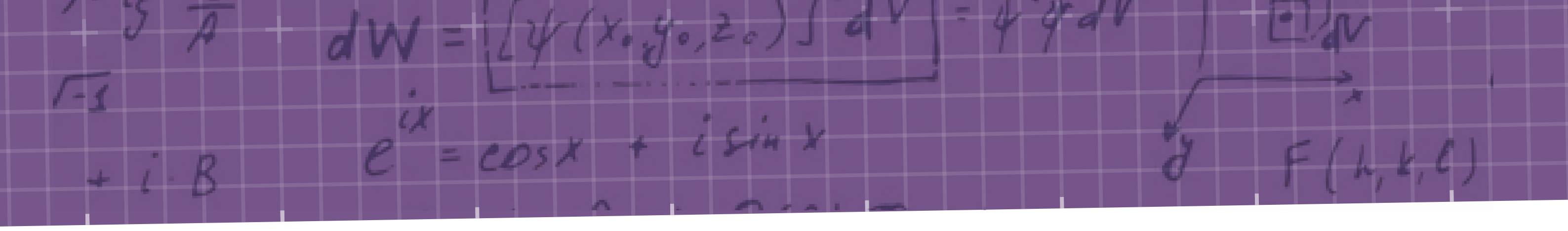
## VDRs

Voltage-Dependent Resistors (VDR), sometimes called tranzorbs, and other surge protectors such as the gas types and the P6KE semiconductor ones are among the more difficult components to test. A 460-volt VDR will only start to conduct at over 500 V. But, if you have access to an electrician's megger or insulation tester, you can do a quick check of these.

Most insulation testers have three test voltages: 250 V, 500 V, and 1000 V. A 460 V VDR should give no deflection or reading at 250 V, some deflection at 500 V and show low resistance at 1000 V. Results may vary with the model of tester used but should be good enough for a go / no go test. Non-working VDRs are either shorted or completely open-circuit, and this test will show either of those conditions. And keep your fingers away from those voltages!

## Polyfuses or PTC Resistors

These are devices that usually have low resistance (a few ohms), but when more than the rated current flows through them, they heat up and go high resistance and will usually stay there until the source of the overcurrent is removed. Test them by putting them on a current-limited power supply and slowly increasing the current. Initially, the voltage



across them will be small, but as the current increases, they increase their resistance and the voltage across them will increase fairly fast. Using this technique, you can get an idea of their rated current (which will be less than the point that causes them to go high-resistance). And don't burn your fingers; they increase their resistance by getting hot!

## DIP Switches

Used DIP switches may have some switches faulty so test them before use. You can test any switch with a multimeter of course, but if you have a lot of them to test, you can make a DIP switch tester with an IC socket, resistors, and LEDs to indicate if each switch pole is open or closed.

In addition, you can use it for testing LEDs. Put an LED where one of the switch poles should be and both it and the tester LED should light if it's good. One I built is shown in **Figure 3**.

## Integrated Circuits

These are more difficult to test, as there are so many varieties. But there are easy tests you can do. Op-amps usually have the same pinouts, so you can make a simple op-amp tester by configuring it to work as an astable multivibrator with a couple of LEDs on the output. It will work for many comparators as well, but bear in mind that some comparators only have a bare transistor on the output and will not source current.

You could make testers for double and quad op-amps as well. Use a decent quality socket — a Zero Insertion Force (ZIF) type if your budget runs to that. You can do the same for the much-loved 555 — have it running as an astable with LEDs on the output.

Digital ICs are more difficult. If you need to test these a lot, your best course of action is to buy a suitable tester. There is a device called the TL866 [2] (see inset), usually widely available at under $50, which is an EPROM programmer but also tests logic ICs. There are a few different models available so look at the specifications before you buy. The TL866II is the current version.

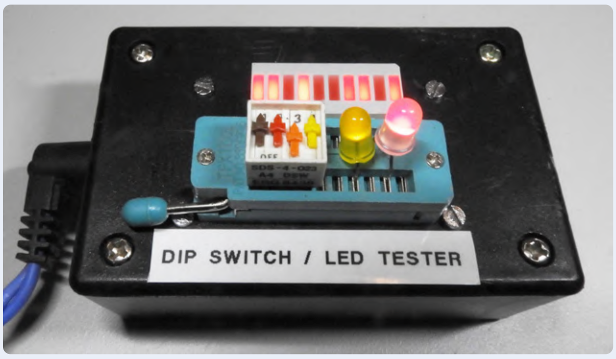
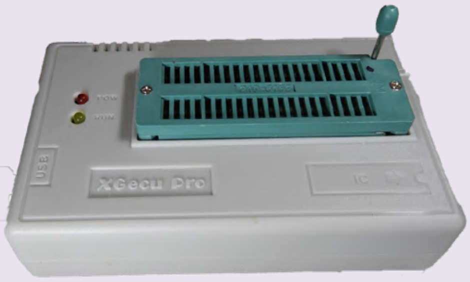


*Figure 3: A DIY DIP Switch / LED tester shown testing a 4-way DIP switch and a couple of LEDs.*

## The TL866 Programmer & Digital IC Tester



*Source: David Ashton*

The TL866 is primarily a programmer of a very wide range of devices. At this it seems reasonably competent and many reviews are available, including Elektor's [6]. However, I will concentrate on its ability to test Logic ICs.

The TL866-II – the current version - presents in a plastic case with a good quality 40-pin ZIF socket and a couple of LEDs, and a USB connection. I had to download the software from the manufacturer's site – something I'm not keen on, but it is the manufacturer's site so the risk is not high. The TL866 has a good printable PDF manual. The software is fairly well laid out and you can quickly find most devices and functions.

In the Device tab on the main menu is the item Logic Test. This brings up a long list of CMOS 4000 and TTL 74 series IC types to choose from, and even some Intel 8080 peripheral devices. I spent a happy half hour with my boxes of CMOS and TTL ICs trying a good many of them and the TL866 tested most of them with no problems. However 74LS21s consistently brought up an error - a bit disappointing on such a basic IC.

But now for the fun part. The TL866 software displays the test sequence it uses, and you can copy and edit them (though you have to store them with a new name). Looking at the 7421 datasheet and the programmed test sequence, I noticed that pins 3 and 11 had "H" (high level expected) marked for those pins. These pins are not used, so they should be ignored. If I changed those pins to "x" (ignore) the test worked, and I successfully tested all my 74LS21s with no errors. I'm now confident I can test just about any digital IC, even if I have to write my own test sequences for some of them. Some other ICs gave errors, primarily monostable multivibrators which would be difficult to test without timing components.

For testing SMD ICs, you can find sets of SMD-to-DIL adapters on eBay and other sites, and in fact you can buy a TL866 with various adapters included for less than $50 – a very handy tester for a good price, and a good addition to any hobbyist's collection of test gear.

They come with a ZIF socket for most sizes of through-hole ICs but often come with handy adapters for testing various SMD package components as well. Some models do ICSP programming if that's useful for you, but the newer ones don't program very old EPROMs. (For this old-timer, that's important!) For just testing digital ICs neither of those is very important. You can write your own testing programs too, for ICs not in the database or those for which the included programs don't work (there are a few).

## Quartz Crystals

Quartz crystals and oscillators are some of the more common components you will find when stripping professional equipment. Usually, these components are marked with their frequency and a simple go/no go test is required. This can be done fairly simply. To verify the frequency, or to test crystals that are not marked, a frequency counter will be needed. You need only two inverting gates to make a crystal oscillate, and another one to detect that it is oscillating. You can use the 4th gate in a quad gate IC as a buffer to feed a frequency counter.

32,768-Hz crystals are commonly used in real-time clocks, but crystals for microcontrollers are typically in the MHz region, so you may need two oscillators to check all crystals. Then add a DIP socket to check oscillators; you just need power and a connection to the output. SMD crystals and oscillators are more difficult. You may need a test jig for them.

## Relays

Relays are still widely used. For isolation and high current handling they can't be beaten. Test the contacts with a multimeter with a continuity buzzer which will show low resistance, and operate the relay with your power supply set to the appropriate voltage and make sure the contacts open or close. If you have a lot of relays of one type it may be worth making a test jig for them, using a load of up to a few amps (e.g., a car light bulb) to weed out any with high resistance contacts.

## Other Tools

If you can only afford one piece of good test equipment, make it a decent multimeter, with transistor test, capacitance, and other ranges (frequency, temperature...). It will be really useful until you can buy or build better test equipment.

One of the most useful things you can buy if you do any SMD work is a tweezer-style probe (**Figure 4**). This plugs into your test meter and allows you to very quickly and easily test SMD resistors, capacitors, inductors, and diodes. They are made by Sparkfun and are widely available. You can get meters built into tweezers if you really do a lot of SMD testing. For example, the DT71 is available in the Elektor Store (see textbox) and a good review is at [3].

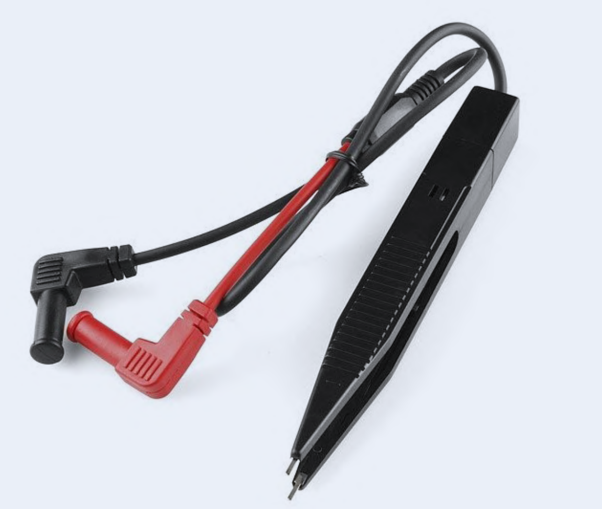


*Figure 4: Tweezer probe for multimeter. (Source: Sparkfun)*

Peak makes a line of "Atlas" hand-held component testers which have good reviews and are reasonably priced. In particular, their DCA75 Semiconductor Analyser (see inset) will identify just about any 2- or 3-terminal semiconductor device and is highly recommended. If you need to do more detailed tests than the simple tests outlined above, it is worth investing in test gear like this. Elektor has recently started stocking the Joy-IT LCR-T7 Multi-function Component Tester [5] which, while not offering the specifications of Peak's tester, is about a quarter of the price and represents outstanding value.

A desk or hand-held lens with LED illumination, and/or a USB microscope, is handy for checking small components for value, polarity, or orientation.

A power supply has been mentioned above in many of the tests. At a minimum, a 0-30 variable supply with a variable current limit of 10 mA or less up to 500 mA or more will be adequate.

Using these simple techniques and some DIY test gear will make your life a lot easier if you re-use components for your projects.

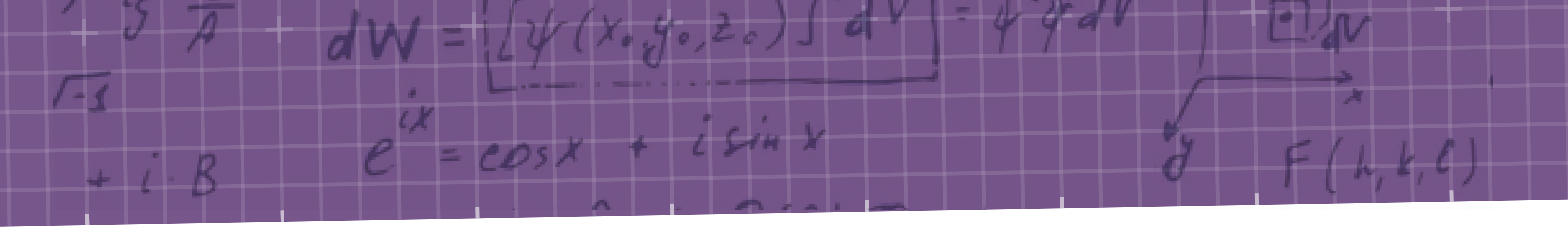
210279-01

## Contributors

Text and Photographs: **David Ashton**
Editor: **Clemens Valens**
Layout: **Harmen Heida**

## Questions or Comments?

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



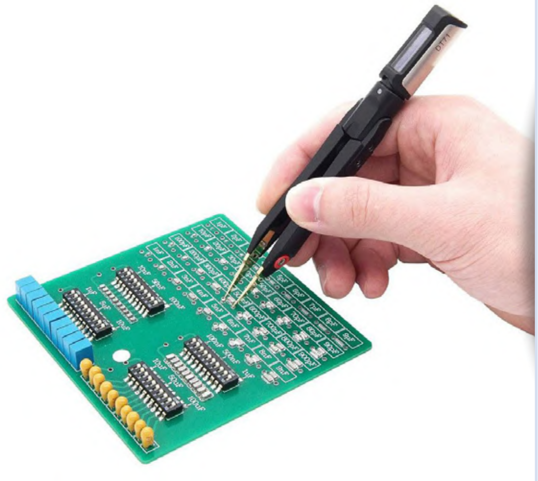
## Peak DCA75 Advanced Semiconductor Analyzer Review

Some time ago I splashed out on a Peak DCA75 Advanced Semiconductor Analyzer. I was a little hesitant, I will admit, because I wondered how such a small tester with only two buttons could be worth what it costs.

I could not have been more pleasantly surprised, however. It does everything it claims and is simplicity itself to use. The interface is sheer genius and hardly needs you to do anything. You connect up your device, any way you like, and the DCA75 will identify what you have connected, give you some basic characteristics, and tell you which lead is which.

Try as I might I could not fox the DCA75. One of the first things I tried was a 0V8F "transistor" from my article on Component Identification [1]. I had not been able to identify or test them- I thought they might be FETs. The DCA75 instantly identified them as Triacs. This alone convinced me that the DCA75 is worth its weight in gold! The DCA75 does have some minor limitations – it cannot test Zeners and voltage regulators of above 8 or 9 volts – and will not identify some of the more esoteric semiconductor devices like Unijunction transistors, but these are not serious drawbacks for the average user.

Using a PC and the included USB cable and software, the DCA75 can also produce characteristic curves of many devices. All in all a really versatile bit of test gear to add to your arsenal. There is a small brother, the DCA55, at about half the price, but my advice would be to stump up for the DCA75. I cannot recommend this tester too highly.



### RELATED PRODUCTS

- **Peak Atlas DCA75 Pro Advanced Semiconductor Analyser (SKU 17567)**
  www.elektor.com/17567
- **Miniware DT71 Digital Tweezers (SKU 19422)**
  www.elektor.com/19422
- **Joy-IT LCR-T7 Multi-function Component Tester (SKU 19709)**
  www.elektor.com/19709
- **Elektor 2 MHz LCR Meter Kit (SKU 19883)**
  www.elektor.com/19883

### WEB LINKS

[1] D. Ashton, "Component Identification," Elektor 3-4/2022 : https://www.elektormagazine.com/210024-01

[2] Review: MiniPro TL866A programmer: https://www.elektormagazine.com/news/review-minipro-tl866a-programmer/14585

[3] H. Baggen, "Miniware DT71 Digital Tweezers," Elektor 7-8/2021: https://www.elektormagazine.com/210182-01

[4] Handheld Advanced Semiconductor Component Analyser: PEAK DCA75: https://www.elektormagazine.com/news/handheld-advanced-semiconductor-component-analyser

[5] Joy-IT LCR-T7 Multi-Function Tester: https://www.elektormagazine.com/news/joyit-lcrt7-multifunction-tester

[6] Review: MiniPro TL866A programmer: https://www.elektormagazine.com/news/review-minipro-tl866a-programmer/14585

# Reducing the Power Consumption of Your Mole Repeller

## An ATtiny13 Replaces a 555

By **Gerhard Dürr** (Germany) and **Luc Lemmens** (Elektor)

Mole hills can ruin your carefully manicured lawn in no time at all. If you guard your precious garden with an electronic mole repeller, this hardware upgrade can increase the life span of its batteries by a factor of 10!

If you have a garden with a lawn, especially if you live in a rural area, you run the risk of being unpleasantly surprised by the presence of moles, or rather by the mole hills they make. The good news is that you will most likely have clean, fertile soil in your garden. You will never — or rarely — see the animals themselves, as they usually stay underground in tunnels that they dig, looking for worms and insects that are on their menu. The soil that they move during their digging has to go somewhere, of course, and they push it upwards and thus form mole hills, the sign that there is a mole in (actually: under) your garden. Not many people are happy about this, and numerous methods have been devised to prevent moles entering a garden or to chase them away once they have settled there. One of the solutions is the so-called mole repeller: a device that uses (sound) vibrations to prevent or end these underground activities. It is a spiked tube that is inserted into the ground; in the tip, there is a sound source (usually in the form of an electronic DC buzzer) that is meant to chase away moles and other vermin like voles and rats.

Some manufacturers claim that the pest repellers they make are producing *ultrasonic* vibrations. Maybe they do exist, but most devices we found – according to their specifications — produce sounds with frequencies in the low kilohertz region or even lower, which is way below the ultrasonic range. In most cases, the term sonic repeller (without ultra-) will be more appropriate. The sound is switched on

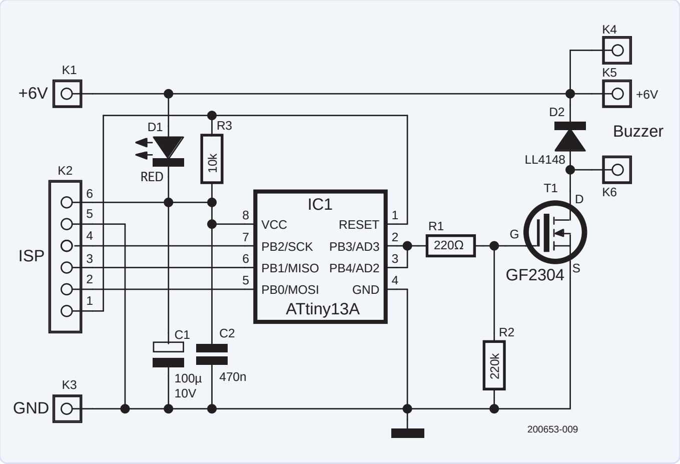


*Figure 1: The circuit diagram of the upgrade of the time circuit.*

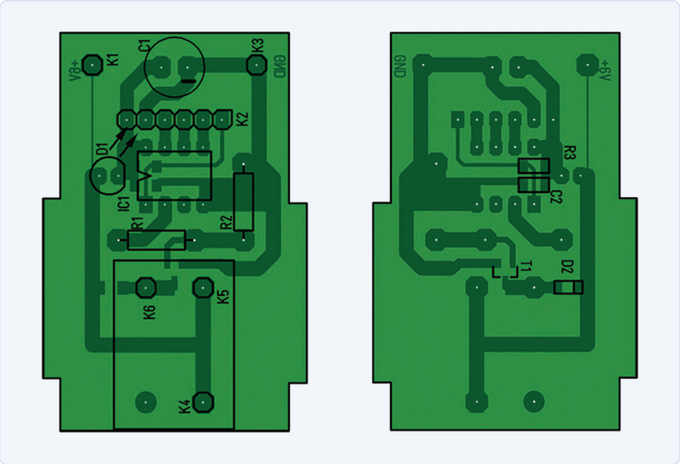


*Figure 2: The PCB layout.*

for about half a second, at intervals of some 30 seconds. These pulses would be so unpleasant for a mole that it would pack up and/or stay away. There are people who are absolutely convinced of the effectiveness of these devices, others claim they are just a scam and that only the manufacturers and sellers benefit from them; we will not comment on that.

Some thoughts on mole repelling in general are shared in the text box. In this article, we will only discuss a way to reduce the power consumption of a battery-powered, sound generating variant.

## How It Works

This type of repeller contains a timing circuit and some kind of sounder; the latter will be a piezo buzzer in most cases, but we have also found devices with vibration motors, like the ones used in cell phones. In electronics, anyone who says "Timer circuits" immediately says "555", perhaps the most famous integrated circuit that recently celebrated its 50th anniversary. There will be exceptions, but the designer of the circuit we are discussing here also came across a mole repellent in which the 555 was used as a bistable multivibrator. Most of the power of the device is consumed when the sound is being produced, but of course some power is also needed to keep the 555 running when the buzzer is switched off. Although tweaking this circuit may save power and thus extends the battery life (e.g., by replacing a standard 555 by a CMOS-version or by changing the values of its external components), the designer chose a different approach. The complete timer circuit is replaced by a small microcontroller that stays in sleep mode most of the time; its watchdog timer wakes it up approximately every half minute and sounds the buzzer for one second and then enters sleep mode again. The current consumption is reduced with nearly a factor of ten: 4 mA with the original 555 timer circuit versus an average of 430 µA with the microcontroller, or a battery life of less than 200 days extended to about 1450 days! In practice this will be shorter, due to factors such as self-discharge of the batteries, but still, this is a great improvement compared to the original current consumption of the mole repeller.

## The New Hardware

The schematic diagram of the microcontroller circuit is shown in **Figure 1**. IC1, an ATtiny13 from Microchip (formerly Atmel), is the heart of the timer circuit. Its internal program memory can be flashed with an ISP programming interface connected to K2. Alternatively, this 6-pin can be omitted if the microcontroller is programmed before it is inserted in the socket for IC1.

The circuit is powered by the batteries, in this case four D-cells in series — i.e., 6 V in total connected to K1 and K3 (the positive and negative terminals, respectively). With four freshly installed 1.5 V batteries, with no load or very low load connected, the total voltage will be higher, more like 6.5 V. However, the ATtiny13 has an absolute maximum supply voltage of 5.5 V. D1 is a standard red LED (preferably *not* a high-efficiency type) which has a forward voltage of about 1.3 V, even at a forward current as low as 1 µA and this effectively lowers the supply voltage to a safe value, well below the 5.5 V maximum for the microcontroller. Alternatively, D1 can be replaced by two or three standard 1N4148 diodes in series. A linear voltage regulator should not be used for this purpose, as it would increase the current consumption of the mole repeller. C1 and C2 are decoupling capacitors for the power supply of the microcontroller.

The buzzer is salvaged from the original hardware and connected to the K4/K5 (positive) and K6 (negative) pins. Some buzzers are inductive and likely to produce spikes, especially when switching off. Diode D2 is the flyback-diode that prevents these spikes from harming the transistor T1. The choice of this switching transistor is not critical at all. With the current through the buzzer being less than 10 mA, also a standard BC547 can be used if base resistor R1 is changed to 4.7 kΩ and R2 to 10 kΩ. Any logic-level NMOS-FET will do too, with R1 replaced by a jumper wire and R2 omitted.

## There Is Even a PCB Design

The schematic was designed in a free version of Target3001!, and the author even made a small PCB for the microcontroller upgrade (see **Figure 2**). The design files are available for download at the Elektor Labs page of this project [1]. Considering the large number of different brands of mole repellers available on the market, it would be a mere coincidence if the PCB exactly fits into the model you have like in

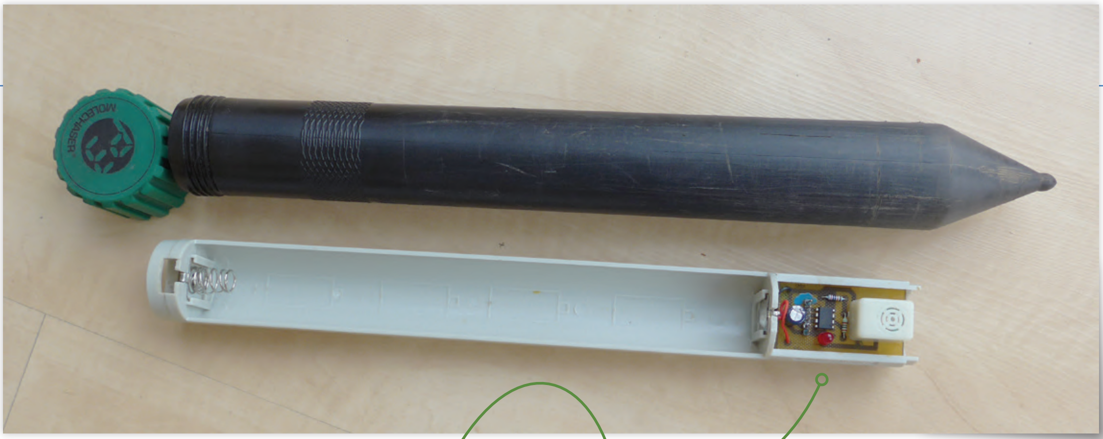


Figure 3a: Overview.

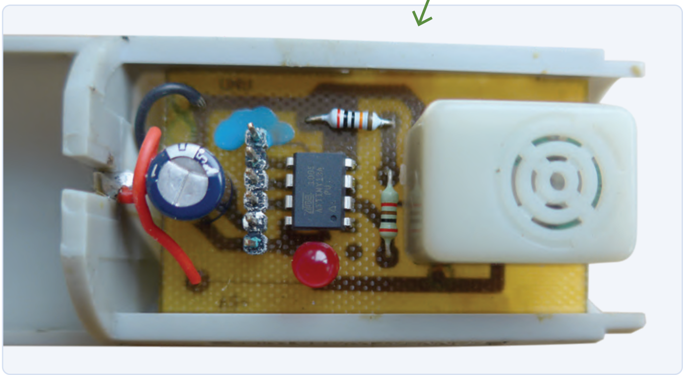


Figure 3b: The new PCB installed inside the original mole chaser.

the pictures the authors made (**Figure 3**). For many of us, it will be a nice opportunity to try and test Target3001! with this relatively simple project if you want to change the PCB, but it will also not be too difficult to redraw the schematic and circuit board if you prefer to use any other CAD program. There are a few SMD components soldered on the copper bottom side of the PCB that can easily be replaced with through-hole parts even without changing the PCB.

## Moles in Your Garden? What (Not) to Do?

If you search on the Internet what you can do about moles in your garden, or how you can prevent them from coming to your property in the first place, you will be surprised what, over the years, has been invented to make life difficult for these creatures. Many of these remedies are doubted by (experience) experts as to their effectiveness. You only know for sure that such a method does not work when — in spite of everything — your beautiful, flat lawn is suddenly marred by mole hills.

We are certainly no mole experts, but according to the information we have found, doing nothing at all seems to be the best advice. If there are molehills, a mole has already made his burrow and it will usually only expand if there is not enough food. If you find the mini-mountains disturbing, the best thing to do is to carefully shovel away the soil; if you knock or pound them, the mole's passage will collapse and there is a good chance that the animal will repair the damage and make a new mole hill in the process.

## Nothing Runs Without Firmware

The source code (**Listing 1**) for the ATtiny13 is developed in AVR Studio 4.19 with AVR-GCC compiler installed. Although this IDE is outdated, it is still available for download, but it won't be difficult to adopt the source to more modern development tools. The source code, the AVR Studio project and the HEX file can be downloaded from the Elektor Labs page associated with this project [1]. The latter allows you to flash the program memory of the ATtiny13 without compiling. The configuration fuses of this microcontroller will stay at factory default (i.e., they don't need to be programmed).

The program largely consists only of assembly statements, written in C. After power on, the GPIOs are configured and the timers, the ADC, the pull-ups, the analog comparator are switched off, and the sleep mode is set to *Powerdown*. The code then enters an endless loop (`while(1)`), which performs the following operations:

- The buzzer is switched on.
- The watchdog is set to generate an interrupt after 128k cycles, with a 128 kHz clock this gives a one second interrupt interval (approximately).
- The interrupt is enabled and the ATtiny switches to *Powerdown* mode with the assembler command sleep.
- After one second, the watchdog interrupt wakes up the microcontroller from *Powerdown.*
- The buzzer is switched off and the watchdog interrupt is changed to its maximum interval of 1024k cycles, i.e. 8 seconds. Since we want the repeller to be silent for approximately half a minute, the watchdog must generate three interrupts before the `while(1)` starts again with switching the buzzer on.

## No More Moles?

With this upgrade of the mole repeller, the battery life of the four D-cells is extended. It will not change the effectivity of the device itself. As mentioned, there are many discussions between people who think these mole repellents are great and others who believe or even know for sure that they have no effect at all. The large number of brands and sellers suggests that a lot of people do believe in them and whether they really work or not. In any case, it is a remedy that has no negative effects on the environment. The most negative thing we found were reports of complaints from neighbours who are bothered by the repellers' repetitive, soft sounds.

200653-01

## COMPONENT LIST

**Resistors**

R1 =220 Ω

R2 = 220 kΩ

R3 = 10 kΩ

**Capacitors**

C1 = 220 µF/25V

C2 = 470 nF

**Semiconductors**

D1 = red LED

D2 = LL4148

T1 = GF2304 (see text)

IC1 = ATtiny13A (DIL-8 style)

**Miscellaneous**

K2 = 6-pin SIL header

**Listing 1: The complete source code.**

```
// Controlling the molechaser-buzzer with the watchdog-timer
// * MC ATtiny13A, all fuse bytes default
// * PB3 and PB4 HIGH: buzzer ON
// * Configurable with watchdog-timer-prescaler
// * Selected: 128K-cycles(~1s) ON, 3*1024K-cycles(~24s) OFF
// * Current consumption: powerdown ~6µA, buzzing ~7.2mA, average 0.43mA
#include <avr/io.h>
#ifndef F_CPU
#warning "Defining F_CPU 1.2MHz"
#define F_CPU 1200000UL
#endif
#include <avr/interrupt.h>
int main(void) {
  uint8_t i=0;
  DDRB  = 1<<PB0|1<<PB1|1<<PB2|1<<PB3|1<<PB4;
  PRR    = 1<<PRTIM0|1<<PRADC;        // Power Reduction Timer0, ADC
  MCUCR = 1<<PUD|1<<SE|1<<SM1|0<<SM0; // Pullup Disable, Sleepmode: Powerdown
  ACSR  = 1<<ACD;                     // Analog Comparator Disable
  while(1) {
    i = 0;
    PORTB |= 1<<PB3|1<<PB4;      // buzzer ON
 // change Watchdog-Interrupt to 1s (measured: 1.14s @ 4.5V, 22°C)
    MCUSR  = 0;                  // clear Reset-Flags
    WDTCR  = 1<<WDCE |0<<WDE;    // Watchdog Change Enable
 // Watchdog Timer Prescaler: 128K-cycles(~1s), Watchdog Timer Interrupt Enable
    WDTCR  = 1<<WDTIE|0<<WDE|1<<WDP2|1<<WDP1;
    sei();
    asm volatile("sleep"::);
    asm volatile("wdr"::);       // Watchdog Reset, wait for WDT-ISR(~1s)
    PORTB &= ~(1<<PB3|1<<PB4);   // buzzer OFF
//  change Watchdog-Interrupt to  3*8s: (measured: 27.9s @ 4.5V, 22°C)
    asm volatile("wdr"::);       // Watchdog Reset
    MCUSR  = 0;                  // clear Reset-Flags
    WDTCR  = 1<<WDCE|0<<WDE;     // Watchdog Change Enable
 // Watchdog Timer Prescaler: 1024K-cycles(~8s), Watchdog Timer Interrupt Enable
    WDTCR  = 1<<WDTIE|0<<WDE|1<<WDP3|1<<WDP0;
    while(i < 3) {               // wait 24s
      i++;
      asm volatile("sleep"::);
      asm volatile("wdr"::);     // Watchdog Reset, wait for WDT-ISR(~8s)
    }
  }
}
EMPTY_INTERRUPT(WDT_vect);       // awaking from powerdown-mode
```

### Contributors

Design: **Gerhard Dürr**

Text: **Gerhard Dürr, Luc Lemmens**

Illustrations: **Gerhard Dürr, Patrick Wielders, Luc Lemmens**

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

Layout: **Giel Dols**

### Questions or Comments?

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

### WEB LINK

[1] This project on Elektor Labs: https://bit.ly/3JWCXBk

# Light Switch **DeLux**

## A Solution for High-Precision Light-Controlled Switching

By Clemens Valens (Elektor)

Light-controlled switches are plentiful and retail for €10 or so, but most of them are changing their state somewhere in the twilight zone. Sometimes applications require better precision and more control than these cheap switches allow for. Do you need a lux-accurate light control? If so, this project is for you.

There exist many, many designs for light-controlled switches and most of them work great in the application they were designed for. These applications usually consist of switching on a light when the ambient light level drops below a certain threshold and switch the light off again when the light level increases. Sometimes a timer is added too.

You might think that every possible application is covered by these designs and yet this is not true. The reason is that they all lack precision. Based on an LDR or phototransistor, they tend to switch somewhere in the twilight zone. However, light levels vary much more than that.

### Brightness Is Subjective

To humans, daylight intensity or brightness is fairly constant. Of course, we notice variations due to clouds and the sun, but we are not very sensitive to them. The reason for this is the eye's logarithmic response to brightness. On a cloudy day brightness can vary between 5,000 and 10,000 lux, yet it looks almost the same to us. Sunlight may result in levels of over 25,000 lux, which we notice, obviously, but we don't experience it as three or more times as bright.

Plants, on the other hand, are way more sensitive to light intensity than humans. Farmers know this, and they shine artificial light on some of their crop even during the day to improve its yield. On sunny days this is usually not needed, but on cloudy days it may help. To do this in an economical way, they therefore need light-controlled switches that can detect brightness differences with more precision than an LDR or phototransistor can do.

Today light sensors exist that convert brightness directly to a value in lux with resolutions of up to 16 bits. Some of these sensors not only measure lux, but also UV and white light intensity. With such a sensor, it is quite easy to build a high-precision light-controlled switch.

## High-Accuracy Ambient Light Sensor

A popular light sensor is the VEML7700 from Vishay. This is a high-accuracy ambient light sensor with I²C interface and it can be found mounted on a small module for a few euros. From the same family we might also cite the VEML6075, a UVA and UVB light sensor also with I²C interface.

Because of its digital I²C interface the sensor does not need an analog-to-digital converter and can instead be connected directly to most microcontrollers. Its output data is available in two 16-bit registers: ambient light (also called 'ALS') and white light. White light covers a wide spectrum from 250 nm up to 950 nm. The ALS spectrum is much narrower, from about 450 nm to 650 nm as it is optimized for human perception. Plants are not human, so the output to use will depend on the application.

Sensor sensitivity is important too. The VEML7700 has a resolution of 0.005 lx per LSB and a maximum detection level of 167,000 lx (minimum is 0.01 lx). Such a wide range would require 25-bit values, but the device is only 16 bits; therefore, a sensitivity value can be specified (sometimes called 'gain') to bring things into range. Its high sensitivity allows the sensor to be used behind low transmittance (i.e., dark) surfaces and still obtain usable results.

For low-light conditions, the sensor features an integration setting of up to 800 ms. Finally, low and high thresholds can be set that can trigger interrupts, making it easy to create alarms or allow for automatic switching.

## Stick It on an ESP32

The VEML7700 module used for my experiments was bought at Adafruit. A good platform to use it with is the ESP32-Connected Thermostat [1] (a.k.a. Automator, see [2]). The module has five pins, but as it has two power supply options, we only need four of them. These four pins have the same order as the OLED display connector on the thermostat, and so we can stick the module on K9. Make sure the VIN pin remains unconnected and the module points upwards (as opposed to how the OLED display would be plugged, see **Figure 1**).

## Software With ESPHome

After connecting the sensor to the ESP32, we must produce some software to make it all work. As the objective is some sort of light-controlled switch, a logical choice would be to use a home automation platform and my favorite is ESPHome. Elektor has published several projects that used ESPHome [3][4], and so you may already know how to use and configure it, but this project introduces a concept that I didn't treat before: creating a custom sensor. If you are new to ESPHome, I recommend reading and watching [3] and [4] first.

## We Need a Custom Sensor

ESPHome supports many sensors, but (at the time of writing) not the VEML7700. It knows of other lux sensors, but not this one. However, this is not a problem as ESPHome offers a method of adding your own sensor. Doing so involves writing some C++ code, so it makes things a bit more involved.

As before (see [3] and [4]) we must declare a *sensor* section in the ESPHome configuration file for this device. The sensor's *platform* now must be *custom*. This tells ESPHome that you will provide all the details for it.

Next follows a *lambda* section which consists of a few lines of C++ code to tell ESPHome to register a sensor of our own devising (that we named *veml7700*). We must also specify the data stream(s) that our sensor produces. In this case it has three outputs: ALS, lux and white light. The order is important and must be respected whenever they are referenced elsewhere in the YAML file.

We continue with normal YAML statements to further specify the sensor outputs. This is done with a *sensors* section (plural!) where we can specify for each data stream its name, its units and how many decimals to use. The order is the same as in the return statement in the lambda section.

Only the lux stream has units ('lx'), and there is no point in having decimals for any of them, so we set them to zero.

The last thing to be done is to indicate where ESPHome can find the driver for the custom sensor. We do this in the *esphome* section at the top of the configuration file where we add a file (veml7700.h) to the *includes* subsection. On the system that runs ESPHome, this file should be in the same folder as the device's YAML file.

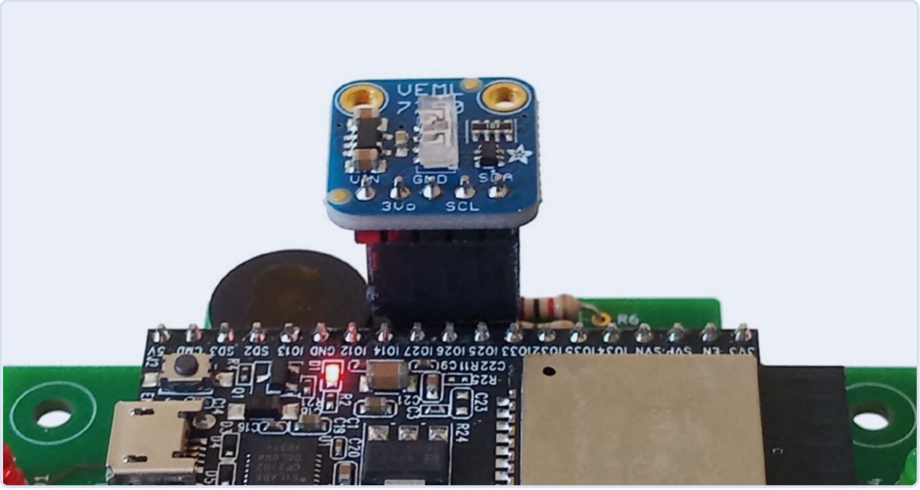


Figure 1: The 5-pin VEML7700 module plugs onto 4-way connector K9. Its VIN pin is not connected even though it may look like it.

## The Hard Part

Done? Not really. Now comes the hard part, which is, of course, creating the driver for the custom sensor. This driver must comply to ESPHome standards for components (a sensor is a component), meaning that it must provide certain functions that ESPHome expects from its components, and it must communicate with the sensor itself.

For the second part, we can rely on Adafruit who, besides manufacturing the VEML7700 module, also created an Arduino library for it [5]. Our driver must provide the interface between ESPHome and the Arduino library.

## Installing a Third-Party Library

ESPHome provides a mechanism for adding open-source community libraries: simply add the exact (!) name of the library to the *libraries* subsection in the *esphome* section. In our case this is “Adafruit VEML7700 Library”. Now, when ESPHome processes the configuration file, it will first install the library (if it didn’t do so already, see **Figure 2**) before compiling everything. In your custom driver, you can simply include the library as any other library.

I haven’t pushed this feature too far, so I don’t know the criteria a third-party library must fulfill to be used this way. You may want to consult the platform.io documentation for more details as it is the toolchain used by ESPHome.

## An Encapsulated Arduino Sketch

Our driver is a C++ class that must provide at least a constructor and a function named *update*. We need a function *setup* too so that we can initialize the Adafruit driver. The functions *setup* and *update* of our class do what would normally be done in the functions *setup* and *loop* of a typical Arduino sketch using the Adafruit driver. Basically, our class encapsulates an Arduino sketch including global variables and adds some ESPHome-specific things to it. For ESPHome we must insert calls to *publish_state* for each data stream (ALS, lux & white light) to the function *update*. This will make the data available to the rest of the world. The call order must be the same as in the return statement in the *lambda* subsection of the *sensor* section (see above).

As our class inherits from the *PollingComponent* class which itself inherits from the *Component* class, other functions are available that you might want to use. A polling component is a component that is called periodically, and the call rate can be specified (e.g., in our constructor). Refer to [6] for more details.

## Adding the I$^2$C Bus

The one thing that remains to be done now is connecting the sensor to the I$^2$C bus inside the software (i.e., create a logical connection between the two, as we already have a physical connection). The Adafruit library uses Arduino’s Wire library for this purpose. In the ESPHome YAML file, we therefore add an *i2c* section so ESPHome knows that it is needed.

The ESP32 has two I$^2$C buses with SDA and SCL signals that may be connected to almost any pin on the chip. ESPHome therefore supports multiple I$^2$C buses with freely assignable pins, making it a matter of specifying what goes where. But the Arduino Wire library supports only one I$^2$C bus, so how do you tell it to use the bus you want? Well, you can’t, as it will always use the default bus. In ESPHome the default I$^2$C bus is the first bus specified in the *i2c* section. It would have been nice to be able to specify an I$^2$C bus by using its ID, and ESPHome would probably be more than happy to let you do it, but the underlying Arduino Wire library doesn’t allow this.

## Done

Here ends this article. If the Automator is programmed by ESPHome with the code described above, you will get a device that can be controlled from a home automation controller like Home Assistant (or by itself). There are no automation rules inside the ESPHome YAML file, so without a controller it will just measure ambient light intensities. Automation rules can be added to the YAML file itself, or they can be created inside e.g. Home Assistant. For a description of the rest of the YAML code, which is nothing special, please refer to [3] and [4].

The YAML configuration file and C++ code is available for download from this article’s the webpage [7].

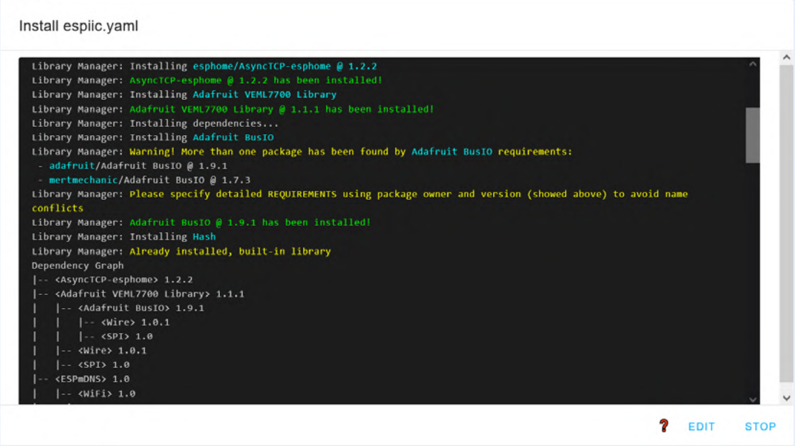
210190-01

### Contributors

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 his article? Email the author at clemens.valens@elektor.com or contact Elektor at editor@elektor.com.

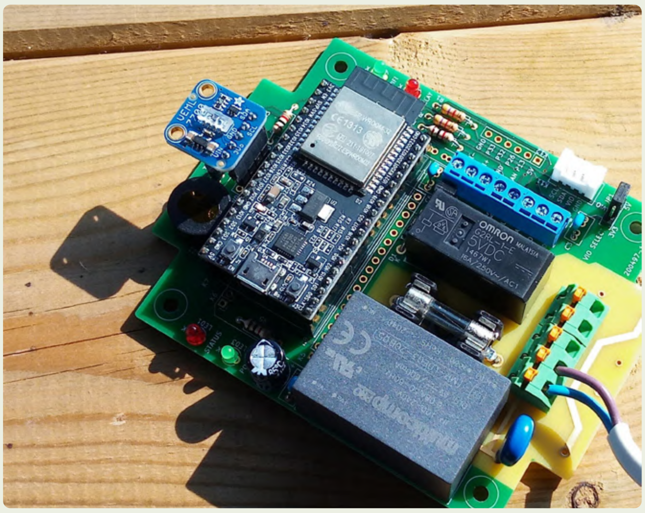


Install espiic.yaml

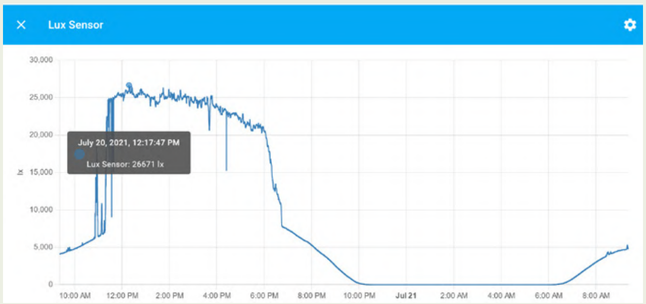
```
Library Manager: Installing esphome/AsyncTCP-esphome @ 1.2.2
Library Manager: AsyncTCP-esphome @ 1.2.2 has been installed!
Library Manager: Installing Adafruit VEML7700 Library
Library Manager: Adafruit VEML7700 Library @ 1.1.1 has been installed!
Library Manager: Installing dependencies...
Library Manager: Installing Adafruit BusIO
Library Manager: Warning! More than one package has been found by Adafruit BusIO requirements:
 - adafruit/Adafruit BusIO @ 1.9.1
 - mertmechanic/Adafruit BusIO @ 1.7.3
Library Manager: Please specify detailed REQUIREMENTS using package owner and version (showed above) to avoid name
conflicts
Library Manager: Adafruit BusIO @ 1.9.1 has been installed!
Library Manager: Installing Hash
Library Manager: Already installed, built-in library
Dependency Graph
|-- <AsyncTCP-esphome> 1.2.2
|-- <Adafruit VEML7700 Library> 1.1.1
|   |-- <Adafruit BusIO> 1.9.1
|   |   |-- <Wire> 1.0.1
|   |   |-- <SPI> 1.0
|   |-- <Wire> 1.0.1
|   |-- <SPI> 1.0
|-- <ESPmDNS> 1.0
|   |-- <WiFi> 1.0
```

*Figure 2: ESPHome installs the Adafruit library automatically.*

## Some Results



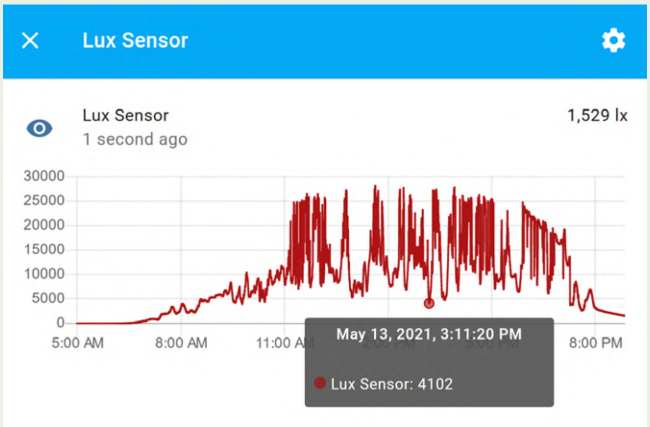
The light sensor placed in bright summer sunlight. At noon, with sensitivity set to 0.125 and an integration time of 100 ms, the sensor reported values of around 48,000 for ALS and a bit more than 30,000 for white light. These values correspond to light intensities of around 22,000 lx. At 10:30 AM with the same conditions, I measured 16,800 lx, a difference of 5,200 lx. However, to my eyes the brightness looked pretty much the same in both situations.



July 20, 2021, 12:17:47 PM

Lux Sensor: 26671 lx

Evolution of brightness on a sunny day in July. The "bouncing" on the left is due to the shadow of a tree. From noon to 6 PM the brightness slowly decreases but you wouldn't really notice it. The dips are caused by clouds. The shadow of the house starts to cover the sensor at 6 PM, which explains the steep fall. Then light intensity continous to decrease until dark. For humans 5,000 lx is already bright.



Lux Sensor

1 second ago

1,529 lx

May 13, 2021, 3:11:20 PM

Lux Sensor: 4102

On a partly cloudy day in May brightness levels may be anywhere between approximately 4,000 and 27,000 lx.

All graphs were created with the help of Home Assistant at a latitude of 47°N.

### RELATED PRODUCTS

- ESP32 DevKitC (SKU 18701)
  www.elektor.com/18701
- 0.96" 128×64 $I^2C$ OLED display (SKU 18747)
  www.elektor.com/18747
- Koen Vervloesem, *Getting Started with ESPHome* (SKU 19738)
  www.elektor.com/19738

### WEB LINKS

[1] Y. Bourdon, "ESP32-Connected Thermostat," ElektorMag 9-10/2021: www.elektormagazine.com/200497-01
[2] Elektor Automator: www.elektormagazine.com/labs/automator
[3] ESPHome starts here: www.elektormagazine.com/labs/how-to-home-assistant-esphome
[4] C. Valens, "Home Automation Made Easy," ElektorMag 9-10/2020: www.elektormagazine.com/200019-01
[5] Adafruit VEML7700 library: https://github.com/adafruit/Adafruit_VEML7700
[6] ESPHome on GitHub: https://github.com/esphome
[7] Downloads for this article: www.elektormagazine.com/210190-01

# The Challenges in Bringing **IoT Solutions** to Market

## Worries Around Security, Scalability, and Competition

*Figure 1: The European Commission has examined whether the big players in voice assistants, the favorite user interface to the smart home, are stifling competition. (Source: ShutterStock/Gorodenkoff)*

**By Stuart Cording (Elektor)**

The Internet of Things (IoT) has been around for more than 20 years, and a host of wireless technologies have sprung up to support its deployment. In the home, voice assistants have established themselves as the primary user interface to an array of smart devices. Despite these advancements, more than a third of IoT projects never make it past the proof-of-concept phase. Furthermore, the European Commission is concerned that a lack of competition in some application spaces may be hindering market entry for EU businesses. So, what are the real challenges, and what is it like to deploy an IoT solution across Europe?

If you want to get a feel for the technologies behind the Internet of Things (IoT), finding a raft of suitable projects doesn't take long. Just search for "IoT" and you're preferred prototyping platform, and you'll be overrun with pages of projects, cloud platforms, technology comparisons, and lists of ideas. *Elektor* is also a great source of information. Since the term IoT was coined more than 20 years ago, our website has collected more than 600 articles on or related to the topic [1].

However, there is a considerable gap between a prototyped platform that demonstrates the core concept of an IoT solution and deploying one for real. The IoT Insights report from Microsoft [2] interviewed more than 3,000 IoT professionals in 2021. They found that 35% of IoT projects experience failure during the trial or proof-of-concept phase, up 5% on their survey from a year earlier. Most cited as the reason for failure at this stage is the high cost of scaling. Other reasons include too many platforms to test, too many use cases to prove, and a lack of resources. A separate study by Cisco [3] reported that only 26% of companies surveyed thought their IoT initiatives had been successful. Responses to their study showed that, while most IoT projects look good on paper, they proved to be more complex than initially thought.

Despite such gloomy feedback on IoT overall, there are sectors where IoT is making huge inroads and delivering growing revenues.

## EU Commission Reviews Consumer IoT Sector

EU citizens have welcomed the range of consumer IoT solutions on offer in recent years. So much so, a Smart Home revenue report from Statista [4] predicts that related revenues will double from around €17b in 2020 to around €38.1b in 2025. Concerned that competition in this sector may be being stifled, the European Commission undertook an inquiry as part of their digital strategy [5]. The report, released in January of 2022, garnered input from manufacturers of wearable devices, connected consumer devices used in the smart home, and those providing services over such smart devices. Additionally, the Commission requested input from standard-setting organizations. However, upon reading, many paragraphs of their analysis are given to the voice assistants (VA) that form the user interface to many IoT products and services (**Figure 1**).

> *Looking at the available IoT landscape, it is clear that business opportunities abound, regardless of whether you are focused on solutions for consumers or industry.*

Having analyzed the landscape of the smart home, the report [6] makes it clear that, in Europe, Google's Google Assistant, Amazon's Alexa, and Apple's Siri are the leading general-purpose VAs. Other VAs are available, but these tend to be more limited in functionality and focus on supporting a single product or a service provider's app. According to ZDNet, of the solutions from the big three players, Amazon provides the highest level of compatibility, supporting around 7,400 brands [7]. By comparison, Google supports around 1,000, while Apple remains the most exclusive, supporting about 50.

## Little Room for Voice Assistant Newcomers

With such powerful industry players already established on the market, there is little room for newcomers, and the technology curve to develop a competitive VA is significant. Thus, if you want to leverage voice control for your IoT solution, you have to play according to the rules set by the big three. An alternative approach would be to license a VA. However, some manufacturers questioned reported that licensing conditions were restricting their choices. These ranged from exclusivity or restrictions to stop more than one VA being used concurrently to licensing that forced the inclusion of other types of software or applications, meaning the VA technology couldn't be used standalone.

Another big concern is access to data. As a third party relying on a VA, you only have limited access to the data collected. The VA provider has access to the audio recordings and would also know how many failed attempts there have been to issue the commands selected for your device. However, your team will not be provided access to the audio recordings, so you'll more likely have to wait for user feedback to discover that your choice of voice commands is suboptimal amongst a group more expansive than that used for testing. Additionally, because the VA provider can analyze everything spoken into it, they could conceivably use this data to develop a solution that competes with yours or leverage user experience provided by your IoT solution to improve their services.

A further issue arises when the VA provider also offers advertising services. Theoretically, the voice input provided by your users could help the provider target advertising more accurately to the demographic represented by your customer base.

Finally, there is the loss of brand recognition and experience. Your carefully crafted solution is at the whims and mercies of the VA. Should they choose to make a significant change, such as the voice used, the wake word, or even roll-out functionality changes that result in a drop in users, you're inevitably going to suffer the consequences too.

The report also examines many other relevant areas, including application programming interfaces (API), standards, interoperability, the imbalance in power between many third-party IoT device developers and the big cloud platform service providers, and contract termination clauses.

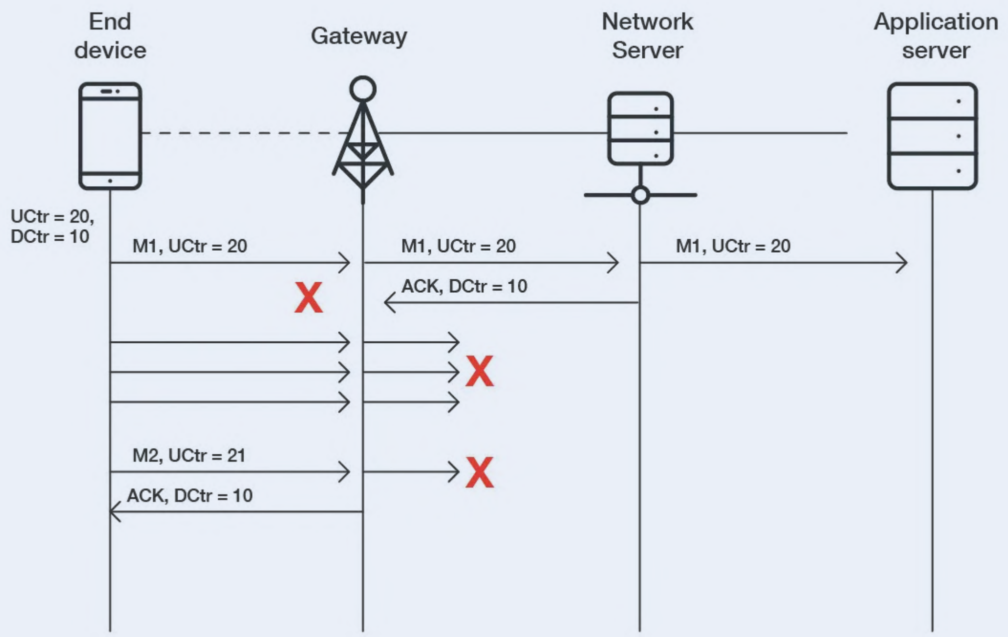


*Figure 2: Researchers discovered a Denial-of-Service (DoS) attack in LoRa 1.0. By replaying a previous successful data transfer, a LoRaWAN node is hindered from sending further data packets. (Source: Trend Micro)*

There are no recommendations in the report. However, the conclusions state that the report's content will contribute to the Commission's standardization strategy and feed into the legislative debate on the Digital Markets Act (DMA).

## Security Concerns Worry IoT Implementers

Reviewing the data collected in Microsoft's IoT Insights report, it is IoT security that is high on the list of worries. This issue is of particular concern for those planning IoT solutions, with 29% indicating that the associated security risks are holding them back from using IoT more. The report also explains that around a third of organizations are concerned about the security risks of IoT, especially data breaches. To combat this, outsourcing is highlighted as the best way of improving peace of mind.

While many engineers know the phrase "obscurity is not security," few are skilled enough in this domain to ensure a solution is protected, end-to-end, from attackers. And while semiconductor suppliers offer a range of single-chip security solutions, developers still need to understand how to use them correctly to ensure that they don't inadvertently introduce new security weaknesses.

Over the past decades, low-power wide-area networks (LPWAN) solutions such as LoRa and Sigfox have established themselves as key wireless IoT technologies supporting long-range communication. Achieving ranges of tens of kilometers, they are an alternative to cellular wireless such as LTE Cat-M1 and NB-IoT thanks to their superior low-power performance for low data volumes [8]. But how secure are they?

## LoRaWAN Under the Magnifying Glass

LoRa and LoRaWAN have been subject to particular scrutiny by the security and hacking community. Sébastien Dudek of Trend Micro, a company focused on IT security solutions, is one of several researchers that has written extensively on some of the potential issues. In a series of three technical briefs [9][10][11], he outlines a range of issues in the implementation and potential attacks. These range from denial of service (DoS) (**Figure 2**) and eavesdropping to bit-flipping (**Figure 3**) and spoofing of acknowledgments (**Figure 4**). The outcomes of such attacks range from the inability to communicate with nodes and reducing the battery life to altering application data.

Many of the vulnerabilities highlighted were resolved between version 1.0.2 and 1.1 of the LoRaWAN standard. However, further challenges arise when operating LoRaWAN nodes with gateways using different versions of the specification. In such cases, there is a need to make modifications to ensure secured backward compatibility between end devices and the back-end, as highlighted in a paper from 2018 by Tahsin C. M. Dönmez [12].

Beyond hacking the wireless link, there is also the issue of bad actors stealing and directly attacking the hardware. Sébastien Dudek also examines this aspect of security. In the case of LoRaWAN, many solutions use a microcontroller (MCU) and a Semtech wireless module. As these are connected via SPI, data passing between the two can be easily captured and analyzed.

Beyond this, there is also the issue of the security of the MCU itself. One attack method simply extracts the firmware from flash

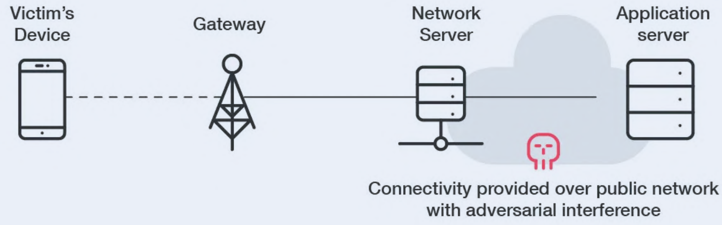

Victim's Device
Gateway
Network Server
Application server
Connectivity provided over public network with adversarial interference

*Figure 3: Due to the known fixed structure of LoRaWAN data packets, it is possible to flip bits (bit-flipping attack) in the content without having to decrypt the message. This requires access to the network server receiving the data. (Source: Trend Micro)*

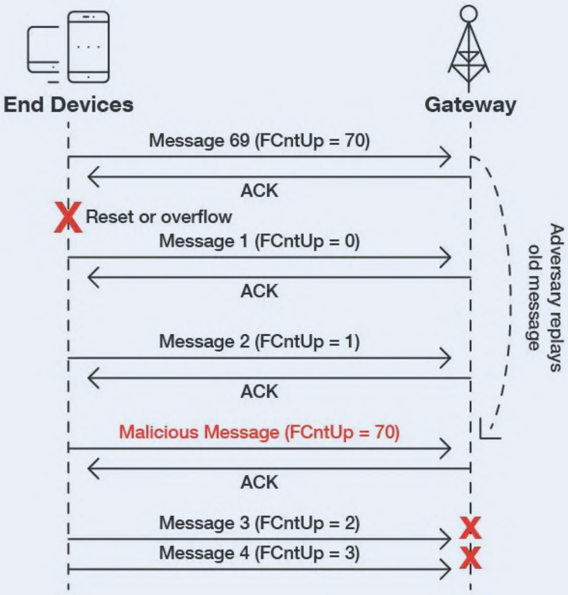


*Figure 4: Jamming the wireless link results in the LoRaWAN node repeating the packet transfer up to seven times, reducing battery life. If the attacker also captures and replays acknowledgment packets (ACK), the node thinks the link is still functional as ACK packets don't declare the message they are acknowledging. (Source: Trend Micro)*

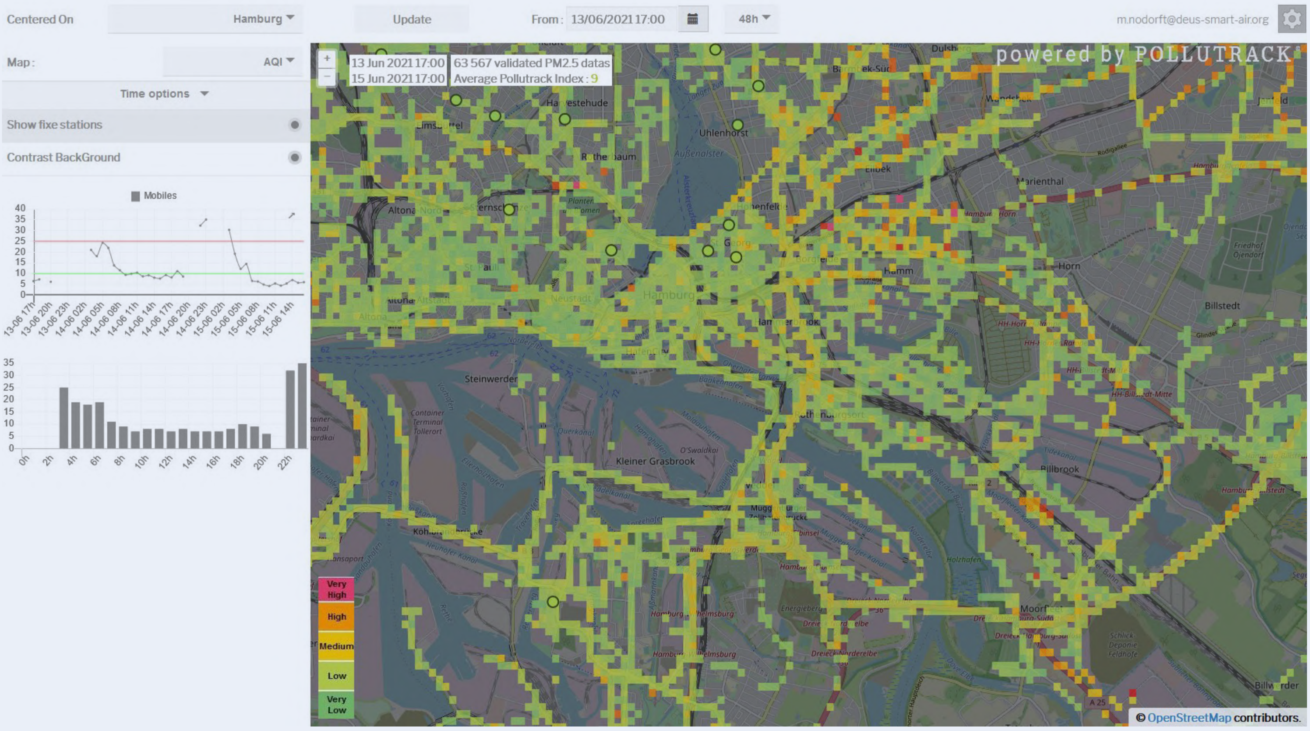
memory, allowing the code to be analyzed. If the security keys are also in the firmware, an attacker can use them to develop nodes that spoof authentic end devices. The recommendation to counter this is the use of Secure Elements (SE), single-chip authentication devices that securely store encryption keys. An approach using Microchip's ATECC608A [13] is one of several for which example code is available. However, while example projects demonstrate how to protect the cryptographic keys, the secure boot feature of this authentication device is not used. Thus, if the same approach were used for a product, the authentication device could be removed and used as an SE with a different MCU and new firmware.

## Security Issues Across the Board

LoRaWAN end-devices provide only limited data bandwidth and, as they have no IP address like a Wi-Fi module, they are not addressable. As such, they offer minimal risk to corporate networks. However, potential risks lie with applications based upon such wireless technologies. These can have implications on lives and the environment should something goes wrong. For example, as part of a smart city network, LoRa-based sensors may be responsible for monitoring water levels to avoid flooding. If the data from the sensors are blocked, flood defense systems may not respond. Conversely, false injected data could result in flood defenses responding to an event that isn't happening with potentially equally disastrous consequences.

And while LoRa and LoRaWAN have been highlighted here, plenty of researchers are examining other LPWAN technologies, including Sigfox and NB-IoT. In a paper by Florian Laurentiu Coman et

*Figure 5: DEUS POLLUTRACK particulate matter sensors gather data from stationary and mobile IoT units. Data is delivered to the back-end using 4G and 5G cellular networks. (Source: DEUS POLLUTRACK)*



Centered On: Hamburg

Map: AQI

Update

From: 13/06/2021 17:00

48h

m.nodorft@deus-smart-air.org

Time options

Show fixe stations ●

Contrast BackGround ●

13 Jun 2021 17:00 | 63 567 validated PM2.5 datas
15 Jun 2021 17:00 | Average Pollutrack Index : 9

*Figure 6: A cloud-based dashboard shows the level of airborne pollutants, as shown here for the German city of Hamburg. Local authorities use such data to make decisions on transportation solutions. (Source: DEUS POLLUTRACK)*

al. [14], several proof-of-concept attacks on wireless networks in addition to LoRaWAN are described. In a technical brief issued by Deutsch Telekom [15] it was stated that implementing Sigfox and LoRaWAN "without [an SE] can [even make] end-to-end encryption useless." It explains that, by contrast, NB-IoT benefits from long-proven LTE security features, such as authentication and secure key generation and exchange. However, it also makes clear that end-to-end encryption is not standard and, if deemed necessary, must be discussed with the network operator.

## Delivering City-Wide IoT Solutions

Concerns regarding security in LPWAN networks influenced technology choices made by DEUS POLLUTRACK Smart City GmbH i.G. for their IoT platform [16]. Their team has been developing IoT sensor networks to monitor particulate matter in cities for more than a decade. With the technology deployed in more than 15 European cities, it enables local leaders of municipalities to make informed environmental decisions regarding air pollution. Their patented optical particle counters (OPC) are capable of monitoring down to the ultrafine particle (UFP) classification (under 0.1 µm). While larger particulates, such as $PM_{10}$ are considered dangerous for the lungs, UFPs can enter the bloodstream and pass to other organs through inhaled air.

DEUS's sensor technology (**Figure 5**) uses a combination of stationary and mobile sensors, networked to back-end dashboards that visualize the data collected. Cities such as Marseille and Paris use 40 stationary sensors complemented by 300 mobile sensors [17]. Mobile sensors are fitted to vehicles of partners, such as the delivery vans of DPD, that are already frequently underway in the target city. These sensors recalibrate themselves against data acquired by the stationary sensors they pass to ensure the accuracy required based. All this requires a robust, reliable, and secure LPWAN choice.

Talking to co-founder Marc Nodorft, he explains that both Sigfox and LoRaWAN were considered during the early development stages. Sigfox offered connectivity infrastructure, simplifying system deployment, but neither provided the data throughput required. LoRaWAN, at the time, wasn't secure enough out-of-the-box and, without infrastructure partners in the cities where the technology was to be deployed, it would be necessary to deploy gateways that connected to the back-end via cellular networks. As a result, 4G and, later, 5G cellular were selected, resolving the issues of coverage, reliability, and security to the levels required.

Nodorft also tells us that, while there are plenty of cheap electronic solutions for IoT available, these are not robust enough for long-term deployment in the environments where their products are installed. Hence the choice was made to develop according to industrial standards, another consideration for those planning their own IoT products.

Another aspect is the back-end operations, which they have developed specifically to the needs of their IoT implementation (**Figure 6**). Moving forward, there is a need to support open-source reporting dashboards to allow government bodies using the system and citizens to access the data, which requires a cloud services provider. And, while there are plenty of choices, the provider is considered as important as the technical solution. Thus, the search is on for a provider that can provide personal support and not just an impersonal customer service chatbot.

With so much experience in significant IoT deployments and learning much about the technical challenges, I wondered what other advice Nodorft could offer those seeking to implement IoT solutions. "We've always stayed true to our vision," he replies, "which has often required us to change the approach." This has meant evaluating different technologies, working with different partners, and modifying the sales strategy on their road to success.

## Teams of Experts and Partnerships Required

Looking at the available IoT landscape, it is clear that business opportunities abound, regardless of whether you are focused on solutions for consumers or industry. However, the journey from concept to deployment is fraught with challenges. While embedded systems developers may be well versed in hardware and firmware development, and may even have experience in wireless technologies, IoT and its security and scalability challenges may be too much for an organization to tackle alone.

According to the European Commission's report, large organizations also have the upper hand in business relationships when it comes to services and platforms. Small players and startups will struggle to get the support they need in these asymmetrical relationships if they go it alone. Without a doubt, expertise, either hired or loaned, is essential to move beyond example applications, demonstration dashboards, and test IoT services. Finally, it is vital to fix your vision while remaining agile in all areas of implementation, from technology choices to target market, to fulfill it.

220053-01

**Contributors**
Text: **Stuart Cording**
Editors: **Jens Nickel, C. J. Abate**
Layout: **Harmen Heida**

### Questions or Comments?

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

## WEB LINKS

[1] Elektor IoT Articles: www.elektormagazine.com/select/internet-of-things-IoT
[2] "IoT Insights, Edition 3," Microsoft/Hypothesis, October 2021: https://bit.ly/3rxMk3a
[3] "The Journey to IoT Value," Cisco, May 2017: https://bit.ly/3GzdJWS
[4] Dr. J. Lasquety-Reyes, "Smart Home - revenue forecast in Europe from 2017 to 2025," Statista, June 2021: https://bit.ly/3LlGiuG
[5] "Sector inquiry into the Consumer Internet of Things," European Commission, January 2022: https://bit.ly/3Lgw9iE
[6] "Final report - sector inquiry into consumer Internet of Things," European Commission, January 2022: https://bit.ly/3B2Htu9
[7] "The best voice assistant," ZDNet, September 2021: https://zd.net/3rxf6Rt
[8] L. Tan, "Comparison of LoRa and NBIoT in Terms of Power Consumption," KTH Royal Institute of Technology, January 2020: https://bit.ly/3JafsUb
[9] S. Dudek, "Low Powered and High Risk: Possible Attacks on LoRaWAN Devices," Trend Micro, January 2021: https://bit.ly/3rA02Tg
[10] S. Dudek, "Gauging LoRaWAN Communication Security with LoraPWN," Trend Micro, February 2021: https://bit.ly/3LhV0T5
[11] S. Dudek, "Protecting LoRaWAN Hardware from Attacks in the Wild," Trend Micro, March 2021: https://bit.ly/3rxquge
[12] T.C.M. Dönmez, "Security of LoRaWAN v1.1 in Backward Compatibility Scenarios," Elsevier, 2018: https://bit.ly/3GtzKq0
[13] Microchip Product Page - ATECC608A: https://bit.ly/3B7zlms
[14] F.L. Coman et al., "Security issues in internet of things: Vulnerability analysis of LoRaWAN, sigfox and NB-IoT," IEEE, June 2019: https://bit.ly/3uwhUQX
[15] "NB-IoT, LoRaWAN, Sigfox: An up-to-date comparison," Deutsche Telekom AG, April 2021: https://bit.ly/3uyUydj
[16] DEUS Pollutrack Website: https://bit.ly/3sHL9O5
[17] DEUS Sensor Measurement Network: https://bit.ly/3Gzjbcc

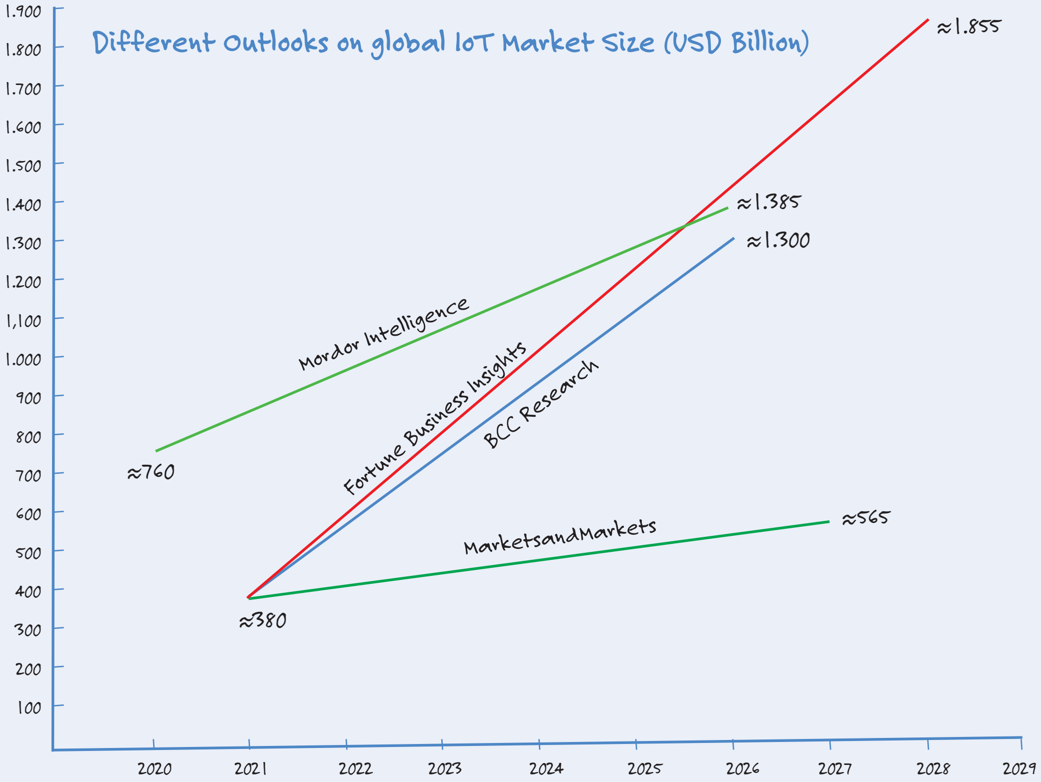
Elektor infographic

By Robert van der Zwan

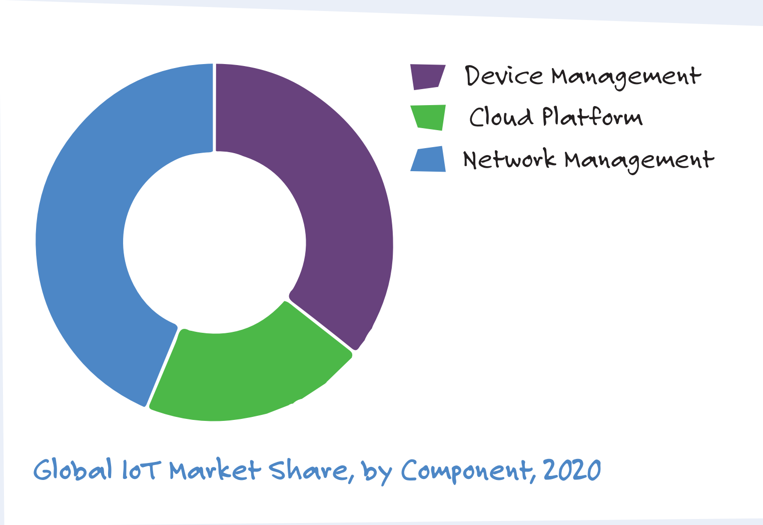
# IoT: Is It the New Holy Grail?

What is IoT? We all know what IoT stands for. The question should rather be: what should count as the Internet of Things? Does the IoT market include Data Warehouse as a Service (DwaaS)? Does IoT include cybersecurity? Or do such sectors stand on their own? Apart from the debate on inclusion and exclusion, it remains surprising how different the outlooks on the global IoT market size can be. The outlooks (yes, plural) of market research companies are so different that the glass is not only half full or half empty, but also completely full to the point of overflowing. There must be something close to a Holy Grail here.

*(Sources: BCC Research, Fortune Business Insights, MarketsandMarkets, Mordor Intelligence)*



## Three Ingredients? **No, There Are Four**



Global IoT Market Share, by Component, 2020

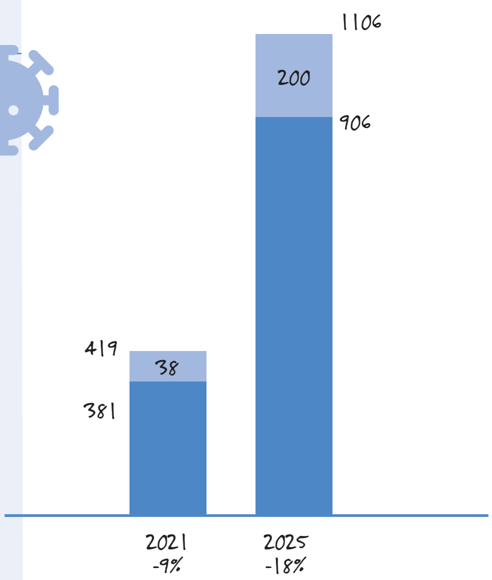
When one thinks of IoT, one thinks of devices and their connections to a central computer. But such a simple view is no longer in line with reality. Apart from a device, a connection and a computer, there is another ingredient: a cloud platform, enabled by data centers. Devices send their information to the cloud, where it is stored and analyzed. A computer of the customer taps into the cloud to get the final results. This Data Warehouse as a Service (DWaaS) or Platform as a Service (PaaS) represents about 20% of global IoT revenue. Cloud platform services are expected to grow round 22% between 2021 and 2026.

*(Source: BCC Publishing, Fortune Business Insights)*

# Negative Effect
## Covid-19 Grows Over the Years

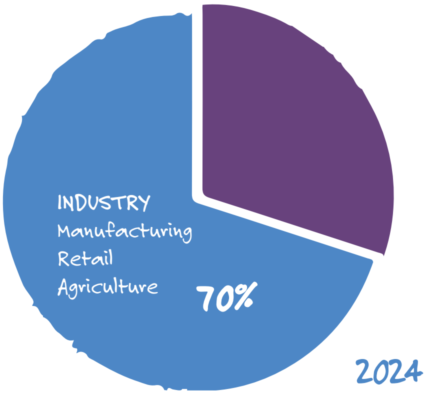
Vodafone does not want to stand in the way of IoT companies painting a rosy picture of their market expectations. However, the telecom company did some research on the effect of Covid-19 on the global IoT market. The multinational concluded that the effects of Covid-19 will be felt for quite some time. Whereas last year the global IoT market lost 9% of its turnover because of the pandemic, this percentage will double to 18% by 2025. Although Vodafone is mindful of the shadow that Covid-19 casts over the coming years, its report on IoT makes equally clear that the sun prevails during that very same period.

*(Sources: Saft Batteries, Vodafone)*



Effect of Covid-19 on Global IoT Market Size

Contribution Industrial Sector to Total of IoT connections



# Where Do All Those
## IoT Connections Go?

Which sectors in society will benefit the most from IoT? When one would have asked that question a couple of years ago, the answer would have been: the consumers. Smart home appliances still are very important considering that anno 2022 the average household in North America includes no less than nine IoT devices, ranging from a frontdoor camera to temperature control. However, industrial applications (including retail and agriculture) will account for 70% of all IoT connections in 2024. The remaining 30% can be divided between consumer electronics and public services like smart parking, smart waste, smart street lighting and smart traffic management.

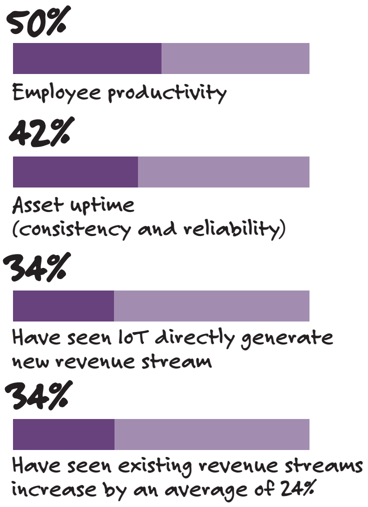
*(Sources: Saft Batteries, Global Information, Inc.)*

# Why Does IoT Make
## Everybody's Heart Tick?

One thing is very clear: IoT will increasingly become a part of our daily lives. But why? Is it because IoT devices are becoming cheaper by the day? This may be true but of course begs the question why vendors can rely on a mass market and lower their prices so much. The answer is basically twofold: (1) the IoT saves money in substantial amounts for its users and (2) it will create new revenue streams, also to a very large degree. When it comes to savings, one can think of an increase in employee productivity and asset uptime. Creating new revenue streams or enlarging existing ones is another way of earning money. How about checking if your houseboat is OK while on holiday?

*(Sources: Saft Batteries, Vodafone)*

Key Benefits of IoT Deployments



# Preferably **Wired** After All

## Tips for Developing a 1 Gbit/s Interface in an Industrial Environment

By Dr. Heinz Zenkner (Freelance Consultant at Würth Elektronik)

Wireless networks are becoming increasingly popular, especially in industrial applications. However, there is a strong case for robust cabling via Ethernet in many cases as the more reliable and secure option. This article demonstrates how to easily implement a 1 Gbit/s interface.

Industrial wireless sensor networks can be established using smart sensors and meters that use efficient modulation and coding techniques with good propagation characteristics and low bandwidths. However, the majority of the use cases explored are limited to low throughput applications. For these use cases the actual throughput is often no more than 1 Mbit/s.

There are no definitive boundaries in a wireless network. For example, even minor adjustments to the access point's antenna positioning can have a significant effect on the signal strength at the other stations. The signal is attenuated by walls, ceilings, and floors, and reflected by metallic objects. While a station may be able to receive the signal from an access point, the access point in turn may not be able to receive the station's signal. In addition, there is a possibility that the network could be accessed from the outside or that the wireless signal transmission could be interfered with.

As a result, wireless data transmission is intrinsically less reliable than transfers through a wired network. Thus, particularly in industrial settings, there may be instances where a wired Ethernet network is the only viable solution.

### Wired Ethernet Network

Similar to wireless networks, wired networks work by exchanging Ethernet frames between endpoints. There are a few rules to follow when setting up a network to avoid problems. The most common cause of network problems is rule breaches. For example, in Ethernet, wire length must not be arbitrary. When cascading, i.e., connecting hubs in series, an arbitrary number of hubs is not permitted, and an unfavorable network configuration can also lead to errors or add unnecessary loads on the network. However, depending on the cable quality and the performance of the hardware, the expected data rates are often not achievable.

Currently, 100Base-TX (100 Mbit/s Fast Ethernet), Gigabit Ethernet (1 Gbit/s), 10 Gigabit Ethernet (10 Gbit/s) and 100 Gigabit Ethernet (100 Gbit/s) are available. For most applications, Gigabit Ethernet works well with regular Ethernet cables, specifically CAT5e and CAT6 cabling standards. These cable types adhere to the 1000BASE-T wiring standard, alternatively referred to as IEEE 802.3ab.

The 1 GB Ethernet interface conforms to the 802.3ab-1999 (CL40) standard and requires four wire pairs/channels for signal transmission. This results in a symbol rate of 125 Megabaud (MBd) and a bandwidth of 62.5 MHz per channel (2 bits per symbol). The signal voltage at 1000BASE-T (GB Ethernet) is typically 750 mV differential, for the limits 820 mV $> V_{\text{Signal}} >$ 670 mV at a load of 100 Ω.

## 1 Gbit/s Ethernet Front-End

A typical front-end for Ethernet is equipped with an RJ45 port. These are intended for full-duplex transmissions, i.e. simultaneous transmission of send and receive data. This is possible because the connector contains two pairs of wires, one in each direction (differential voltage principle). The IEEE standard specifies galvanic separation via a transformer for each RJ45 connection. This transformer protects the devices from damage caused by the line's high voltage and prevents voltage offsets caused by potential variations between the devices. The circuit diagram for a Gigabit Ethernet interface is shown in **Figure 1.**

## Discrete Circuitry of the Gigabit Ethernet Interface

The Ethernet transformer (LAN transformer) is the device-to-Ethernet cable interface. The transformer provides the necessary galvanic isolation between the device and the cable while at the same time matching the impedance of internal logic and the balanced wire pairs. Furthermore, the transformer protects the device from transient interference and suppresses common mode signals between the transceiver IC and the cable, both within the device and between the external cable and the device's electronics. However, the device must also be capable of transmitting data at a rate of up to 1 Gbit/s without significantly degrading the transmit and receive signals. Additional components are needed to meet the matching and electromagnetic compatibility (EMC) criteria.

**Figure 2** depicts a circuit diagram of the Gigabit Ethernet interface using discrete components. The LAN transformer provides DC isolation between the electronics and the network cable. The primary-side winding's middle tap depicts the so-called "Bob Smith" termination.

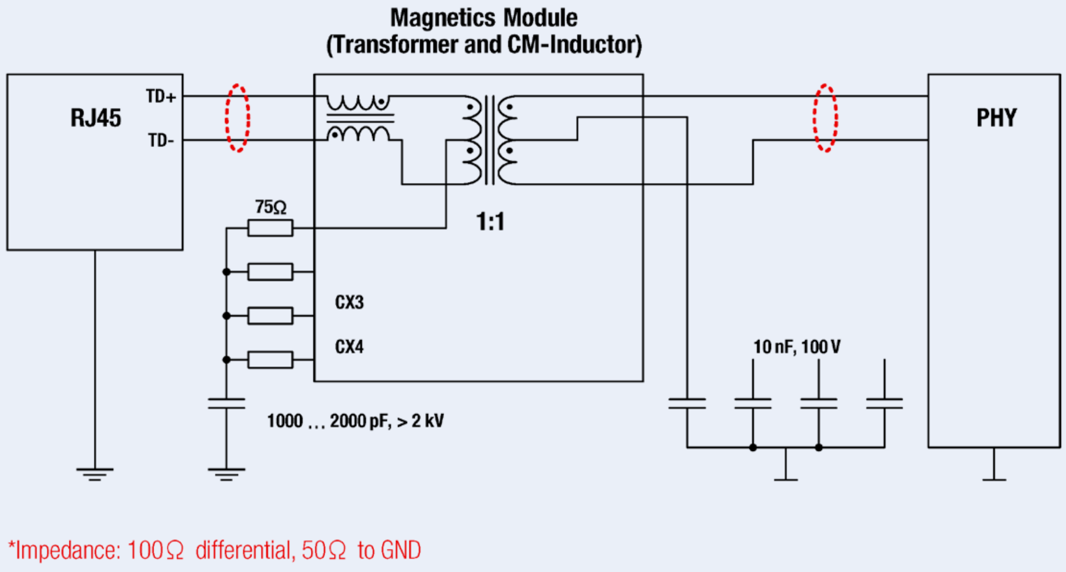


*Figure 1: Basic circuit of a Gigabit Ethernet interface. Representation of a transmission channel with a total of four channels. (Source: Würth Elektronik)*

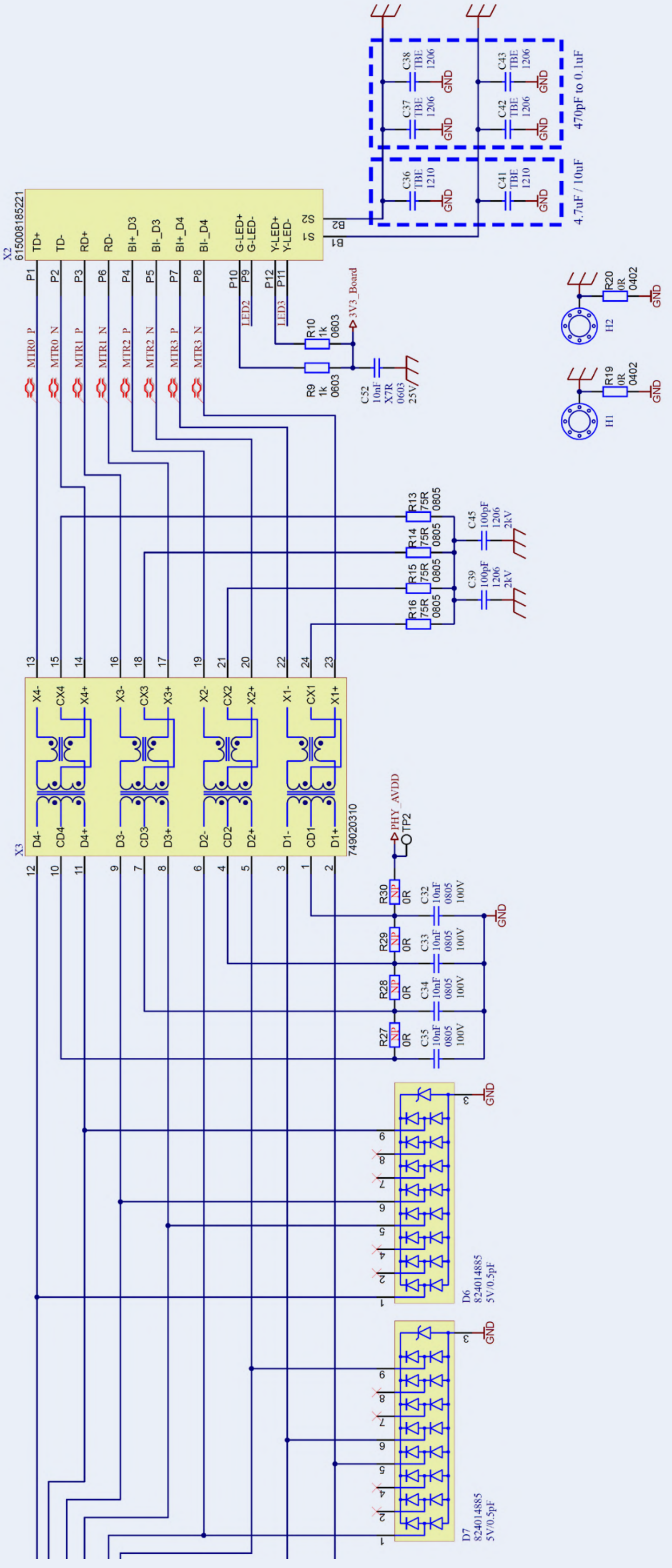


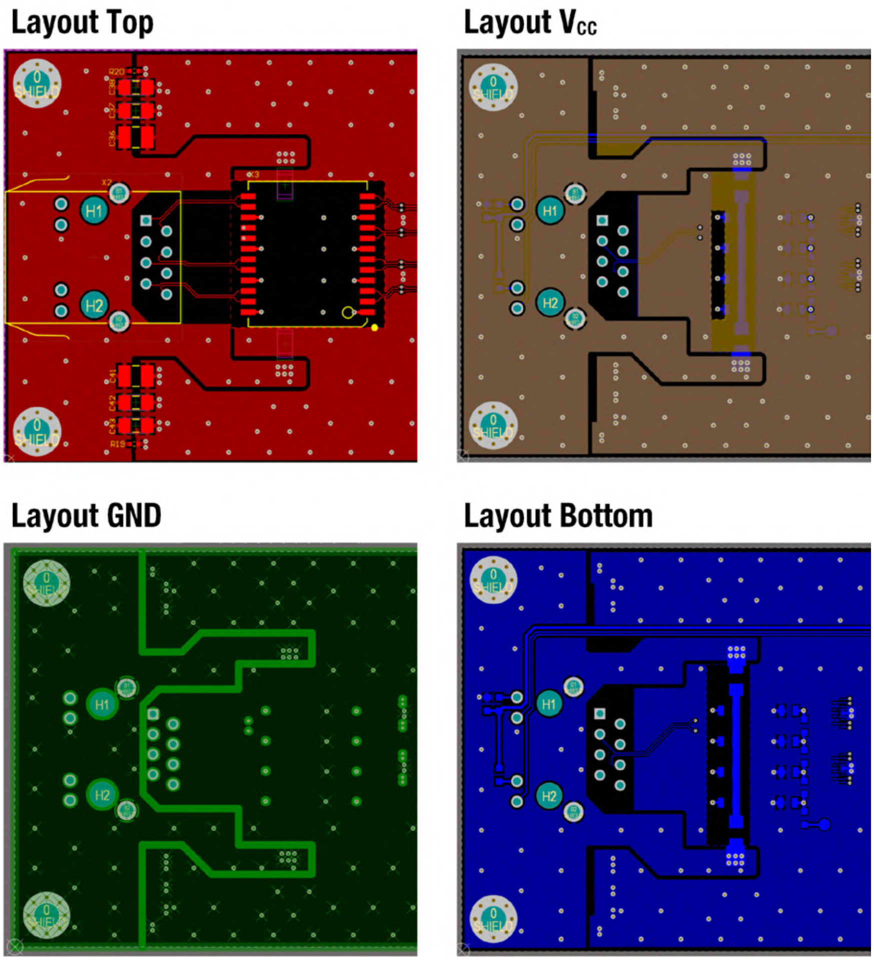
Figure 2: A Gigabit Ethernet interface’s discrete circuit. Module X3 contains the LAN transformers and common-mode chokes necessary to prevent interference.

Here, one 75-Ω resistor is connected to each wire pair to form a “star point”, which is then galvanically isolated and connected to the housing ground via two parallel 100-pF/2-kV capacitors. The additional common-mode chokes integrated in module X3 mitigate interference that is coupled both capacitively and inductively via the long Ethernet cables and could potentially impair Ethernet data communication as common-mode interference.

In Figure 2, R9, R10 and C52 are used to power the LEDs that are typically integrated in the connector socket. Through the capacitors C36 to C38 and C41 to C43, the shielding of the Ethernet socket can be connected to the board ground (GND). For sheet metal housings, it makes sense to omit these capacitors and connect the electronics’ ground (GND) directly to the housing via screw connections. For plastic housings the capacitors should be fitted to connect the shield of the Ethernet cable to the reference ground. The 0-Ω resistors R19 and R20 have the same purpose. However, unlike with capacitors, there is no galvanic separation here. Alternative configurations were included in this section for “experimental” purposes to compare the shielding quality of various Ethernet cables. The capacitors C32 to C35 on the secondary side of the transformers connect the center taps of these in an RF manner to ground (GND). Galvanic isolation via capacitors is essential to eliminate DC equalization currents from the PHY. The resistors R27 to R30 are included to comply with some PHY manufacturers requirements (Current Mode Line Driver - Option) but are typically not needed if the PHY operates in “Standard Voltage Mode”. However, the TVS diode arrays D6 and D7 are indispensable because they isolate transient interferences on the interface side of the PHY from the circuit ground (GND). On the secondary side, i.e. after the transformers of the X3 module, transient disturbances occur in common mode, and therefore a TVS diode must be connected to each terminal of the transformers against the reference ground. The secondary side of the transformer, however, has lower interference levels than the primary side. Important for the TVS diodes to function properly is a low-impedance connection of the diodes, on the one hand looped into the signal lines and on the other hand to ground.

**Figure 3** illustrates the layout of all four layers of the board starting with the Ethernet interface area. The package/socket ground is isolated from the electronics GND in all four layers. Thus, the package ground's surfaces do not overlap with those of other layers to keep capacitive coupling as low as possible. The ground planes were plated through every 4 mm in a grid pattern. The Ethernet socket's signal lines are balanced, with a differential impedance of 100 Ω routed to the reference ground. The conductor pairs have a track width of 0.154 mm and are spaced 0.125 mm apart. The Ethernet socket is positioned on the PCB's edge to ensure a low-impedance connection to a metal enclosure if necessary.

The transformer module (X3) is placed nearby to minimize the effects of electrical coupling, or interference from long traces. As with the primary side, the secondary side of the transformer module must maintain a differential impedance of 100 Ω to the reference ground for the conductor paths. To avoid voltage drop due to parasitic inductance, the TVS arrays must be connected directly into the signal path and to GND.


Layout Top

Layout $V_{CC}$

Layout GND

Layout Bottom

*Figure 3: The layout of all four board layers of the Ethernet interface area.*

## EMC Compliance

In terms of electromagnetic compatibility (EMC), the board complies with industry standards for immunity (EN61000-6-2) and EN55032 Class B radio interference emission levels for multimedia equipment. Numerous factors must be considered when designing a 1 Gbit/s Ethernet interface. These include an RF-compatible circuit and layout design, a system-dependent ground concept, and the right choice of components. Only when all these factors are taken into account a product that functions reliably and meets stringent requirements can be developed. Further information on these topics, as well as on other interface standards, is available in various app notes published by Würth Elektronik at [1]. ⏮

220182-01

## About the Author

Dr.-Ing. Heinz Zenkner is a freelance consultant at Würth Elektronik in the areas of technical marketing and application engineering as well as a lecturer at the technical academy in the area of EMC. At the same time, Heinz is a publicly appointed and sworn EMC expert. He has authored numerous technical journals and books, and has worked as a lecturer at various universities, the IHK and at numerous seminars.

## WEB LINKS

[1] Würth Elektronik Application Guide: https://www.we-online.com/applicationguide/en

# Bringing Real-Time Object Detection

## to MCUs with Edge Impulse FOMO

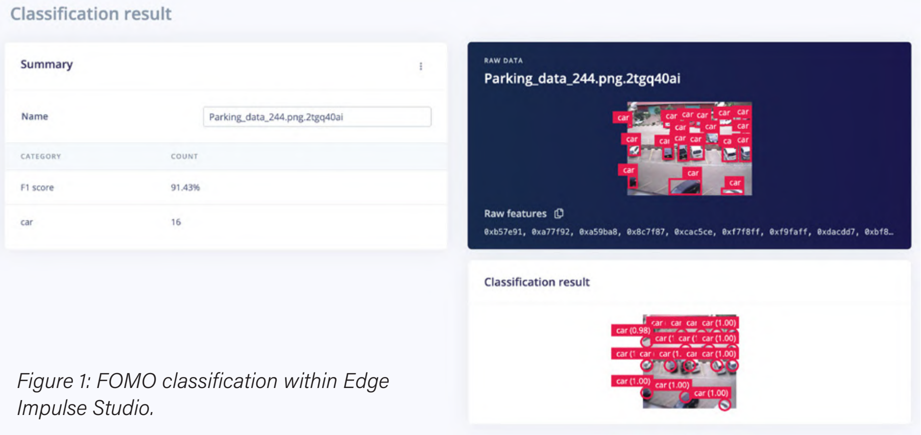
By Jan Jongboom, Edge Impulse

We humans rely heavily on sight to perform many daily tasks, from some of the most basic to the most complex. With one look, we know if there are people in the room with us, if there's an elephant nearby, or how many free parking spaces are available. Despite the importance of vision, though, many embedded devices still can't perceive things visually. Wouldn't it be amazing if we could teach *all our devices* to see the world the way we do?

In recent years, there have been some amazing developments in computer vision, fueling progress in things like self-driving cars and biometric immigration gates (very useful if, like me, you travel a lot!). But these use cases are incredibly computationally expensive, requiring costly GPUs or special accelerators to run.

The awesome thing is that not all computer-vision tasks require such intensive compute. Any yes/no question ("Do I see an elephant?," "Is this label properly attached to the bottle?") can add tremendous value to constrained embedded devices. What's more, these problems of image classification can even be solved by today's microcontrollers.

Imagine if we could add even more advanced vision capabilities to every embedded device!



Classification result

Summary

Name: Parking_data_244.png.2tgq40ai

| CATEGORY | COUNT |
|---|---|
| F1 score | 91.43% |
| car | 16 |

RAW DATA

Parking_data_244.png.2tgq40ai

Raw features

```
0xb57e91, 0xa77f92, 0xa59ba8, 0x8c7f87, 0xcac5ce, 0xf7f8ff, 0xf9faff, 0xdacdd7, 0xbf8…
```

Classification result

Figure 1: FOMO classification within Edge Impulse Studio.

### Say Hello to FOMO

We're making it a reality. We developed a novel neural network architecture for object detection called Faster Objects, More Objects, or FOMO (**Figure 1**). It's designed from the ground up to run in real-time on microcontrollers, so embedded engineers can (ahem) avoid the fear of missing out when it comes to computer vision.

### Fast, Lean & Flexible

FOMO is capable of running on a 32-bit MCU, like an Arm Cortex-M7, with a frame rate of 30 frames per second. And the next time you choose a Raspberry Pi 4 type device, you'll be able to do object detection at a rate of about 60 frames a second. That's roughly 30 times faster than MobileNet SSD or YOLOv5.

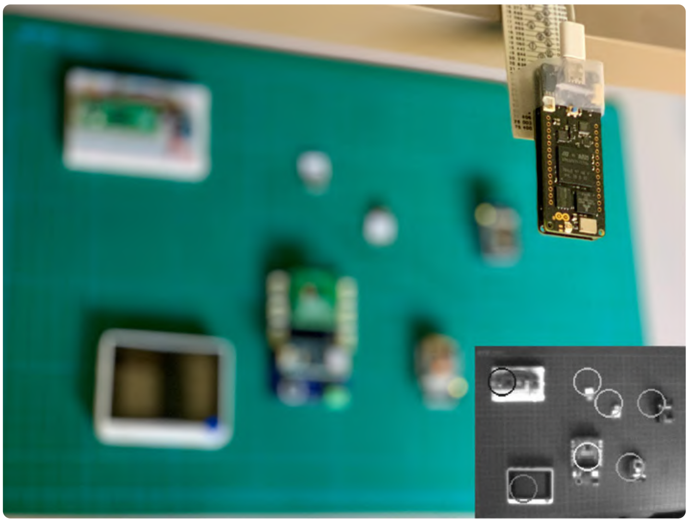


Figure 2: Run object detection on a wide variety of dev boards, including the Arduino Portenta.

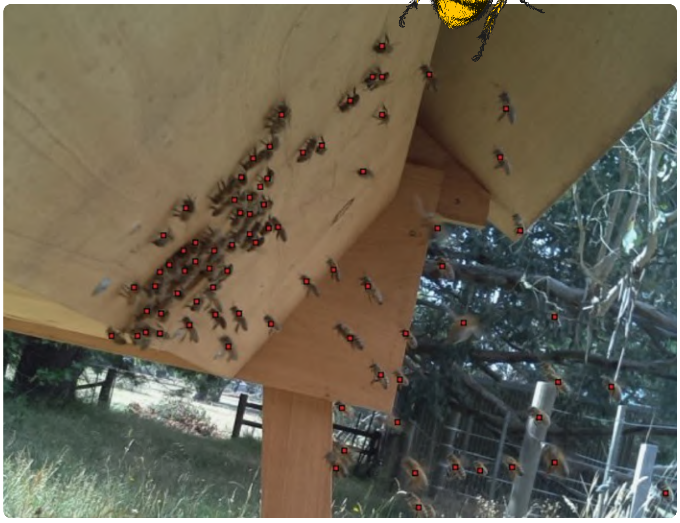


Figure 3: Here's a former iteration of the FOMO approach used to count individual bees.

FOMO scales down to about 100 kilobytes in RAM, making it possible to run object detection in real-time on everything from highly-constrained Arm Cortex-M4 cores to more powerful ones, like the Cortex-M7 cores on the Arduino Portenta H7 (**Figure 2**), the new Arduino Nicla Vision (another dual Arm Cortex-M7/M4 CPU), or even specialized DSPs such as the Himax WE-I.

FOMO can scale from the tiniest microcontrollers all the way to full gateways or GPUs. This high degree of flexibility also makes FOMO useful when fault detection requires identifying variations that are very, very small within an image.

In an MCU with strictly limited compute and memory capacity, it's best to use an image size of about 96x96 pixels. But with a larger microcontroller device, 160x160 pixels is probably fine. The important thing is that FOMO is fully convolutional, so it works on any arbitrary input size. If you need higher granularity, more detail, or more objects, you can just scale up the input resolution.

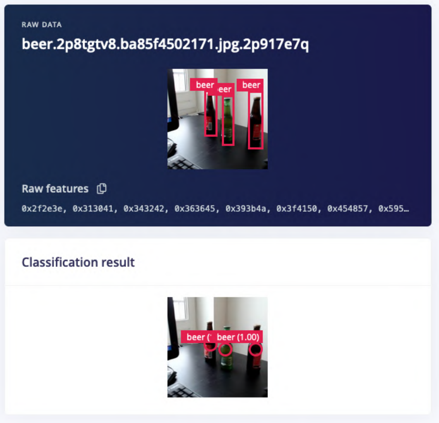
## It Sees the Little Stuff

As long as the objects in the frame are of similar size and don't overlap, this new architecture can even spot and count lots of very small objects very effectively (**Figure 3**). That's something that MobileNet SSD and YOLOv5, despite being larger and more capable models, can't do very well.

## No More Missing Out

FOMO is available today, runs on a wide variety of computing platforms, and is compatible with Linux systems, Cortex-M microcontrollers, and specialized DSPs. Add a camera and Edge Impulse, and you're all set.

With FOMO, you can quickly add object detection to just about any camera-based device, and avoid the fear of missing out that, until now, embedded engineers have had to deal with when it came to computer vision (**Figure 4**).



RAW DATA

beer.2p8tgtv8.ba85f4502171.jpg.2p917e7q

Raw features

```
0x2f2e3e, 0x313041, 0x343242, 0x363645, 0x393b4a, 0x3f4150, 0x454857, 0x595…
```

Classification result

Figure 4: Training on the centroids of beer bottles. On top the source labels, at the bottom the inference result.

To learn more about FOMO and experiment with your own algorithm, visit **edgeimpulse.com/fomo**.

220207-01

## About the Author

Jan Jongboom is an embedded engineer and machine learning advocate, always looking for ways to gather more intelligence from the real world. He has shipped devices, worked on the latest network tech, simulated microcontrollers and there's even a monument in San Francisco with his name on it. Currently he serves as the cofounder and CTO of Edge Impulse, the leading development platform for embedded machine learning with 80,000+ projects.

# Traveling Wave Tubes

## Peculiar Parts, the Series

By **Neil Gruending**

The world of RF amplifiers is fascinating because of the wide range of methods in use. Today, many are constructed from solid-state components, but there are still situations where vacuum tubes are the only suitable choice. We've looked at klystrons in the past, so, this time, let's look at the traveling-wave tube amplifier, another unsung electronic hero.

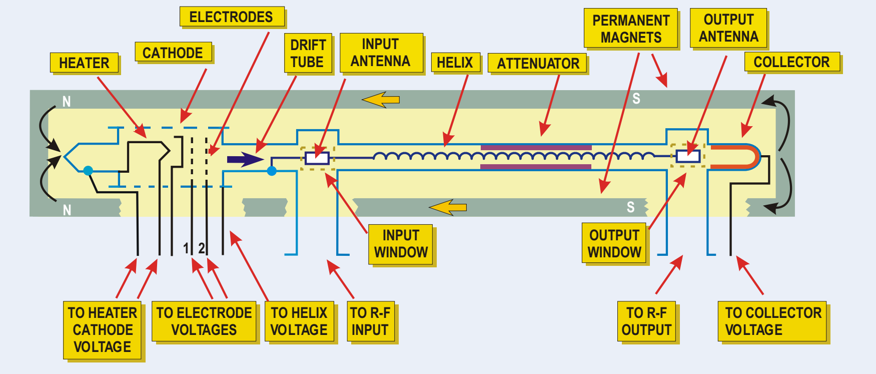


*Figure 1: Basic design of a traveling-wave tube.*

One of the most fascinating things about traveling-wave tube amplifiers is how they work. They feature a heater, cathode, and acceleration electrodes to form an electron gun, much like a cathode-ray tube, that beams a stream of electrons to the collector (**Figure 1**). This stream is focused by an external magnetic field that is usually made of permanent magnets. Using velocity modulation, which mixes the electron stream with the incoming RF electrons, the tube amplifies the applied RF input signal.

Since the streaming electrons travel much more slowly than the RF electrons, the RF signal is fed through a spiral wound wire, called the helix. This slows the RF signal down to match that of the electron stream.

As the RF electrons proceed down the helix, they modulate the velocity of the electrons in the stream because the in-phase electrons speed up, and the out-of-phase electrons slow down. These modulated electrons then bunch together, inducing an amplified signal back into the helix that is then picked off the end of the helix using a directional coupler.

When compared to klystrons [1], traveling-wave tubes have the advantage of wider bandwidths. Additionally, they don't require resonant components, making them ideal for lower power microwave applications like radar or even spacecraft and satellites. A great example is the Collins Radio S-Band amplifier (**Figure 2** and **Figure 3**) used in the Apollo space program [2]. It was a compact, 32-pound, 20-W amplifier that transmitted all of the voice, data, and television back to NASA's network of 26-m, earth-based dish antennas. By comparison, the ground station used a focused 10,000-W signal to communicate back to the craft.

Even though traveling-wave tubes are mostly the domain of commercial applications, a small group of enthusiasts still like experimenting with these wonderful little amplifiers in amateur microwave transmitters [3]. Their biggest challenge, however, is finding the tubes!

210418-01

### Questions or Comments?

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

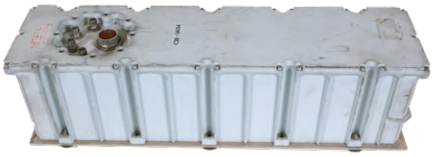


*Figure 2: Collins Radio S-Band traveling-wave tube amplifier used for communication with earth during the Apollo mission. (Source: Ken Shirriff)*

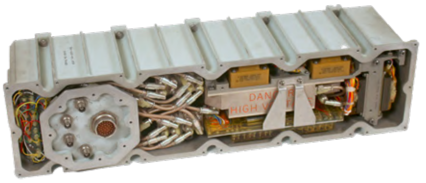


*Figure 3: Operating at several thousand volts, the Collins Radio amplifier was a tightly-packed tangle of coaxial cables. (Source: Ken Shirriff)*

### WEB LINKS

[1] N. Gruending, "Klystrons, Weird Component # 12," Elektor 3/2015: https://bit.ly/2UW4k9G
[2] K. Shirriff, "Inside a 20-Watt Traveling-Wave Tube Amplifier from Apollo," Ken Shirriff's Blog, July 2021: https://bit.ly/3ea8lOn
[3] H. Griffiths, "Travelling Wave Tube Amplifiers," The National Valve Museum, September 1980: https://bit.ly/3wA8aCn

# Narrowband Internet of Things

## Standards, Coverage, Agreements and Modules

By **Tam Hanna** (Slovakia)

Curious about Narrowband Internet of Things (NB-IoT)? Is it for you? Let's take a look.

source: shutterstock.com

Along with LoRa and Sigfox, mobile communication networks are also a good option for the transmission of IoT sensor data. The upgrade from EDGE to UMTS made this option even more attractive, since using a faster transmission system can in the end be better than using a lower-power but slower system. However, the immense bandwidth and power hunger of 4G/LTE make this rule of thumb a bit less relevant. The power consumption of the transmitters is significantly higher, and on top of that the modules are more expensive. Nevertheless, it can be worthwhile.

In the framework of the specification *3GPP Release 13*, designated by LTE as 'informational', the GSM Association defines two systems for the Internet of Things. The first is Narrowband Internet of Things (NB-IoT), and the second is LTE-M, also known as LTE Cat-M1 or eMTC.

LTE-M is basically a 'light' version of LTE (4G) with a bandwidth of 1.4 MHz, while NB-IoT is a dedicated wireless communication standard for the Internet of Things. The key difference is that LTE-M additionally supports voice transmission with VoLTE, while an NB-IoT system exclusively transmits data messages.

The NB-IoT channels, each with a width of only 180 kHz, use a subset of the methods implemented in the full version of LTE. Uplink uses a simple version of the frequency division multiple access (FDMA) method, while downlink uses orthogonal FDMA (OFDMA). The quadrature phase shift keying (QPSK) modulation method does not require especially complex hardware in terms of processing power.

However, it should be noted that introducing NB-IoT usually incurs

Table 1: Frequency bands.

| Region | Bands |
|---|---|
| Europe | 3, 8, 20 |
| (Former) CIS countries | 3, 8, 20 |
| North America | 2, 4, 5, 12, 66, 71, 26 |
| Asia Pacific (APAC) | 1, 3, 5, 8, 18, 20, 26, 28 |
| Sub-Saharan Africa | 3, 8 |
| Middle East and parts of North America | 8,20 |
| Latin America | 2, 3, 5, 29 |

additional costs for the carrier for the new hardware. Due to the extremely narrow bandwidth, NB-IoT can easily fit into the guard band surrounding the LTE frequency packets. On the other hand, using NB-IoT in stand-alone mode is of course also possible.

## Looking at Performance

Even the technically most attractive wireless communication standard is of no use if the transmission capability is insufficient for the intended task. In the case of NB-IoT, the version is an important consideration because there are differences between LTE Cat NB1 (Release 13) and LTE Cat NB2 (Release 14). The older version can only achieve 26 kbit/s in upstream, but Cat NB2 is significantly faster with 127 kbit/s upstream and 159 kbit/s downstream. For comparison, conventional (not HSDPA) 3G initially achieved 380 kbit/s. LTE Cat M1 currently runs at around 1 Mbit/s upstream and downstream, and Release 14 raises this to 4 Mbit/s upstream and 7 Mbit/s downstream.

The differences in latency times are enormous. LTE-M can usually achieve 15 ms, while with NB-IoT the recommended 'working range' is from 1.6 s to as much as 10 s. The module manufacturer Sierra Wireless, especially popular in the USA, describes the situation as follows:

"Another important fact to consider is that there are no NB-IoT use cases that LTE-M can't also support. In other words, LTE-M supports any LPWA application, whereas NB-IoT is designed for simpler static sensor type applications."[1]

In addition, only version 2 of the NB-IoT standard supports position data provision by the network operator. If the module does not have GPS capability or you want to do without an external antenna, you can use this approach to obtain basic position data. Release 14 also accelerates searching for new cells, which is mainly beneficial for moving devices. Despite these new benefits of Cat NB2, LTE-M is still the best choice for automotive and other mobile applications because it provides smarter cell handover. The final improvement concerns transmit power: Super Low Power transmitters [2], which can operate with only 14 dBm, are only allowed in Release 14.

If at some point in time you got your hands on a 4G module for Verizon, the natural question here is which bands are used. Band 13, which is only important for North America, has caused problems for many Asian or European module providers. **Table 1** is taken from the *Deployment Guide* [3] of the GSM Association. You should make sure that the module you choose supports all the bands that are used by your preferred carrier.

## Availability and Agreements

It goes without saying that wireless communication standards are only worthwhile if they are also available in practice. In the case of the two IoT wireless communication standards, you should have a look at the interactive world map of the GSM Association in **Figure 1** [3] (status as of September 2021). As you can see, Mexico is the only country where only CAT-M is available (probably because of the larger range), while "NB-IoT only" is more widely available in the rural areas of Asian countries and, remarkably enough, in Eastern Europe. In the highly industrialised regions of Europe, North America, Asia, Australia and Oceania, both versions are available.

CAT-M agreements are generally ordinary agreements in which the total usage volume and the number of SIM cards determine the overall cost. For the sake of completeness, it should be noted that with regard to cost, an IoT provider such as PodGroup is often a better choice than a prepaid SIM card purchased on the open market.

The claim that NB-IoT is not subject to duty cycle restrictions is not borne out by the author's practical experience as a consultant. Talking with your mobile provider about IoT connectivity is and will remain a matter of negotiation, and all too often limits on the number of packets in a given time interval will be imposed. Operators rarely publish their exact conditions in this regard, which makes the following statement from T-Mobile USA all the more remarkable:

"Join the first nationwide NB-IoT network to power asset tracking, connected cities, and more. Limited time offer; subject to change. Taxes and fees may be additional. Plan includes 10 single-packet transactions per hour at up to 64 Kbps, up to 12 MB. Full service payment due at activation."[4]

Interestingly, this is only an individual opinion, and Hutchison Holding Ltd has confirmed that the total volume of data traffic (within the bounds of the agreed amount) can be used up in one day. Tom Tesch, the Austrian spokesperson of Hutchinson, says in this regard:

"The data rate of NB-IoT — in accordance with the standard — is very low and primarily suitable for the transmission of individual measurements or status values. For this reason, more than 5 to 10 MB per month is very rarely needed for NB-IoT devices. For more bandwidth-intensive applications, such as the transmission of photos or videos, 3G/4G and of course 5G are more suitable technologies. There are currently no limits on when the volume can or may be used, which means that the entire volume can also be used up in one day."

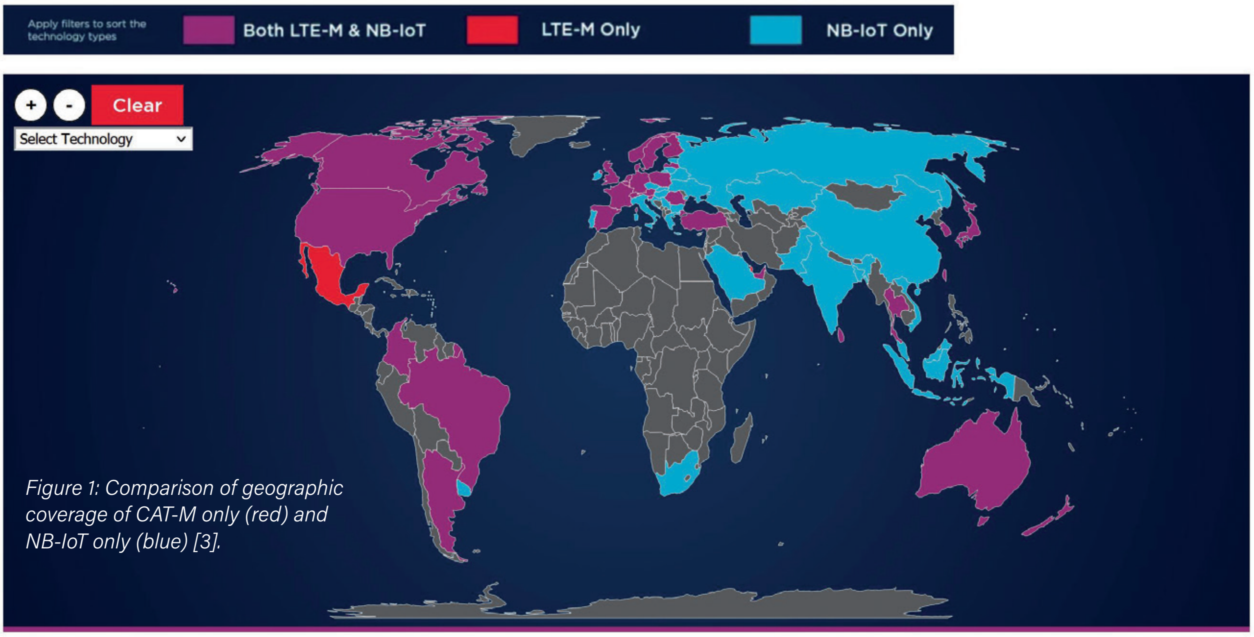


*Figure 1: Comparison of geographic coverage of CAT-M only (red) and NB-IoT only (blue) [3].*

## How to Get Started

After these basic considerations, it's time to start thinking about how you can integrate NB-IoT into practical systems. Of course, development of customer-specific modems is not feasible for most companies, but in the past we have described the 'design in' process for wireless modules in detail, for example in *Elektor* 5-6/2021 [5].

If you don't want to start developing your own board right away, one option is to use a 'turnkey' evaluation board — although the availability of Qualcomm ICs is proving to be a problem in this regard.

Two possibilities are the NBIOT-BG96-SHIELD from Avnet, which integrates a Quectel BG96 module, and the 5G NB IoT click board from MikroElektronika, which hosts a Cinterion module. Arduino also offers a small development board in the form of the MKR NB 1500. However, both of these boards cost more than 50 dollars.

In many cases, it is no longer permitted to supply evaluation board equipped with SIMs, so a massive rollout of applications based on NB-IoT is far from easy. The reason for this is that network operators have not yet packaged the technology for end users. This is also openly admitted by operators, as illustrated by the following statement from Hutchison:

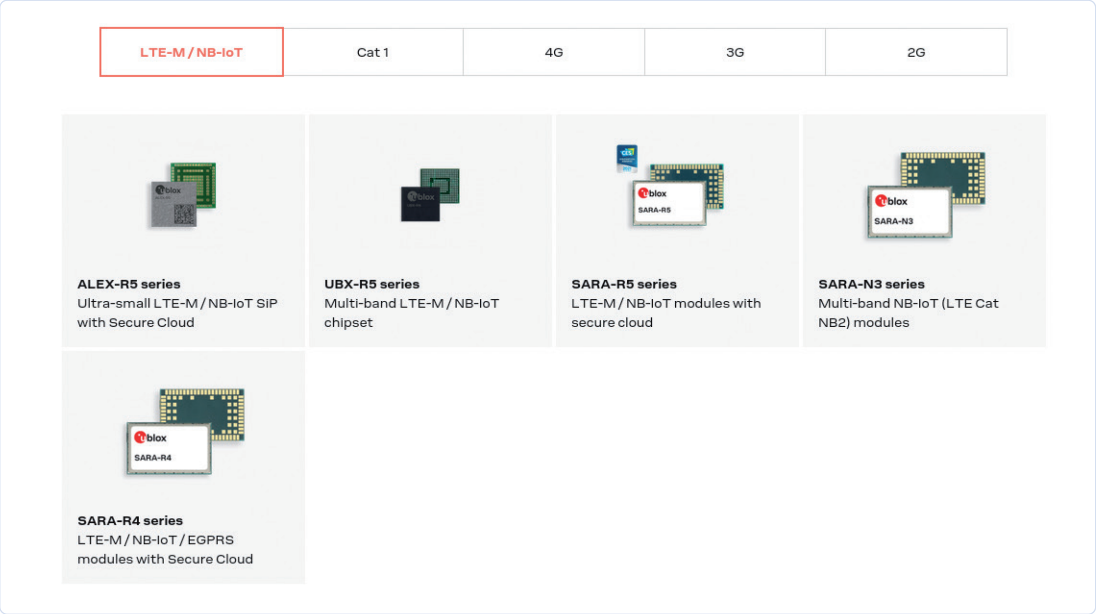
"NB-IoT is a very young and innovative network. As there are hardly any devices available on the market, the target group primarily consists of business customers in the hardware (and software) development environment. This means that our offering is currently exclusively oriented to business customers, for which we create a tailored offer in the course of a consultation process."

When working with 'ordinary' 2G/3G/4G systems, one way to get around this is to use a 'virtual' mobile communication provider such as PodGroup. When asked about this, they answered that NB-IoT is currently not really suitable, especially for 'global' solutions that need to work with a single SIM card.

There are two reasons for this. Firstly, that the NB-IoT rollout is still relatively limited. And secondly, that roaming agreements between the different network operators have generally not yet been adapted to the new NB-IoT wireless communication standard. Liked tax treaties between countries, it takes a long time to achieve such adaptations. In short: international NB-IoT roaming is still in its infancy.

## Is It Worthwhile?

Searching for a practical module that supports only NB-IoT is certainly a very tricky endeavour. Quectel, for example, offers two versions even with the smallest series (BC660): one with only NB-IoT, and the other with both eMTC and NB-IoT. Both wireless standards are also present in larger families, such as the very popular BG95 and BG96. Open market prices for these modules can only be found at SOS Electronic: the BC660K-GL costs €7.63 is small quantities, while the version with LTE-M and NB-IoT is not listed. The price there for the BG96 is €19.



LTE-M / NB-IoT | Cat 1 | 4G | 3G | 2G

**ALEX-R5 series**
Ultra-small LTE-M / NB-IoT SIP with Secure Cloud

**UBX-R5 series**
Multi-band LTE-M / NB-IoT chipset

**SARA-R5 series**
LTE-M / NB-IoT modules with secure cloud

**SARA-N3 series**
Multi-band NB-IoT (LTE Cat NB2) modules

**SARA-R4 series**
LTE-M / NB-IoT / EGPRS modules with Secure Cloud

*Figure 2: Searching for a pure CAT-M module doesn't turn up many results [6].*

A search for u-blox [6] yields more results. The SARA-N3 family includes a module exclusively intended for the NB-IoT set of protocols, but the Swiss company does not offer a pure CAT-M device (see **Figure 2**).

At Gemalto, whose takeover by Thales has made the website even more confusing than it used to be, there is a pure CAT-M module in the form of the EMS31, along with a pure NB-IoT module (ENS22) with the same form factor. At the Czech distributor Sectron you can compare prices: the EMS31 costs €14, the ENS22 only €8.

Information on current consumption (in roundabout form) can be found in the data sheets, which go by the name 'Hardware Interface Description'. The highest current consumption of the EMS31 occurs when operating in Band 4 and is 239 mA with a supply voltage of 3.8 V. For the ENS22, the highest current listed is 404 mA in Band 28, but it should also be noted that wireless modules often require peak currents like this only for a very short time.

*Figure 3: A traditional Stierian windmill serving as a scarecrow. (Source: Martin Geisler, CC BY-SA 4.0 [7]).*

## WEB LINKS

[1] LTE-M vs. NB-IoT: What are the Differences?: https://www.sierrawireless.com/iot-blog/lte-m-vs-nb-iot/
[2] Wikipedia entry on narrowband IoT: https://en.wikipedia.org/wiki/Narrowband_IoT
[3] GSMA: Word map of IoT wireless communication standards: https://www.gsma.com/iot/deployment-map/
[4] T-Mobile Narrowband IoT web page: https://t-mo.co/3EC5Jo4
[5] Tom Hanna, "Do Not Fear the Cellular Module!," Elektor Mag 5-6/2021: https://www.elektormagazine.com/magazine/elektor-175/59527
[6] u-blox mobile communication modules: https://www.u-blox.com/en/cellular-modules
[7] Klapotetz: https://bit.ly/3nOr0Fb

## What's In It for You?

From a technical point of view, NB-IoT works perfectly, and once you have arranged an agreement with a carrier, the effort for network operation is limited to a phone call to your lawyer, unlike the situation with a home LoRa WAN. The relatively low peak and quiescent current consumption of the modules also helps to keep your electricity bill within bounds.

Whether or not it's worthwhile in the end is primarily a matter of scale, just like tax havens such as Dubai or Monaco. If you buy five modems in a year, operate with a 'full' 4G module or, even better, a module with a beefier power adapter and costing a few euros more, the sad experience of the author is that in practice you will repeatedly need the 'other' wireless standard, if only because some base stations do not support every wireless communication standard.

Naturally, the situation looks different if you are purchasing 50,000 modems that will all go to the same customer. If the mayor's office of Großdorf am Klapotetz (**Figure 3**) needs NB-IoT, the local carrier will probably upgrade their network, and the cost savings from the large number of devices will also help.

180021-01

## Questions or Comments?

Do you have technical questions or comments about this article? If so, please contact the author at tamhan@tamoggemon.com or the Elektor editorial staff at editor@elektor.com.

## Contributors

Text: Tam Hanna
Editor: Rolf Gerstendorf
Translation: Kenneth Cox
Layout: Giel Dols

## RELATED PRODUCTS

- **H. Henrik Skovgaard, *IoT Home Hacks with ESP8266* (Elektor, 2020, SKU 19159)**
  www.elektor.com/19159

# Dragino LPS8 Indoor Gateway

Speedy LoRaWAN Gateway Setup

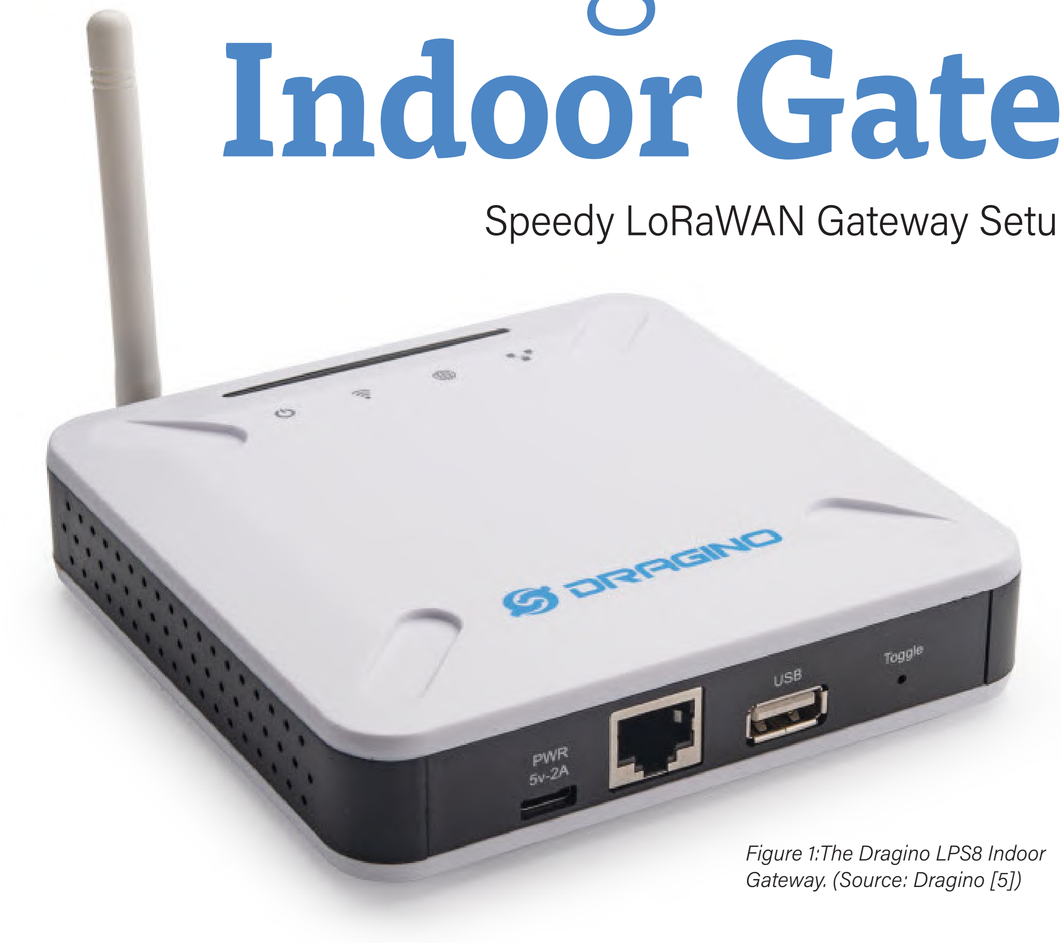


Figure 1:The Dragino LPS8 Indoor Gateway. (Source: Dragino [5])

**By Mathias Claußen (Elektor)**

We have often described how you can interconnect your own electronics devices using a LoRaWAN link. If you are not within range of an existing LoRaWAN gateway, or if you simply want to delve a bit deeper into the topic, you can set up and operate your own gateway. We tried this using the low-cost Dragino LPS8 Indoor Gateway.

LoRaWAN is a topic we have featured many number times in Elektor. It is relatively easy to build a basic LoRaWAN node which has an associated sensor or actuator module. In this type of setup, a LoRaWAN module (which handles communication to the network) is connected to a microcontroller board such as an STMicroelectronics STM32 or Raspberry Pi Pico [1, 2], and this provides an interface to the sensor. In order for the data sent to and from the node via LoRa to be transported further, a remote station is required. In this case, a LoRaWAN gateway will accept the data over the air via LoRa and forward it to an Internet platform like The Things Network (TTN). You can use a pre-existing gateway that's already been set up in your area (many are run by volunteers), or you can set up your own gateway. I have been using a Dragino LPS8 to provide an Indoor Gateway for more than a year now.

## The Dragino LPS8

The Dragino LPS8 Indoor Gateway (**Figure 1**) is housed in a plastic enclosure and could easily be mistaken for a Wi-Fi router. The electronics inside are powered by a small Atheros (today Qualcomm) AR9331 Wi-Fi SoC clocked at 400 MHz which is specifically designed for use in router platforms and access points. With 64-MB RAM and 16-MB Flash, its processing power is not spectacular when compared with something like a Raspberry Pi Zero 2 W, but it is more than enough for the functions that the gateway needs to perform. The SoC also supports Wi-Fi according to 802.11 b/g/n and provides a 10/100 Mbit LAN port. The communication rates available are more than capable of handling the relatively slow data rate used by LoRaWAN. The gateway itself does not have to provide a lot of computing power either, as it only takes care of the integrated LoRa

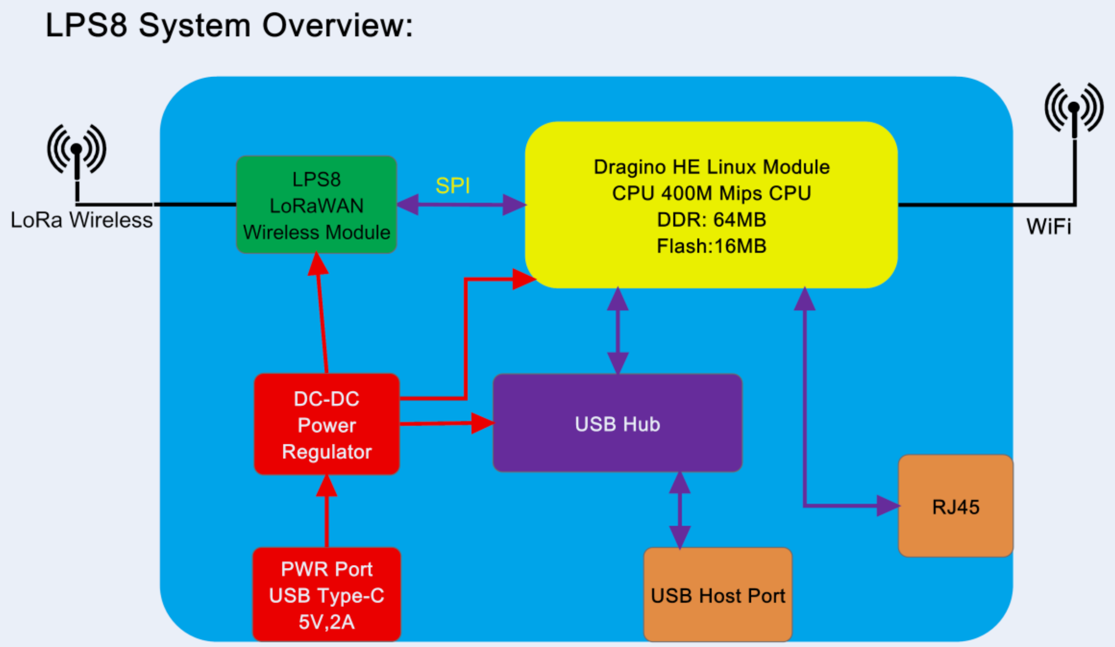


*Figure 2: Block diagram of the Dragino LPS8. (Source: Dragino [6])*

transceiver module and forwards the data to the Internet. A block diagram can be seen in **Figure 2**.

The LoRa transceiver is a combination of a Semtech SX1308 LoRa baseband chip (**Figure 3**) and two SX1257 front-end modules (**Figure 4**). This combination provides the conversion from the radio interface to Ethernet. The gateway is powered via its USB type C port and requires a 5 V/2 A (10 W) mains adapter.

As the name of the gateway suggests, the device is not weatherproof and is intended for use inside a building, so the environment should be dry and relatively dust-free. The building structure and internal walls will reduce the radio coverage compared to an equivalent device mounted outdoors in free space with a mast-mounted antenna.

## The LPS8 Manual, Firmware and Setup

The most recent version of the Dragino manual (available online [3]) describes how to set up the gateway. The manual has been continuously maintained since the product was released and reflects the features and updates of the current firmware. This is indeed praiseworthy; I just wish that some other product manufacturers would adopt the same attention to detail when it comes to documentation.

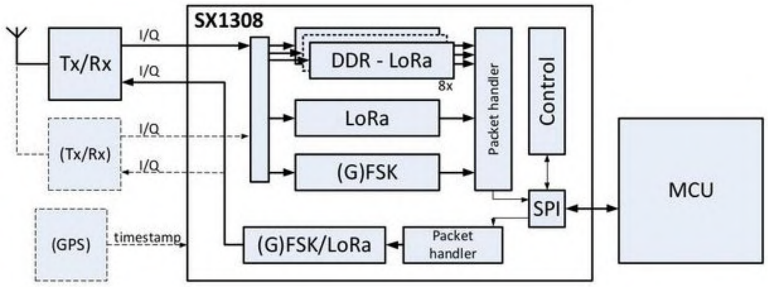


*Figure 3: Block diagram of the SX1308 baseband chip. (Source: Semtech [7])*

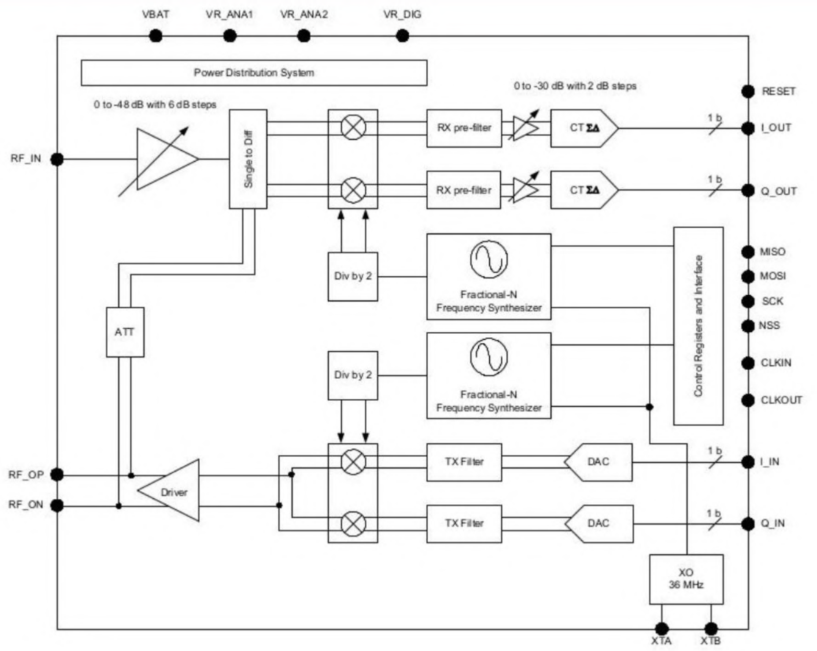


*Figure 4: Block diagram of the SX1257 frontend module. (Source: Semtech [8])*

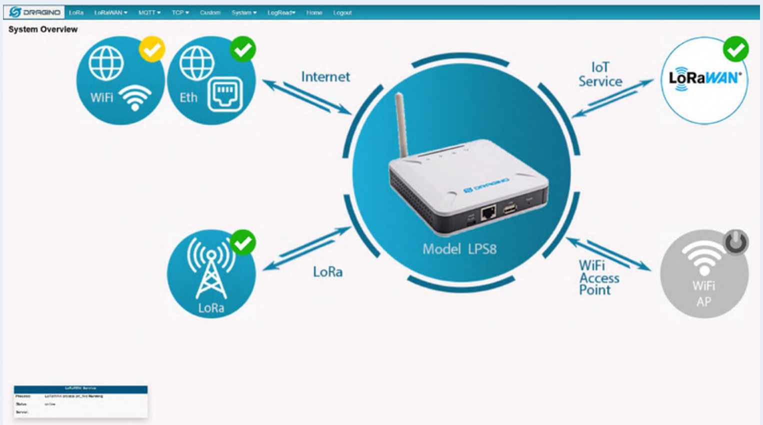


*Figure 5: Communication paths available via the LPS8 gateway.*

The firmware itself is also well maintained. The current release is dated November 4, 2021 (as of December 15, 2021) [4]. It's advisable to update to the latest version before the gateway is put into service. That will ensure any known bugs or security weaknesses should be ironed out as far as possible.

The manual guides you through the setup. All you need to do is configure the network appropriately and make the settings for the LoRaWAN link (e.g., The Things Network). From this point, the LoRaWAN gateway is ready for use (**Figure 5**).

## An OpenWRT Substructure

Even though the first page of the web interface doesn't suggest it, the Linux-based Open Wireless Router (OpenWRT) firmware is used as the basis for the Dragino LPS8 indoor gateway. This not only takes care of the LoRaWAN gateway function, but also provides a number of other settings for the router (IP addresses, forwarding, Wi-Fi).

Thanks to the OpenWRT substructure, an LTE or 5G modem can also be connected to the gateway USB port if no other link to the Internet is possible at the device's location. If you like, you can also access the Linux command line using SSH. (Do so at your own risk!) Additional packages can be installed via the web interface or command line to add more functions to the device.

## A Reliable Solution

I have personally been using a Dragino LPS8 now for over a year. During that time, it has proved to be a low-maintenance and reliable LoRaWAN gateway, which is really all you could ask of such a device. It continues to do a good job servicing my various LoRaWAN nodes and gives excellent coverage throughout the building (and surrounding area). If you are thinking of installing an inexpensive LoRaWAN gateway in a domestic environment, you should take a closer look at the Dragino LPS8 Indoor Gateway, which is currently available from the Elektor Store [5]. ◀

210680-01

## Questions or Comments?

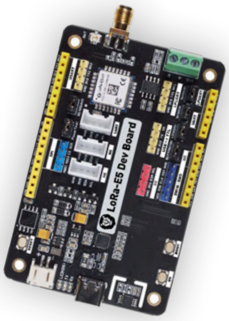
Do you have any technical questions or comments about this article? Contact the author at mathias.claussen@elektor.com or contact the Elektor team at editor@elektor.com.

## Contributors

Text: **Mathias Claußen**
Editor: **Jens Nickel, C.J. Abate**
Translator: **Martin Cooke**
Layout: **Harmen Heida**

## RELATED PRODUCTS

- **Dragino LPS8 Indoor LoRaWAN Gateway (868 MHz) (SKU 19094)**
  www.elektor.com/19094
- **Seeed Studio LoRa-E5 STM32WLE5JC Development Kit (SKU 19956)**
  www.elektor.com/19956



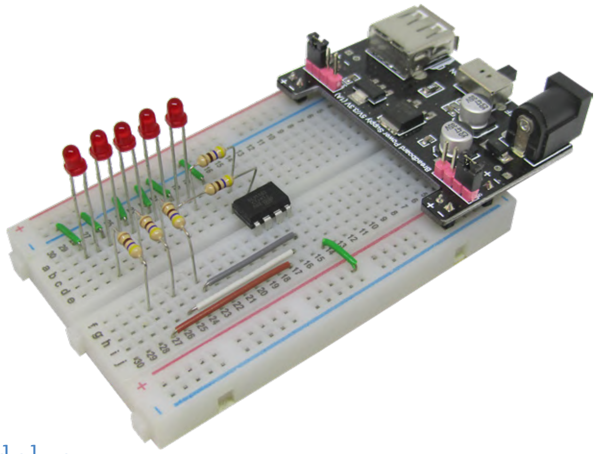
## WEB LINKS

[1] M. Claußen, "My First LoRaWAN," ElektorMag 3-4/2020: http://www.elektormagazine.com/magazine/elektor-141/57159
[2] M. Claußen, "LoRa with the Raspberry Pi Pico," ElektorMag 7-8/2020: http://www.elektormagazine.com/magazine/elektor-179/59721
[3] Dragino LPS8 Indoor Gateway Handbook: http://www.dragino.com/downloads/index.php?dir=LoRa_Gateway/LPS8/
[4] Firmware download of Dragino LPS8 Indoor Gateway: https://bit.ly/LPS8-firmware-release
[5] Gateway picture resource: http://www.dragino.com/media/k2/galleries/148/LPS8-10.jpg
[6] Dragino LPS8 Indoor Gateway Manual: https://bit.ly/LPS8-user-manual
[7] The Semtech SX1257 Front-End Data sheet: https://sforce.co/3fZmy1f
[8] The Semtech SX1308 Transceiver Data sheet: https://sforce.co/32zxAqV

# Explore ATtiny Microcontrollers Using C and Assembly Language

## Sample Chapter: ATtiny I/O Ports

By Warwick A. Smith (South Africa)



I/O ports control the pins of a microcontroller and allow them to be individually configured as input pins or output pins. That's such a broad statement, it should fit just about any newbie course or introduction to microcontroller programming. However, to truly comprehend and exploit a microcontroller's I/O abilities, you need to delve deeper into the machine. In this article, Elektor book author Warwick Smith pulls it off by demo-ing some assembly-code programming on the popular ATtiny micro. Convincing? Let's check it out!

**Editor's Note.** *This article is an excerpt from the 376-page book* Explore ATtiny Microcontrollers using C and Assembly Language (W. Smith, Elektor, 2021) *The excerpt* was *formatted and lightly edited to match Elektor Mag's editorial standards and page layout. Being an extract from a larger publication, some terms in this article may refer to discussions elsewhere in the book. 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.*

I/O ports are configured and controlled using a set of four registers in the ATtiny13(A) and ATtiny25/45/85. When a pin is configured as an input pin, a program running on the AVR can read the logic level on the pin. If a switch is attached to the pin, the logic level on the pin can be read to see if the switch is open or closed. When a pin is configured as an output pin, it can be used to switch the logic level of the pin high (to logic level 1) or low (to logic level 0). An output pin can be used to drive an LED as was done in the LED blink project described elsewhere in the book.

### Configuring I/O Pins as Outputs in Assembler

In this section, we look at how to configure more than one pin as an output using an 8-pin ATtiny with five LEDs attached to pins with series resistors. Code is used to configure the LEDs as a 5-bit binary counter that counts up from zero.

**Figure 1** shows a circuit diagram of an ATtiny13(A) or ATtiny25/45/85 AVR microcontroller with five LEDs attached to I/O pins PB0 to PB4. Pin PB5 of the ATtiny microcontroller in the circuit is used as the debugWIRE pin for programming and debugging the microcontroller. Remember that to use this configuration a programmer/debugger, such as an Atmel-ICE, or AVR Dragon must be used.

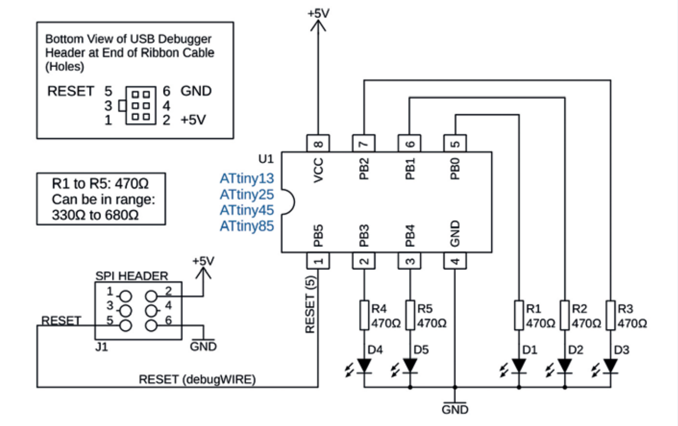


*Figure 1: Five-LED Counter circuit diagram.*

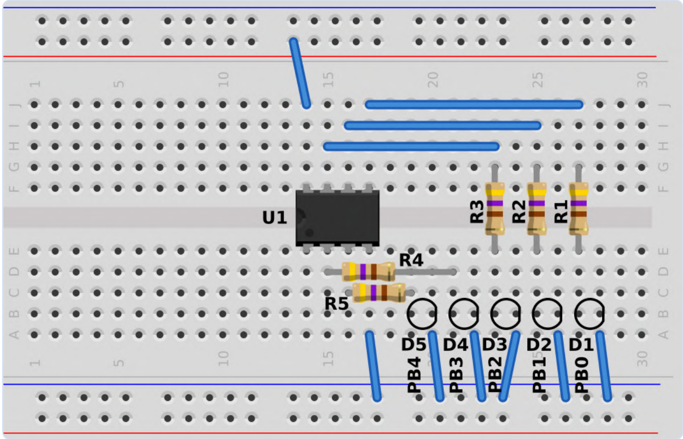


*Figure 2: Five-LED count circuit breadboard layout.*

A program-only USB programmer will not work in debugWIRE mode, and can not perform debugging. Reminder: **Do not set the `DWEN` fuse with a program-only USB programmer because it will not be able to take the AVR back out of debugWIRE mode**. The idea of using debugWIRE mode with this example circuit is to free up the other pins of the AVR that are normally used in ISP/SPI programming mode.

### Program-Only Programmers

USB programmers with program-only capabilities can be used with the circuit of Figure 1 to load the example program that follows and see the count value displayed on the LEDs.

Wire the LEDs as shown in Figure 1 and **Figure 2**. The original USBasp design and the original USBtinyISP design both have protection resistors on the lines that drive out from these programmers, protecting the programmer, and the target AVR chip. The Arduino Uno programmed as an ArduinoISP does not have any protection resistors, but these can be added on the Arduino Uno MOSI line and SCK line which drive into the target AVR. 270-Ω protection resistors are used in the original USBasp design, and are placed on the MOSI, SCK and RESET lines. 1k5 resistors are used on the MOSI and RESET lines of the original USBtinyISP design.

Protection resistors prevent a short-circuit should one of the target AVR pins drive an output voltage at the same time that the programmer drives an output voltage of opposite polarity.

### Peripheral Hardware Devices Interfering with Programming

Although the circuit of Figure 1 can be programmed using the ISP/SPI interface with the LEDs and series resistors attached, other circuits may have hardware attached that interferes with programming. If any peripheral hardware attached to pins of an AVR to be programmed will interfere with programming the AVR, there are some solutions to this problem. Of course, those readers who have a debugWIRE capable USB programmer/debugger, such as the Atmel-ICE, can simply put the target AVR into debugWIRE mode, and thereby use only one pin for programming. Another solution is to program the AVR on another breadboard using ISP/SPI, and then plug it into the target circuit afterwards.

Alternatively, an AVR with more pins can be used, but the free port pins don't always match the same pins from the same port of an 8-pin PDIP ATtiny. For example, if using port pins PB0 to PB4 like the circuit of Figure 1 does, there are not five consecutive free pins available on the 14-pin ATtiny24/44/84 range of AVRs, because the ISP/SPI pins use up pins from both port A and port B. On the 20-pin ATtiny26/261/461/861 range, the whole of port A is free with an ISP/SPI programmer connected, but this means that the software needs to be changed to use port A instead of port B. Fortunately, the 20-pin ATtiny2313/4313 range does have pins PB0 to PB4 free with an ISP/SPI programmer connected.

### Putting the AVR into debugWIRE Mode

In order for the microcontroller to be programmed using a single debugWIRE line as shown in Figure 1, it is necessary to first connect all of the ISP header lines to the microcontroller from the USB programmer/debugger and then set the `DWEN` fuse of the AVR to put it into debugWIRE mode. Your AVR will now be in debugWIRE mode. If not, attach your programmer/debugger, such as an Atmel-ICE or AVR Dragon, and ensure you are ready and able to set the `DWEN` fuse. Once the `DWEN` fuse is programmed, all of the ISP header connections can be removed from the circuit except RESET, +5 V (Vcc) and GND as Figure 1 shows.

## Build the 5-LED ATtiny Circuit on Breadboard

If you have the hardware, then build the circuit of Figure 1 on an electronic breadboard. Make sure that the five LEDs are connected in a row on the breadboard with pins PB0 to PB4 connected in order starting at PB0 on the right. LED D1 will then be on the right of the row and D5 on the left. In other words, we want a single row of LEDs with PB0 connected to the LED on the right, PB1 connected to the LED next to it on the left, PB2 connected to the LED third from the right, and so on, as can be seen in the breadboard layout of Figure 2. The image shows just the outlines of the LEDs so that they do not block the view of the wire connections and resistors. If you do not have the hardware, then follow the programs using a simulator.

## Assembly Code for the 5-LED Count Circuit

Start a new Microchip Studio AVR Assembler Project called *led_count_asm*. Type the code shown in **Listing 1** into *main.asm* of the project, replacing the skeleton code.

If you are using the hardware of Figure 1, then select the hardware tool (your debugger), such as the Atmel-ICE within Microchip Studio, with debugWIRE as the interface. You do so by clicking the *hammer* icon on the second top toolbar. If you are using the simulator, select *Simulator* as the tool instead. If using a "hobby" programmer, read on to be prompted to load the program to the target ATtiny.

The *led_count_asm* program configures pins PB0 to PB4 as output pins so that the LEDs attached to these pins can be driven on and off by the program. The program displays an incrementing binary number, or count, on the LEDs starting at 0 (represented by all LEDs off). When the count reaches its maximum value (all LEDs on) it wraps around to zero and starts counting up again. Each "off" LED represents a binary zero digit or logic 0 level, and each "on" LED represents a binary one digit, or logic 1 level. It is important to lay out the LEDs as shown in Figure 2 so that the count displays correctly with PB0/D1 as the LSB and PB4/D5 as the MSB of the count value.

Build the program and load it to the AVR if you are using the physical hardware of Figure 1 and Figure 2. If using an Atmel-ICE or AVR Dragon, use the *Start Without Debugging* icon on the top toolbar of Microchip Studio or keyboard shortcut *Ctrl + Alt + F5* to load the program to the AVR. If the debugger interface is set up correctly and the chip is in debugWIRE mode, the program will load and start running. The incrementing binary count will be seen on the LEDs. If using a program-only USB programmer such as a hobby programmer, then load the program to the target AVR using the appropriate function. If the program was typed correctly, the circuit wired correctly, and the program was saved and built after typing it in, you will see the binary count value counting up on the LEDs and you can use the simulator with the rest of the text that follows.

If you don't have the physical hardware, you can still see the count value by using the Simulator in Microchip Studio. The program can be stepped through by using the *Start Debugging* and *Break* icons. With the simulator started, open the I/O window from the top menu by clicking *Debug Windows I/O*, and then click on the *I/O Port (PORTB)* item. Step over the program using the *Step Over* icon or *F10* keyboard key. Look at the PORTB item at the bottom of the I/O window to see the count value that would be displayed on the LEDs if they were attached. The count is updated in PORTB each time that the `OUT` instruction is stepped over in the main loop.

## How the LED Count Assembler Program Works

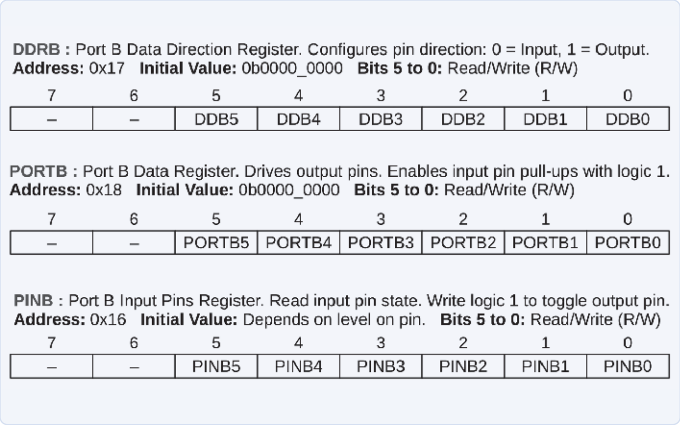
Half of the *led_count_asm* program code consists of the delay subroutine that was used in the *LED blink* program discussed elsewhere in the book. This subroutine is called once in the main loop of the program so that the count on the LEDs is visible to the eye without flashing past too fast. Have the *led_count_asm* project open in Microchip Studio, with the code from *main.asm* also open while following the explanation of the code that follows.

The first two instructions of the program are used to set pins PB0 to PB4 as output pins to drive the LEDs by setting bits in the DDRB register. **Figure 3** shows the DDRB register at the top of the figure. Each bit in this register corresponds to a pin on the microcontroller. For example, bit DDB0 corresponds to pin PB0, DDB1 corresponds to pin PB1, and so on.

When a bit in DDRB is set to logic 1, the corresponding pin becomes an output pin. If a bit in DDRB is cleared to logic 0, the corresponding pin becomes an input pin which is the default of all pins at power-up or reset. At the top of the program, 0b0001_1111 is first written to register R16, and then written from R16 to the DDRB register using the OUT instruction. It is necessary to first load the constant value to R16 because there is no instruction to directly write a constant value to an I/O register such as DDRB. Writing 0b0001_1111 to DDRB sets bits DDB0 to DDB4 making pins

**Listing 1: led_count_asm : main.asm**

```
    ; Set up pins PB0 to PB4 as output pins
     ldi     r16, 0b0001_1111
     out     DDRB, r16
     clr     r18              ; Clear count register
loop:
     out     PORTB, r18       ; Display count on LEDs
     rcall   delay
     inc     r18              ; Increment count
     andi    r18, 0b0001_1111 ; Clear unused bits
     rjmp    loop
; Delay subroutine
delay:
     ldi     r16, 0xff
delloop1:
     ldi     r17, 0xff
delloop2:
     dec     r17
     brb     SREG_Z, delloop2
     dec     r16
     brbc    SREG_Z, delloop1
     ret
```



**DDRB** : Port B Data Direction Register. Configures pin direction: 0 = Input, 1 = Output.
**Address:** 0x17 **Initial Value:** 0b0000_0000 **Bits 5 to 0:** Read/Write (R/W)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| – | – | DDB5 | DDB4 | DDB3 | DDB2 | DDB1 | DDB0 |

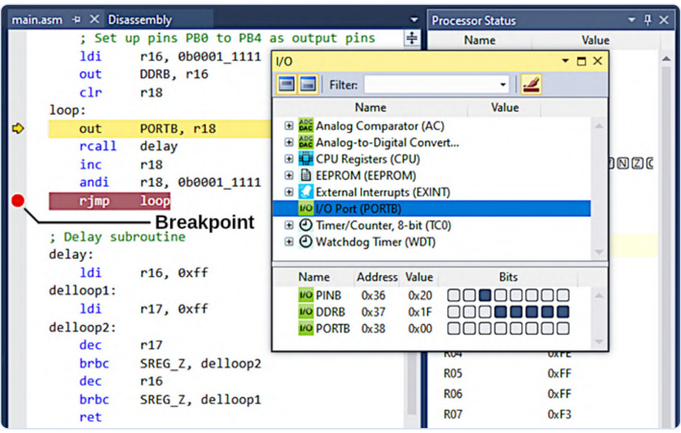
**PORTB** : Port B Data Register. Drives output pins. Enables input pin pull-ups with logic 1.
**Address:** 0x18 **Initial Value:** 0b0000_0000 **Bits 5 to 0:** Read/Write (R/W)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| – | – | PORTB5 | PORTB4 | PORTB3 | PORTB2 | PORTB1 | PORTB0 |

**PINB** : Port B Input Pins Register. Read input pin state. Write logic 1 to toggle output pin.
**Address:** 0x16 **Initial Value:** Depends on level on pin. **Bits 5 to 0:** Read/Write (R/W)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| – | – | PINB5 | PINB4 | PINB3 | PINB2 | PINB1 | PINB0 |

*Figure 3: ATtiny13(A) and ATtiny25/45/85 I/O port registers.*



```
    ; Set up pins PB0 to PB4 as output pins
    ldi     r16, 0b0001_1111
    out     DDRB, r16
    clr     r18
loop:
    out     PORTB, r18
    rcall   delay
    inc     r18
    andi    r18, 0b0001_1111
    rjmp    loop

; Delay subroutine
delay:
    ldi     r16, 0xff
delloop1:
    ldi     r17, 0xff
delloop2:
    dec     r17
    brbc    SREG_Z, delloop2
    dec     r16
    brbc    SREG_Z, delloop1
    ret
```

| Name | Address | Value | Bits |
|---|---|---|---|
| PINB | 0x36 | 0x20 | □□■□□□□□ |
| DDRB | 0x37 | 0x1F | □□□■■■■■ |
| PORTB | 0x38 | 0x00 | □□□□□□□□ |

| Name | Value |
|---|---|
| R05 | 0xFF |
| R06 | 0xFF |
| R07 | 0xF3 |

*Figure 4: Inserting a Breakpoint in the Microchip Studio Debugger.*

PB0 to PB4 output pins.

Although there are four registers for controlling I/O port B, only three are shown in Figure 3. The fourth register, MCUCR has only one bit that applies to port B. This bit is a global pull-up disable bit that we do not use in this chapter. Refer to the Register Description part of the datasheet in the I/O Ports section to see this register — use either the datasheet for the ATtiny13, ATtiny13A, or ATtiny25/45/85.

R18 is used in the program to hold an incrementing count value that is displayed on the LEDs. R18 is cleared to 0 using `CLR` before the main loop so that the count starts from 0.

In the main loop, the contents of R18 are sent to the PORTB register using the OUT instruction. The PORTB register can be seen in the middle of Figure 3. Again, each bit in this register corresponds to a pin on the microcontroller. For pins that were set up as output pins using DDRB, the logic levels written to the PORTB register appear on these pins. For pins that are set up as input pins, a logic 1 in PORTB enables an internal pull-up resistor on these pins. A logic 0 disables the pull-up resistor on the corresponding pin.

Because pins PB0 to PB4 are set up as output pins, the count value written to PORTB appears on these pins as logic levels that switch the LEDs on and off. A logic 1 in the count value switches the corresponding LED on, and a logic 0 level in the count value switches the corresponding LED off.

After the count value is written to PORTB using the `OUT` instruction, the delay subroutine is called to leave the count value on the LEDs for a while. Register R18 is then incremented by 1 using the `INC` instruction so that the count value that it is holding increments. `ANDI` is used to clear the top three bits of the count value in R18, by using a mask value of 0b0001_1111. This ensures that these top bits are always zero. Back at the top of the loop when R18 is written to PORTB again, 0 is always written to the top three bits, because they were cleared using `ANDI`. When the end of the loop is reached at the `RJMP` instruction, program execution starts at the top of the loop again, with the new count value written to PORTB, and displayed on the LEDs.

Some things to note about this program: I/O registers DDRB and PORTB are both used to set up and control port B of the microcontroller in both programs. The LED blink program only needs to control a single pin, so uses the `SBI` and `CBI` instructions to set and clear a single bit in the I/O registers directly — none of the working registers R0 to R31 are needed for this. The LED count program needs to access five LEDs or pins at the same time so uses the `OUT` instruction to write to multiple bits in a register at the same time.

Register usage is important to keep track of in an assembly language program. For example, an immediate value is loaded to R16 at the beginning of the program. R16 is only used temporarily in this instance, so can be used later in the program without the need to first save its value. It is used again in the delay subroutine. Because R16 and R17 are used in the delay subroutine, R18 was chosen to hold the count value and is not used for anything else. This ensures that the count value is never overwritten. If a program gets very big and there are no spare registers to use for dedicated purposes, then `PUSH` and `POP` instructions can be used at the start and end of a subroutine to preserve the values in registers that are used.

## Using a Breakpoint in the Debugger

With the simulator, or the physical hardware and a hardware debugger such as the Atmel-ICE, use the debugger to step through the program by clicking *Start Debugging* and *Break* as we have done before. View the I/O port registers by opening the I/O window in Microchip Studio. With the debugger running, select *Debug Windows I/O* and then click *I/O Port (PORTB)* in the I/O window as shown in **Figure 4**.

When program execution enters the main loop, it is convenient to place a breakpoint on the `RJMP` instruction and execute the entire loop by clicking the *Continue* toolbar button (keyboard key *F5*). A breakpoint can be placed on the `RJMP` instruction by clicking the gray area at the very left of the instruction and of the Microchip Studio window. This places a red dot in this area indicating that a breakpoint has been set on the instruction, as can be seen in Figure 4. Alternatively, click the instruction that you want to place a breakpoint on so that the cursor is placed on it, and then select *Debug Toggle Breakpoint* from the top menu, or use the *F9* keyboard key. Once the breakpoint is set, use the *Continue* toolbar button or press *F5* to step though the entire loop and only break at the bottom. In this way, the count can be seen incrementing by 1 in the PORTB in the *I/O window* as well as in register R18 in the *Processor Status window*, without having to individually step over each instruction in the loop.

Remove the breakpoint from the `RJMP` instruction by clicking the red dot at the left of the instruction, or use the menu or *F9* to toggle the breakpoint off, assuming that the cursor is on the `RJMP` instruction line. Put a breakpoint on the `INC R18` instruction. Now use *F5* to step through the loop. The new breakpoint ensures that the PORTB register and R18 contain the same value when program execution stops. If program execution stops on `RJMP`, then R18 is incremented before writing its new value to PORTB, thus they are unsynchronized when observing these values in the debugger windows in Microchip Studio.

The software released by the author in support of the book is available for free downloading. Head over to [1], scroll down to the *Downloads*, and click on the file name: *Software_Explore ATtiny Microcontrollers using C and Assembly Language*. Save the ZIP archive file locally (approx. 29 kB) and then extract it.

220045-01

## Contributors

Text and Graphics: **Warwick Smith**
Editor: **Jan Buiting**
Layout: **Giel Dols**

## Questions or Comments?

Do you have any technical questions or comments related to this article? Email the author at warwsmi@axxess.co.za or Elektor at editor@elektor.com.

## WEB LINK

[1] Book resources/info page: www.elektor.com/20007

## RELATED PRODUCTS

- Book: W. Smith: *Explore ATtiny Microcontrollers using C and Assembly Language* (SKU 20007)
www.elektor.com/20007

- E-Book: W. Smith, *Explore ATtiny Microcontrollers using C and Assembly Language* (SKU 20008)
www.elektor.com/20008

elektor books
IN-DEPTH
AVR Architecture and Programming
Explore ATtiny Microcontrollers using C and Assembly Language
Warwick A. Smith
elektor

# Err-lectronics

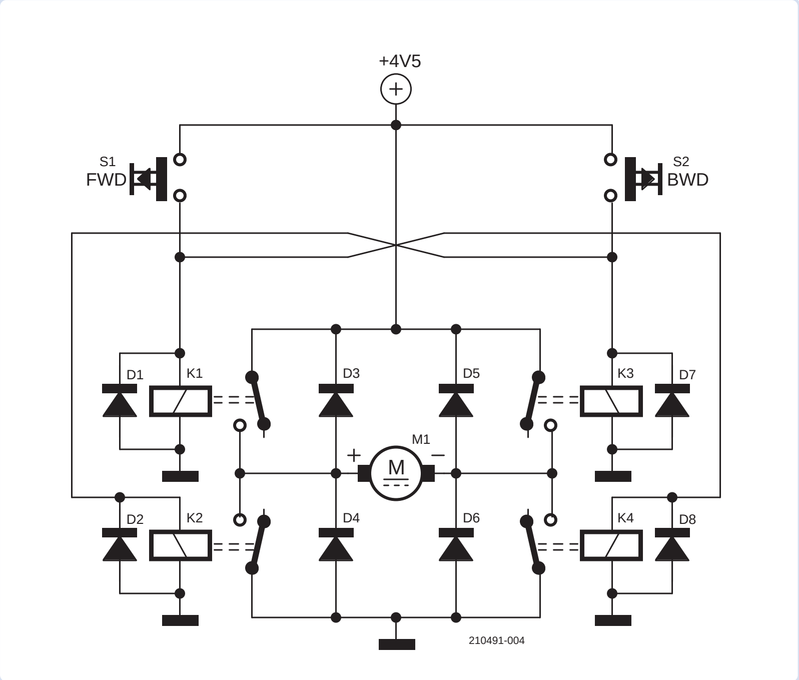
## Corrections, Updates and Readers' Letters

Compiled by Ralf Schmiedel and Jens Nickel (Elektor)

### Driving Motors with H-Bridges

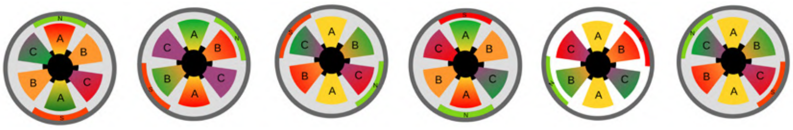
**Elektor 01-02/2022, P. 6 (210491)**

Unfortunately, the H-bridge with relay diagram in this article (Figure 4) is missing a connection between the Motor M- and the common switched contacts of relays K3 and K4. We have included here a corrected version of the diagram. Many thanks to the eagle-eyed readers who spotted this blooper.



Another error crept into the descriptions of Figure 14 and 20. The chip shown is actually an L298N device and not an L294N as suggested in the description.

The last two diagrams shown in Figure 23 do not show the correct sequence of field excitation and rotor orientation. Here is the correct picture:



### Simple Earth-Leakage Tracer

**Elektor 1-2/2022, P. 106 (200576)**

It is essential that the Earth leakage tracer unit described in this article is only used to test equipment which has been completely isolated from the mains supply. It could prove deadly if is used on any piece of equipment that is powered up from the mains! The ground fault meter works well for troubleshooting problems on portable devices, but only when they are in a de-energized and isolated state.

*Jörg Stäudle*

Thanks for emphasizing that point Jörg. You are absolutely correct.

*The Editorial Team*

### A Compass Rose...

**Elektor 09-10/2021, P. 78 (200597)**

I found your article concerning the GY-271 compass module very interesting, but I am not sure why it's necessary to move the sensor through a figure-of-eight pattern?

*Jac Hettema*

I did some research to discover if there is a way I could improve a smart phone's compass accuracy and found several web sites including this one: *https://bit.ly/3rQfdYJ*. Some cell phone experts say that moving the sensor in a figure eight pattern ensures that it is rotated twice through 360° and serves to perform calibration.

*Rolf Hase (Author of the article)*

### Electronic Load for DC and AC

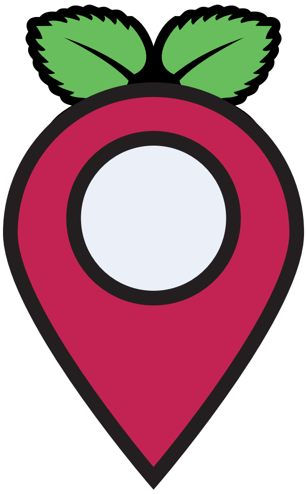
**Elektor 09-10/2021, P. 20 (191206)**

The schematic (Figure 2) indicates that the Zener diode D3 shown on page 23 has a 3.3-V reverse voltage threshold, but it should in fact have a 10.0 V threshold.

220052-01

### Questions or Comments?

If you have any questions or comments, please let us know. You can reach us by email at editor@elektor.com.



# LoRa GPS Tracker Update

## Receive and Show Location Using a Raspberry Pi

**By Hans Schneider (Germany)**

The article, “LoRa GPS Tracker,” featured in the 6/2020 edition of *Elektor* described how to collect the tracking data from a TTN server using Node-RED and then visualize it on a map. In principle, this should work, not only with a PC, but also with a Raspberry Pi. Indeed this is possible, but there are one or two hurdles that need to be overcome. One of our readers found a good solution and describes how he ported the Node-RED flow to the single board computer.

In the article, “LoRa GPS Tracker” (Elektor, Nov/Dec 2020), Mathias Claussen describes hardware and software that can be used to track moving objects. A compact tracker module sends the data from a GPS sensor via LoRa to a gateway, which forwards it to a *The Things Network* server, where the raw data can be called up manually via the Internet, provided the necessary authorization is granted. Of course, this is more convenient when it is accessed automatically via a framework such as Node-RED, which can also control the display on a (world) map. The Node-RED-Flow described in the article works very well on a PC, but this has one major drawback: an energy-guzzling PC needs to be powered up for this purpose and we need the Node-RED server running, otherwise we will get no data.

It’s likely that many tech-savvy households will already have at least one Raspberry Pi on which software such as Pi-Hole [1], Homebridge [2] or Flightradar24 [3] is running and continuously waiting to receive and send data. Why shouldn’t we entrust another task to this Raspberry Pi? One note of warning: the resources used by Node-RED can sometimes conflict with a Homebridge installation. With that in mind, the installation procedure is also not entirely without difficulties.

### Installing Node-RED on the Raspberry Pi

First, we need to install Node-RED on the Raspberry Pi. It is assumed that the reader is familiar with entering commands using the Raspberry Pi Terminal app. Next, we need to log into the terminal via SSH, by using the following command line: `bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)`. You may need to enter `sudo apt install build-essential git` so that npm is able to build the binary modules that need to be installed. The script takes some time to run; you should expect it to take around half an hour to complete.

When the Script is finished, the command `sudo systemctl enable nodered.service` can be input via the Terminal so that Node-RED starts automatically when the Raspberry Pi is powered up or rebooted. This command can be disabled by entering `sudo systemctl disable nodered.service`.

The Node-RED server can be started using `node-red-start`, and stopped with `node-red-stop.` To stop and restart the server use `node-red-restart`. The terminal can be exited after starting Node-RED, and the server will continue to run. When the terminal is restarted, you can start the output service of the server using `node-red-log` and see the output in the terminal. This is useful if you want to check if everything is still working without starting a new instance of the server.

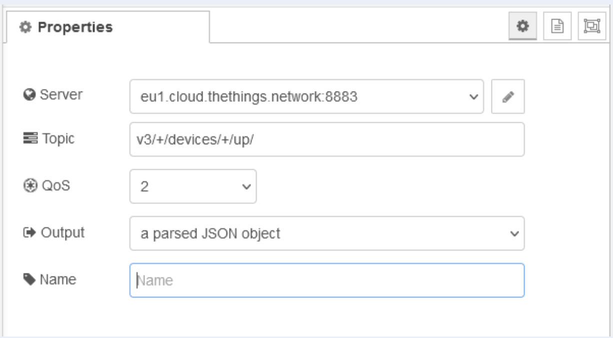
### Installing the Required Modules in Node-RED

The modules can be installed in Node-RED by using either the palette manager in the browser window or by using the npm package manager from the terminal. My recommendation would be to use npm because in my experience that works more reliably. In this case, however, Node-RED must be terminated beforehand.

If, on the other hand, the Node-RED service runs on the Raspberry Pi, you can log into the user interface using a browser on the PC. For this, you need to enter the address line: `http://<IP-address of Raspberry Pi>:1880`, which will get you into the Flow-Editor. The required modules are installed here first; in the Flow-Editor, this is done via the palette manager. The modules you will need are (some of them were already installed when installing Node-RED):

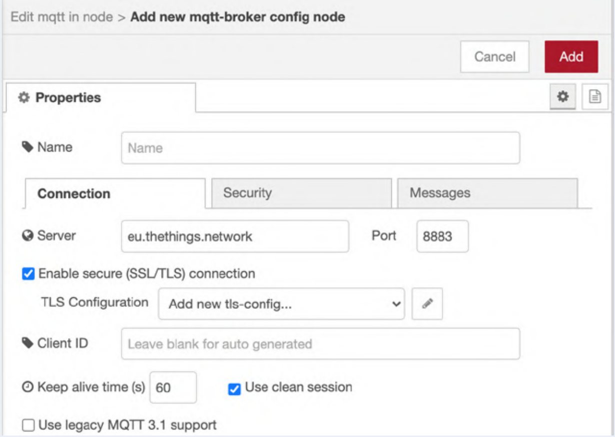
- *node-red*
- *node-red-contrib-worldmap*

Figure 1: MQTT-Node in Node-RED.



Server eu1.cloud.thethings.network:8883
Topic v3/+/devices/+/up/
QoS 2
Output a parsed JSON object
Name Name

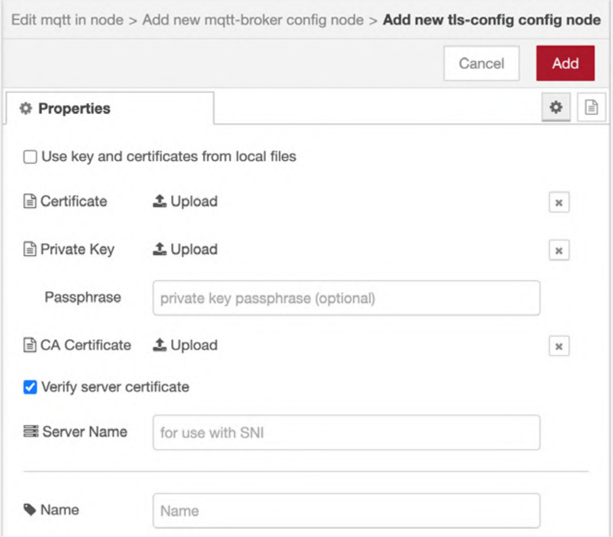
Figure 2: MQTT-node server property setting.



Edit mqtt in node > **Add new mqtt-broker config node**

Name Name
Server eu.thethings.network Port 8883
☑ Enable secure (SSL/TLS) connection
TLS Configuration Add new tls-config...
Client ID Leave blank for auto generated
Keep alive time (s) 60 ☑ Use clean session
☐ Use legacy MQTT 3.1 support

Figure 3: Add a new Server.



Edit mqtt in node > Add new mqtt-broker config node > **Add new tls-config config node**

☐ Use key and certificates from local files
Certificate Upload
Private Key Upload
Passphrase private key passphrase (optional)
CA Certificate Upload
☑ Verify server certificate
Server Name for use with SNI
Name Name

Figure 4: TLS configuration of the new Server.

- *node-red-node-rbe*
- *node-red-node-sqlite*
- *node-red-node-base64*
- *node-red-tail*

Some patience is called for here when installing via the Palette Manager; if the activity monitor has been hidden, it takes a while before the success message pops up at the top of the window.

If the installation via the Palette Manager is unsuccessful, exit the Editor again, and in the Terminal, stop the Node-RED service on the Raspberry Pi and do it manually. For example, the statement `npm i --unsafe-perm node-red-node-sqlite` installs the *node-red-node-sqlite* module, and you can do the same for the other modules.

If `node-red-node-sqlite` does not install the *node-red-sqlitedb*, it will need to be installed afterwards. This is only possible via npm, because in this case the Palette Manager detects a conflict with the *node-red-node-sqlite* module.

## The Missing TTN Module

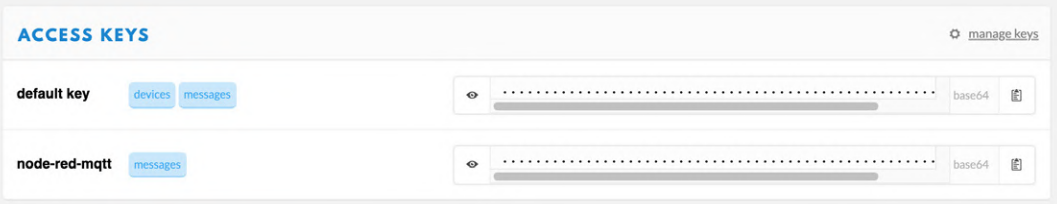
So far, so good. At this point, you may have noticed that the TTN-Module (The Things Network) discussed in the original article is missing. The reason is it is not supported by the Raspberry Pi environment. For example, once *node-red-contrib-ttn* has been loaded it is no longer possible to start Node-RED-Service on the Raspberry Pi. It took me a while to figure this out, but I'm glad now I can pass this on to save you the stress of that particular troubleshoot.

The solution lies in the *mqtt* module built into Node-RED. It is included in the origin flow instead of the TTN uplink module. You can download and use my flow, which is specially adapted for the Raspberry Pi from the LoRa GPS Tracker page [5] at Elektor Labs. The flow is imported by simply dragging the downloaded JSON file into the *Flow Editor* window. By the way, this flow adapted for the Raspberry Pi also runs on the PC in place of the flows with the special TTN nodes.

Using the *Flow Editor*, you can now double-click on the *mqtt* symbol (**Figure 1**) and edit the settings in the window that opens (**Figure 2**). Now "*v3/+/devices/+/up*" is entered under *Topic* as shown in the figure in order to receive the data of all devices in the TTN application. As *Output* you can keep the default configuraton "*a parsed JSON object*". A new server can be entered in the *Server* field by using the pencil symbol and the window shown in **Figure 3** opens.

The Application ID from the TTN Console is used as the *Name*, and *eu1.cloud.thethings.network* is the *Server*. The default port is *1833* (MQTT without TLS) which needs to be changed to *8883*. Lastly, what's missing is the checkmark next to *Enable secure (SSL/TLS) connection*. A click on the pencil symbol of the *TLS Configuration* field calls up the next window (**Figure 4**).

Now the settings can be accepted by clicking on *Add*. Back in the *mqtt-broker config node* window, select the *Security* tab (**Figure 5**).



ACCESS KEYS

manage keys

| | | |
|---|---|---|
| default key | devices messages | base64 |
| node-red-mqtt | messages | base64 |

Figure 5: Access key management of the MQTT nodes.

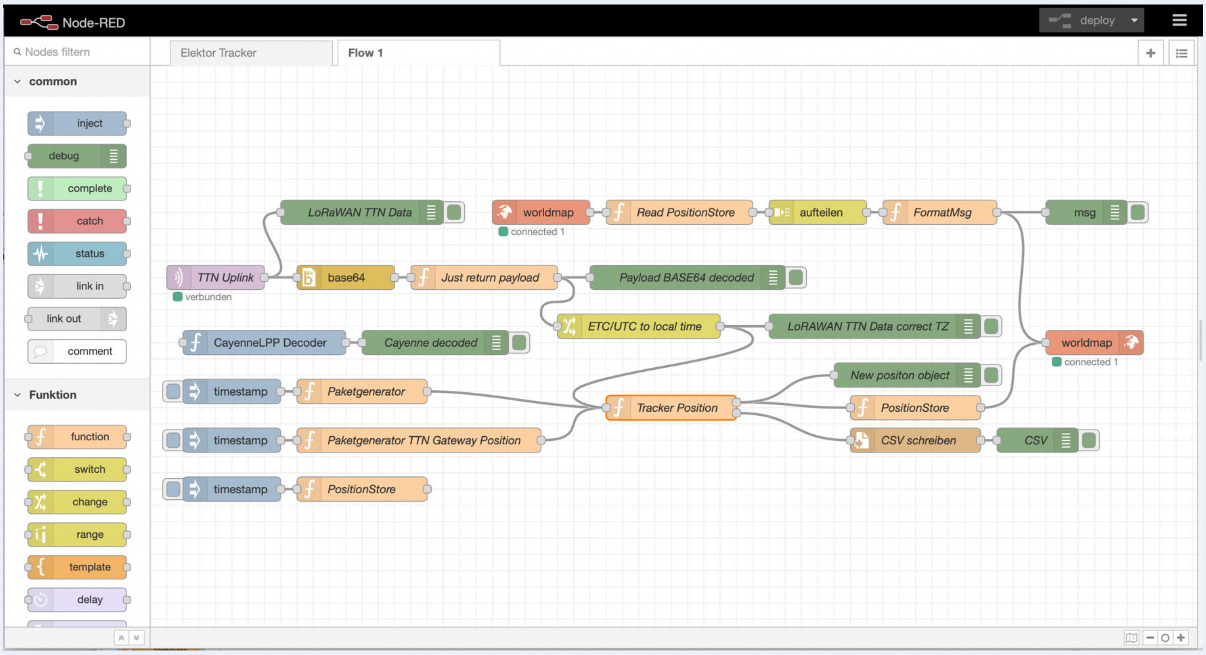


Figure 6: The finished flow diagram in Node-RED.

The *Username* is the Application ID from the TTN console with *@ttn* appended. An access key with at least "Messages Rights" must be generated as the password in the TTN console. The Access Key copied from the TTN console is then pasted under *Password* and confirmed by clicking on *Add*. Back in the *mqtt in* element, you can give the element a name (e.g. *TTN uplink*); lastly a click on *Done* completes the task.

**Figure 6** shows the updated Node-RED flow chart. The CayenneLPP decoder shown in the flow is included for future system expansion.

210120-01

## Contributors

Idea and Text: **Hans Schneider**
Editors: **Mathias Claußen, Jens Nickel, C.J. Abate**
Translation: **Martin Cooke**
Layout: **Harmen Heida**

## Questions or Comments?

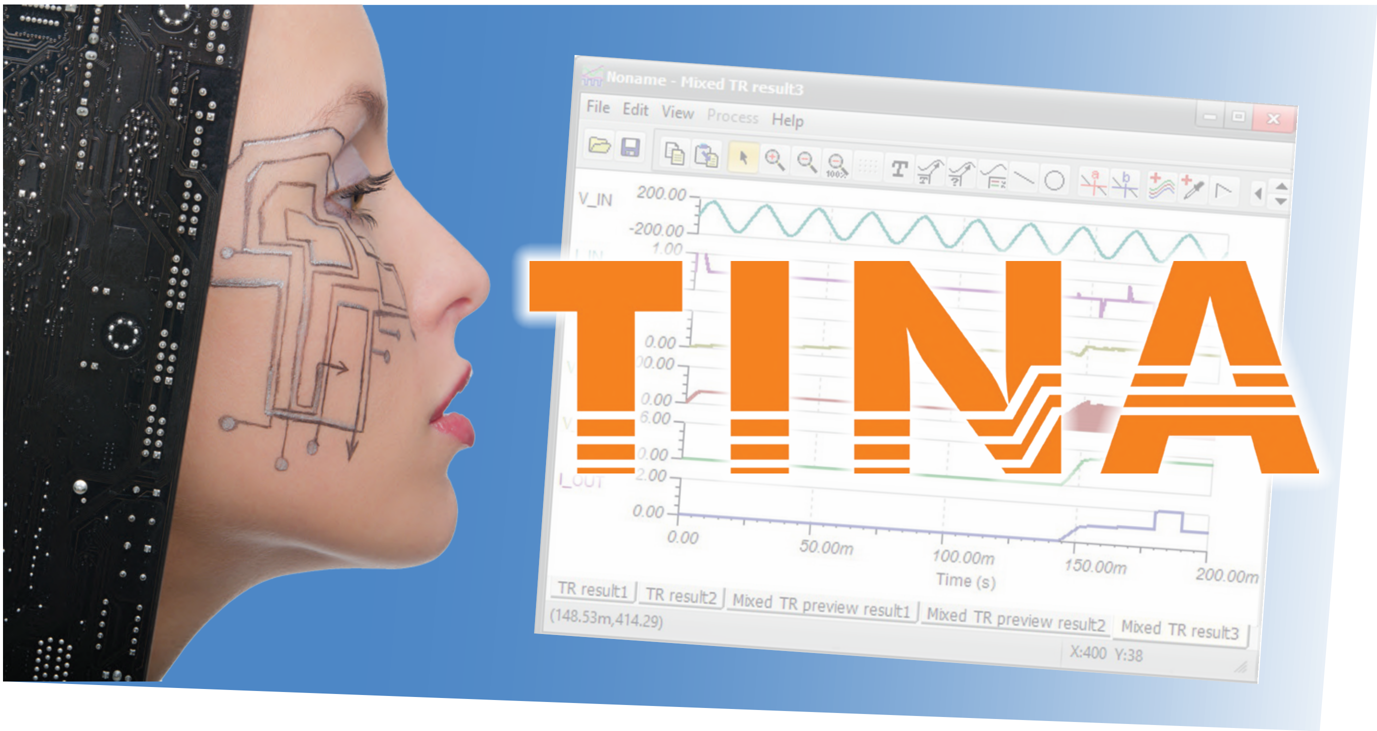
If you have any questions or comments regarding this article, you can contact the author at hans.schneider@belgacom.net or the team at Elektor at editor@elektor.com.

## RELATED PRODUCTS

- **Elektor LoRa Node – Partly Assembled Module (SKU 19175)**
  www.elektor.com/19175
- **SeedStudio RFM95 Ultra-long range LoRa Transceiver Module (868 MHz) (SKU 18715)**
  www.elektor.com/18715
- **Dragino PG1301 LoRaWAN GPS Concentrator for Raspberry Pi (868 MHz) (SKU 19367)**
  www.elektor.com/19367

## WEB LINKS

[1] M. Claußen, "LoRa GPS Tracker," ElektorMag 11-12/2020: www.elektormagazine.com/lora2020
[2] Pi-hole: https://pi-hole.net
[3] Homebridge: https://homebridge.io
[4] Flightradar24: www.flightradar24.com/
[5] "LoRa GPS Tracker," Elektor-Labs.com: www.elektormagazine.com/labs/lora-gps-tracker


Noname - Mixed TR result3
File Edit View Process Help
V_IN
200.00
-200.00
1.00
0.00
0.00
0.00
6.00
0.00
2.00
0.00
I_OUT
0.00
50.00m
100.00m
Time (s)
150.00m
200.00m
TR result1 | TR result2 | Mixed TR preview result1 | Mixed TR preview result2 | Mixed TR result3
(148.53m,414.29)
X:400 Y:38
TINA

# Circuit Simulation with TINA Design Suite & TINACloud

## Sample Chapter: Sinusoidal Oscillators

**By Dogan Ibrahim (United Kingdom)**

Reportedly, one of the best features of TINA is that a simulated circuit can easily be implemented on a PCB with auto-placement and auto-routing capabilities. Users can also use the Gerber plotting and CNC drilling options of TINA to learn, design, and implement a prototype of their projects. That's "awesome," but before any PCB can come into play, you have to master elementary circuit simulation. Here's an up-tempo primer to TINA you cannot afford to miss and do on your PC.

A sinusoidal oscillator consists of an amplifier and a feedback network (**Figure 1**). The following two conditions must be satisfied to have a working oscillator:

- the loop gain (A × B) in Figure 1 must be greater than or equal to unity;
- the total phase shift around the circuit must be 0 or 360°.

Some simulations of operational amplifier-based oscillator circuits are given in this section.

### Simulation 1 — Phase-shift Oscillator

This is sometimes called the RC oscillator. Each RC pair introduces a 60° phase shift. Three resistors and capacitors are used here to introduce a 180° phase shift to the feedback loop. The total loop phase shift is, therefore, 0° as required for oscillation.

### TINA schematic

**Figure 2** shows the circuit diagram. The RC network is connected to the inverting input of the operational amplifier. Assuming the resistors and capacitors are the same, the requirement is that the voltage gain of the amplifier must be greater than or equal to 29, i.e.:

$$Gain = \frac{R_f}{R} \geq 29$$

The frequency of oscillation is given by:

$$f = \frac{1}{2\pi RC\sqrt{6}}$$

*Editor's Note. This article is an excerpt from the 440-page book* Circuit Simulation with TINA Design Suite & TINACloud *(Elektor 2022). The excerpt was formatted and lightly edited to match Elektor Mag's editorial standards and page layout. Being an extract from a larger publication, some terms in this article may refer to discussions elsewhere in the book. 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.*

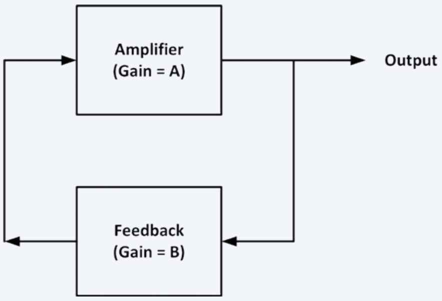


*Figure 1: Oscillator principle.*

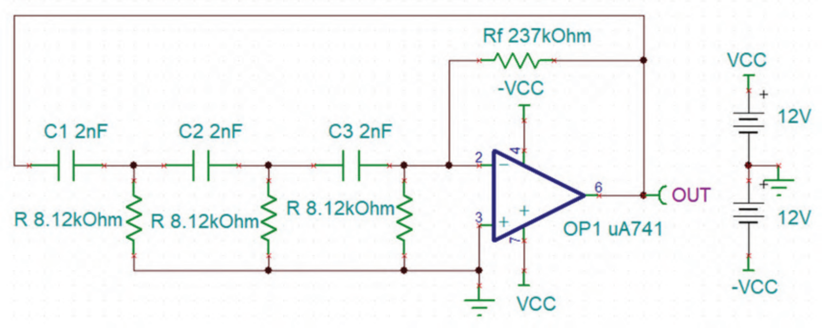


*Figure 2: Circuit schematic.*

In this example, the required frequency is 4 kHz. Choosing C = 2 nF, we can find the required value of R from:

$$R = \frac{1}{2\pi f C\sqrt{6}} = \frac{1}{2\pi \times 4 \times 10^{3} \times 2 \times 10^{-9} \times \sqrt{6}}$$

which gives R = 8.12 kΩ. Then,

$$\frac{Rf}{R} \geq 29$$

and we choose $R_f$ = 237 kΩ. A type UA741 operational amplifier is used in this project. The TINA circuit is available as a file *sim9* (see article end).

### TINA simulation

The steps to run the simulation are:

- Click *T&M* -> *Oscilloscope* and *Run*. Set the *Time/div* to *100 u*.
- Check the output waveform (**Figure 3**). The period is 250 µs which corresponds to 4 kHz. Note the glitch in the output waveform on the oscilloscope. The oscilloscope can be synchronized to create a stable picture. To do this, select *Normal* under *Mode* and *synchronizing signal* under *Source*. It may also be needed to set the (Trigger) Level.

## Simulation 2 — The Wien Bridge Oscillator

This is one of the simplest sinusoidal oscillators. The Wien Bridge oscillator is a two-stage RC coupled circuit that has good stability at its resonant frequency, low distortion, and is very easy to tune, thus making it a popular circuit as an audio frequency oscillator.

The circuit uses a series RC connected with a parallel RC. The phase shift of the circuit is 0° at the resonant frequency, and the circuit is connected to the positive input of the operational amplifier so that the overall phase shift is 0°. Usually, a non-inverting amplifier configuration is employed.

The condition for oscillation is that the voltage gain must be greater than or equal to 3.

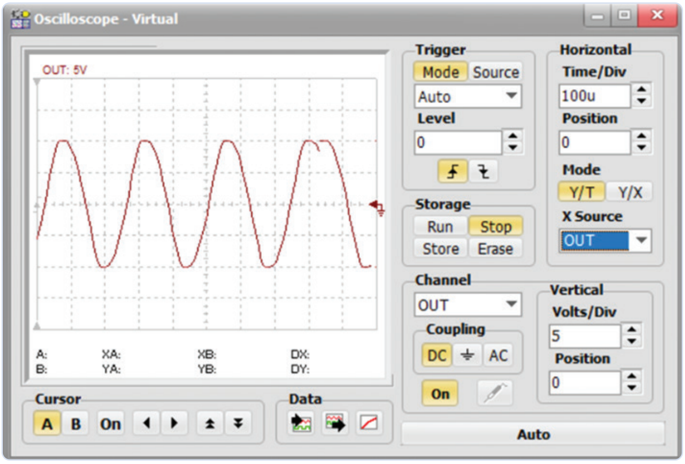
Assuming the same resistors and capacitors are used, the frequency of oscillation is given by:

$$f = \frac{1}{2\pi RC}$$

In this example, the required frequency is 5 kHz. Choosing C = 3 nF, we can find the required value of *R* from:

$$R = \frac{1}{2\pi f C} = \frac{1}{2\pi \times 5 \times 10^{3} \times 3 \times 10^{-9}}$$

which gives *R* = 10.6 kΩ.


Oscilloscope - Virtual

OUT: 5V

Trigger: Mode: Auto; Level: 0
Horizontal: Time/Div: 100u; Position: 0
Mode: Y/T
X Source: OUT
Channel: OUT
Coupling: DC
Vertical: Volts/Div: 5; Position: 0

*Figure 3: Output waveform.*

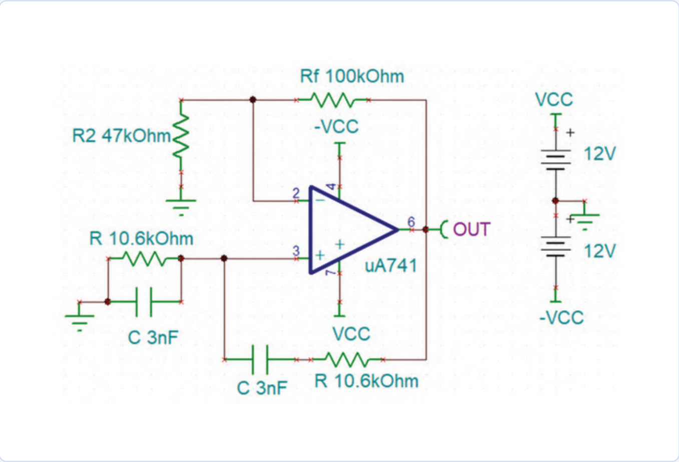
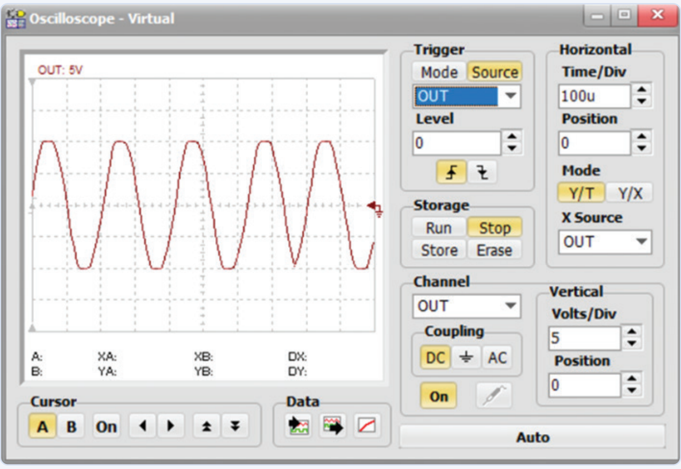


Figure 4: Circuit schematic.



Source: OUT
Level: 0
Channel: OUT
Time/Div: 100u
Position: 0
X Source: OUT
Volts/Div: 5
Position: 0

Figure 5: Output waveform.

To satisfy the gain condition, for a non-inverting amplifier (see **Figure 4**):

$$Gain = 1 + \frac{R_f}{R_2}$$

Choosing $R_f$ = 100 kΩ:

$$1 + \frac{R_f}{R_2} \geq 3$$

Which gives:

$$R_2 = \frac{R_f}{3-1} \leq 50K$$

Choose *R2* = 47 kΩ.

**TINA schematic**

Figure 4 shows the circuit diagram. The feedback circuit is connected to the noninverting input of the operational amplifier and the gain is set through $R_f$ and *R2*. A type UA741 operational amplifier is used in this project. The circuit is available as a file: *sim10*.

**TINA simulation**

The steps to run the simulation are:

- Click *T&M* -> *Oscilloscope* and click *Run*. Set the *Time/div* to *100u*.
- Check the output waveform (**Figure 5**). The period is 200 µs which corresponds to 5 kHz.

## Simulation 3 — The Colpitts Oscillator

The Colpitts oscillator uses a capacitive voltage divider network as its feedback source. Two capacitors, *C1* and *C2* are placed across a single common inductor, *L*, where *C1*, *C2*, and *L* form the tuned tank circuit. The feedback circuit is usually connected to the negative input. *C1*, *C2*, and *L* provide the additional 180° phase shift required to make the total phase shift 0°.

The frequency of oscillation is given by:

$$f = \frac{1}{2\pi\sqrt{LC_T}}$$

Where $C_T$ is the series combination of *C1* and *C2*, i.e.:

$$C_T = \frac{C_1 C_2}{C_1 + C_2}$$

The amount of feedback depends on the values of *C1* and *C2*. Therefore, by changing the values of these capacitors, we can adjust the amount of feedback voltage returned to the tank circuit. The ratio of *C1* to *C2* is the feedback ratio, *B*:

*B* = *C1* / *C2*

For oscillations, *AB* ≥ 1, where *A* is the amplifier gain.

Or, *A* ≥ *C2* / *C1*.

In this example, the required frequency is 10 kHz. Choosing *L* = 10 mH we can find the feedback ratio from:

$$C_T = \frac{1}{4\pi^2 f^2 L} = \frac{1}{4\pi^2 \times 10^8 \times 10 \times 10^{-3}}$$

which gives 25.3 nF.

Choosing *C2* = 250 nF gives *C1* = 28.15 nF, which corresponds to a feedback ratio of *B* = *C2* / *C1* = 250 / 28.15 = 8.88.

We can therefore choose the amplifier gain to be around 10 (see **Figure 6**), giving:

$$Gain = \frac{R_f}{R_1} = 10$$

Choosing *R1* = 5 kΩ gives $R_f$ = 50 kΩ.

**TINA schematic**

Figure 6 shows the circuit diagram. The feedback circuit is connected to the inverting input of the operational amplifier and the gain is set through $R_f$ and *R1*. A type UA741 operational amplifier is used in this project. The circuit is available as a file: *sim11*.

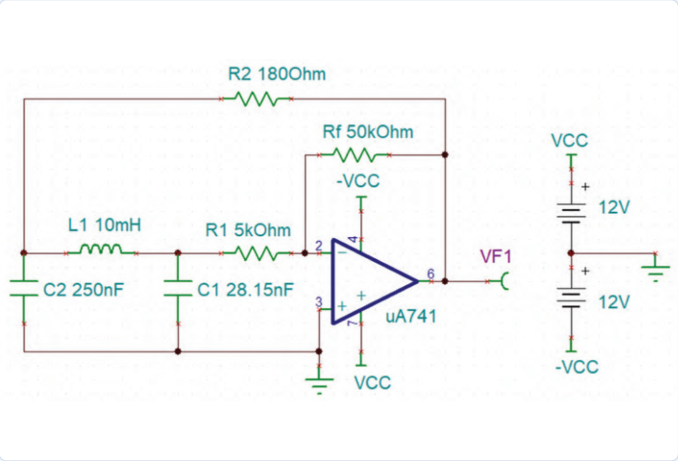
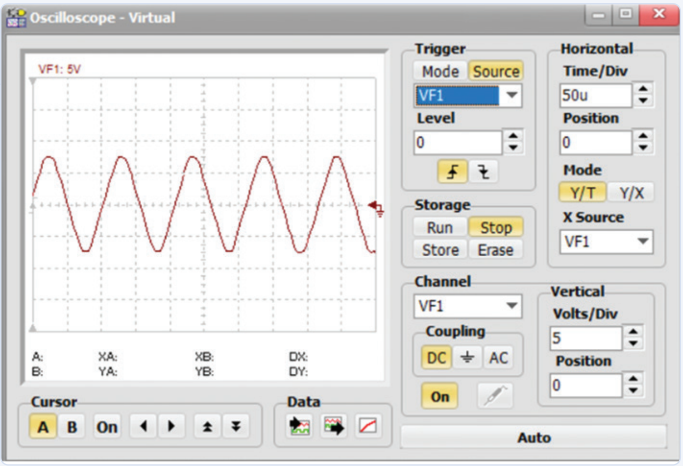


Figure 6: Circuit schematic.



Trigger Source: VF1
Level: 0
Channel: VF1
Horizontal Time/Div: 50u
Position: 0
X Source: VF1
Vertical Volts/Div: 5
Position: 0

Figure 7: Output waveform.

### TINA simulation

The steps to run the simulation are:

- Click *T&M* -> *Oscilloscope* and *Run*. Set the *Time/div* to 50u.
- Check the output waveform (**Figure 7**). The period is 100 µs (100u) which corresponds to 10 kHz.

## Where to get it?

The TINA simulation files mentioned in this article are contained in the software bundle released by the author and DesignSoft in support of the book. The software is available for free downloading. Head over to [1], scroll down to the *Downloads*, and click on this file name:
*Contents_Circuit Simulation with TINA Design Suite & TINACloud*.

Save the ZIP archive file locally (2.45 MB)) and then extract it. Open your version of TINA and load the files *sim9*, *sim10*, *sim11* as mentioned in this article. Feel free to modify them for your own applications.

**Limited Time Offer**: The TINA book comes with a free, one-year license of TINA Cloud Basic Edition.

220025-01

## Contributors

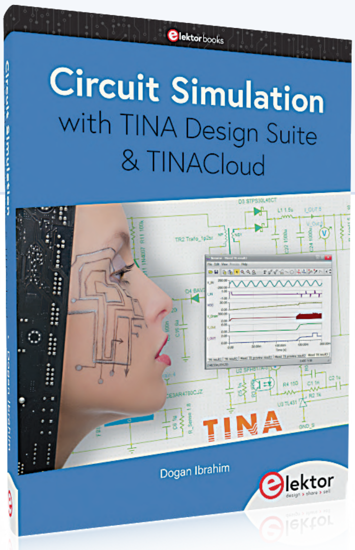
Text and Graphics: Dogan Ibrahim
Editor: Jan Buiting
Layout: Giel Dols

## Questions or Comments?

Do you have any technical questions or comments related to this article? Email the author at d.ibrahim@btinternet.com or Elektor at editor@elektor.com.

## RELATED PRODUCTS

- Book: D. Ibrahim, *Circuit Simulation with TINA Design Suite & TINACloud* (SKU 19977)
www.elektor.com/19977
- E-Book: D. Ibrahim, *Circuit Simulation with TINA Design Suite & TINACloud* (SKU 19978)
www.elektor.com/19978



**Both publications come with a FREE one-year License for TINACloud Basic Edition (Limited Offer).**

## WEB LINK

[1] Book resources/info page: www.elektor.com/circuit-simulation-with-tina-design-suite-tinacloud

## developer's zone

Tips & Tricks, Best Practices and Other Useful Information

# From Life's Experience

## Assembly Line Work

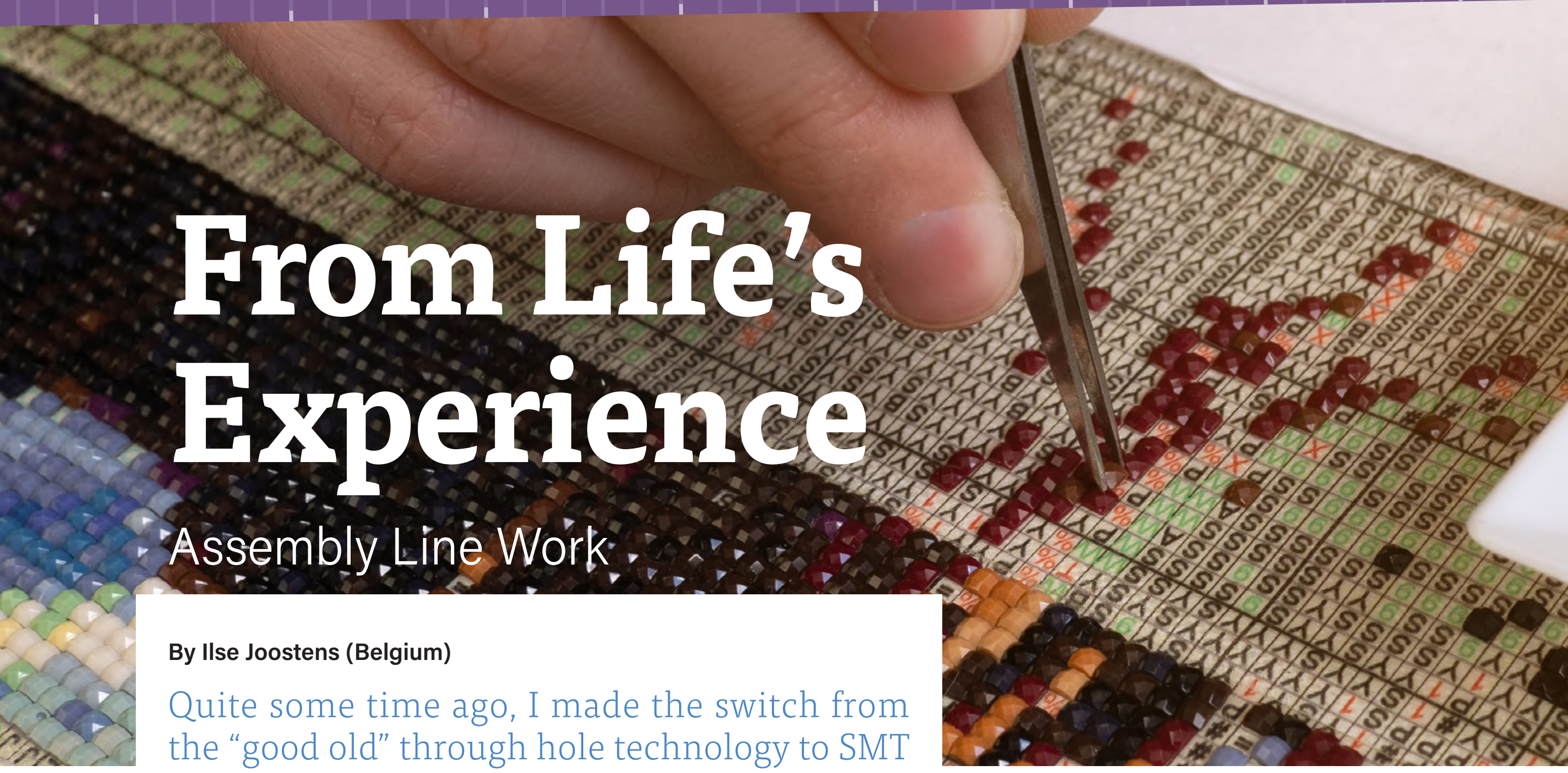


Photo: Yekatserina Netuk / Shutterstock.

By Ilse Joostens (Belgium)

Quite some time ago, I made the switch from the "good old" through hole technology to SMT components. Despite some initial trepidation, working with these horribly tiny parts turned out to be not as bad as imagined and soon I began dreaming about having my designs manufactured. The latter turned out to be a big disappointment however, and before I realised it, I spent entire days manually assembling circuit boards with tweezers, real drudgery therefore. It became so bad that even when I closed my eyes I continued to see the assembly line of SMT components moving along. And it will not surprise you that I had a slightly annoyed response when for my birthday I received a diamond painting, ostensibly for relaxation. According to the seller's website, diamond painting is stress-relieving and you can make master pieces that take your breath away. Maybe that is so, but to me it looks suspiciously similar to SMT assembly.

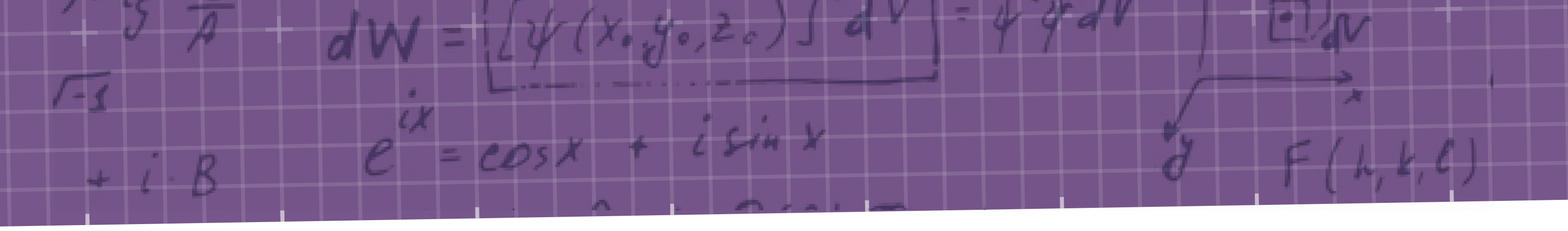
### Grit

A little over 10 years ago, I worked with a small German company and they had all their circuit boards assembled in China. If you are active in the electronics sector, you are likely familiar with the concepts of "grit" or "chicken feed". This is a euphemism for those cheap passive components that are normally mounted in large numbers on a circuit board, such as ceramic capacitors for the decoupling of the power supply rails. Unfortunately, the Chinese contract assembler took the term "grit" a little too literally. It appeared that the components were merely thrown on the board from a distance and many were on an angle or mounted the wrong way around, and then don't even mention the solder connections. Some components were even cracked, and it appeared they were pushed onto the circuit board with some considerable force. Now this hours-long rework of such circuit boards is not immediately the most productive activity and most certainly not when there is some time pressure because customers are waiting for it.

The circuit boards themselves were also of questionable quality. It was almost impossible to desolder components without lifting the copper pads as well. Broken traces and inadvertent short circuits between traces also made their appearance every now and then. When I became truly fed up with this and conveyed this to my German supplier, he shrugged his shoulders and said nonchalantly: "Chinese quality!" For my own products, I didn't want to fall into that trap and decided to go with European suppliers. The quality was perfectly fine, but now instead of the familiar smoke signals, there were other dark clouds along the horizon.

### Chocolate Cake

I normally ordered my circuit boards for prototypes from a European PCB manufacturer, because the wait times would otherwise be too long. Even with larger production quantities the higher price was still acceptable,



because the cost of the circuit board is only a small part of the total cost of a product. It is of course important to avoid multi-layer circuit boards and solder masks in fancy colours as much as possible. But after a while that green solder mask looks a bit boring, and I was secretly a little envious of the circuit boards in fancy colours from the competition.

SMT assembly on the other hand was a big disappointment, with the high set-up costs being an absolute killer. Just the manufacturing of the stencils for the solder paste printer already ran into the hundreds of euros, and that is not counting the documentations costs, the set-up costs for the SMT assembly machine, and the Automated Optical Inspection (AOI) as well.

Unfortunately, my volumes were such that it was too much to assemble everything myself by hand and too few to cover the costs of automated assembly. And so many a lonely evening in the attic room was my lot. Besides the repeated looking for the required parts, it was the application of solder paste to the circuit board that was the most cumbersome task. To make it easier for myself, I normally cut a window in a piece of 3-mm thick perspex that fits the circuit board exactly. With two circuit boards stacked on top of one another, the top board is pretty much flush with the perspex and the stencil can be positioned in the correct location and secured in place with some tape. Only the solder paste is a mess to work with every time. It doesn't matter how carefully you work, but after a number of circuit boards, the table and your hands are covered in solder paste and with a bit of bad luck it is also on your clothes. You know the classic image of a baby who has just eaten a piece of chocolate cake, where the baby as well as the high chair are covered in chocolate [1]. After the application of the solder paste, a similar feeling overcomes me and with a big sigh start with the cleanup.

## Geopolitics

These days we normally order our circuit boards from the Far East, mainly because of some pressure from our customers to keep prices down. And yes, it goes well, although the quality is perhaps a little lower, but in any case certainly acceptable, and we can finally choose from a range of fancy colours without an immediate financial penalty.

In the past few years, we have, despite everything, been able to have some products assembled, including the sand clock and the Swiss Pi [2] expansion board for the Raspberry Pi. For a while now, the assembly quality in China has improved immensely, despite the democratically low prices and ditto set-up costs. Other services too, such as 3D printing and CNC machining of various materials are offered, so that it becomes very tempting to place orders there.

Nevertheless, it doesn't feel right, because apart from the pricing, there is very little that is democratic in China. Their worldwide expansion drive and geopolitical developments in recent years make them just a little less sympathetic. So, I will store my tweezers not too far away. ⏮

220031-01

## Questions or Comments?

Do you have any technical questions or comments prompted by this article? Send an email to the editor of Elektor via editor@elektor.com.

## Contributors

Text: **Ilse Joostens**
Editor: **Eric Bogers**
Translation: **Arthur de Beun**
Layout: **Giel Dols**

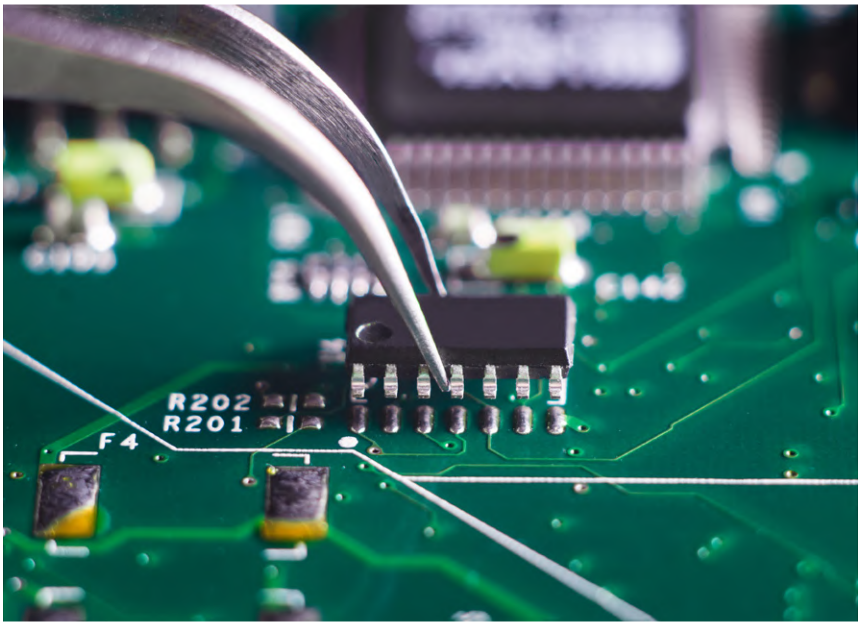


*Photo: Oleg Shvydiuk / Shutterstock.*

## WEB LINKS

[1] "Is cake smashing the dumbest cake trend ever?", Anges de Sucre/Anges Bakery: https://bit.ly/3nyQeGD
[2] P. S'heeren, "Swiss Pi – A Swiss Army Knife for the Raspberry Pi," Elektor 09-10/2016: https://www.elektormagazine.com/magazine/elektor-201609/39796

# The WinUI Graphics Framework for Windows Apps

## A Small Demo Application

By **Dr. Veikko Krypczyk** (Germany)

Software designed to control various electronic applications often runs in a Windows environment. Using a locally installed desktop application gives you direct access to all of the PC's system interfaces. Microsoft is currently rationalizing its support for developers working with Windows interface frameworks. The new graphics interface WinUI 3 is the way forward. Here, we will take a look at the technical background and show how to use it by building a small demo app for electronics technicians.

For many electronics projects, a Windows PC is used to provide control functions, data logging and other tasks; it will typically be necessary to develop a graphical user interface to run under Windows to carry out this function. Locally executed desktop applications are often the method of choice here; they give you complete access to the system environment and, via the appropriate drivers, also provide links to connected peripherals.

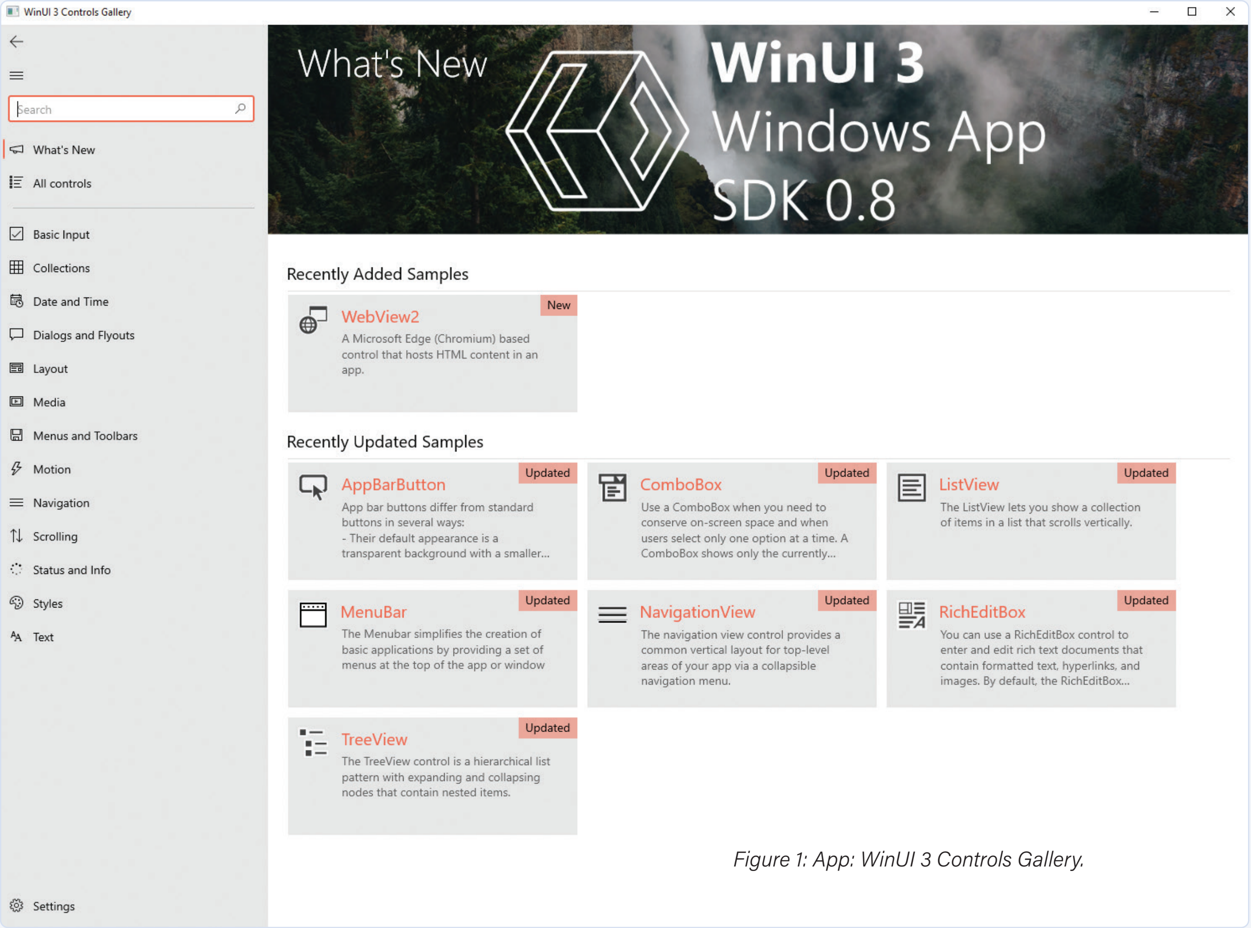
Microsoft, as the manufacturer of the Windows operating system, is currently undergoing a major technological upheaval. The focus of this activity is on the connectivity and design of user interfaces. The developments are coordinated under the project names *WinUI 3* and *Windows App SDK*. In this article, we show what this new graphic interface is all about and where we can use it. First, we get an overview of the possibilities of programming Windows applications with a graphical user interface. The purpose of WinUI 3 will become clear by using different technologies and application types. Not content with a purely theoretical review of this relatively new type of Windows application, we will also go ahead and create our first practical application.

### The Technology

Basically, current applications using a graphical user interface for the Windows operating system can broadly be divided into two types. On the one hand, we have *Desktop Applications*. These are essentially based on the use of the *Win32 API*. There are different approaches, frameworks and programming languages for their development. The technologies Windows Forms (WinForms) and Windows Presentation Foundation (WPF) come from Microsoft. WinForms relies on the Windows GDI interface. WPF is based internally on DirectX and was originally intended as a replacement for WinForms. Both graphics frameworks were based on the .NET framework, which was intended for Windows programs and whose further development ended at version 4.8.

Version .NET 5, on the other hand, is the technological successor to .NET Core. This framework is not limited to Windows, but can also be used with other operating systems. Microsoft has surprisingly transformed both WinForms and WPF to .NET Core. If you are creating an app for Windows today and opt for WinForms or WPF, then you have the choice between the previous .NET framework and .NET Core. It is also possible to migrate existing applications, but, as is so often the case with such projects, it is often associated with a number of issues. Other manufacturers of development tools for Windows applications have mostly based them on the graphic interface of the operating system (GDI) and encapsulated this in their own framework.

The second category of Windows applications are apps for the *Universal Windows Platform (UWP)*. These run in an individually isolated area of the operating system and have only limited system access. Users install these apps through the store. In practice, however, this type of app has not proved to be popular and its uptake is quite low. One of the reasons for this is that system access from these apps is very limited. The UWP however has the advantage that the graphics framework WinUI 2 used here is significantly more modern than the technologies of WinForms and WPF. An appealing design, new visual components, the use of materials and the orientation towards the design language *Fluent Design System* are its stand out features. In other words, apps for the UWP look modern, contemporary and fresh, but their usability is somewhat limited. To achieve similar effects with the WinForms or WPF technologies, we need to do some pull-ups, use extensive third-



WinUI 3 Controls Gallery

Search

What's New

All controls

Basic Input

Collections

Date and Time

Dialogs and Flyouts

Layout

Media

Menus and Toolbars

Motion

Navigation

Scrolling

Status and Info

Styles

Text

Settings

What's New

WinUI 3
Windows App
SDK 0.8

Recently Added Samples

WebView2 — New
A Microsoft Edge (Chromium) based control that hosts HTML content in an app.

Recently Updated Samples

AppBarButton — Updated
App bar buttons differ from standard buttons in several ways:
- Their default appearance is a transparent background with a smaller...

ComboBox — Updated
Use a ComboBox when you need to conserve on-screen space and when users select only one option at a time. A ComboBox shows only the currently...

ListView — Updated
The ListView lets you show a collection of items in a list that scrolls vertically.

MenuBar — Updated
The Menubar simplifies the creation of basic applications by providing a set of menus at the top of the app or window

NavigationView — Updated
The navigation view control provides a common vertical layout for top-level areas of your app via a collapsible navigation menu.

RichEditBox — Updated
You can use a RichEditBox control to enter and edit rich text documents that contain formatted text, hyperlinks, and images. By default, the RichEditBox...

TreeView — Updated
The TreeView control is a hierarchical list pattern with expanding and collapsing nodes that contain nested items.

*Figure 1: App: WinUI 3 Controls Gallery.*

party components or “mix” the WinForms/WPF technologies with the UWP. This approach however quickly leads to a more complex app structure and brings with it the typical disadvantages, such as higher susceptibility to errors and poorer maintainability.

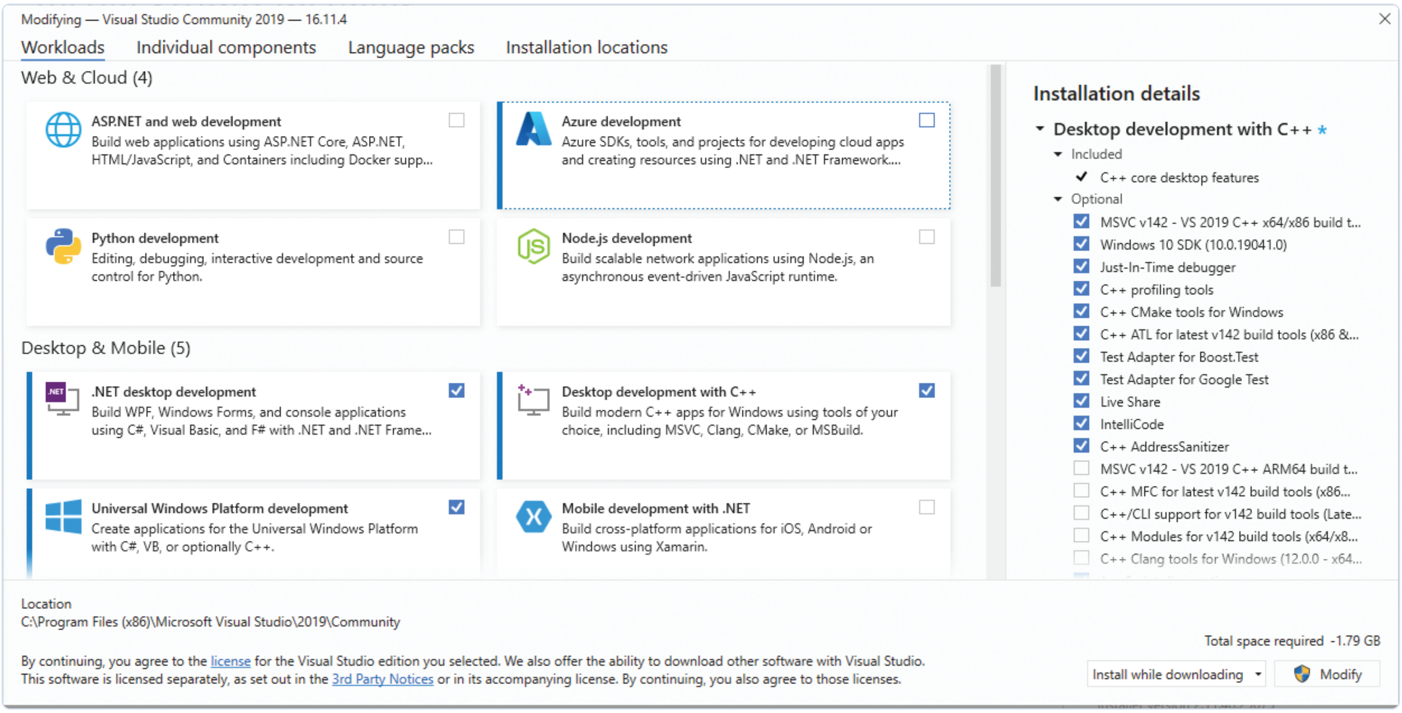
Software designed for electronic control, development, etc., are almost without exception classic desktop applications. You can also build these using other tools and frameworks. For the programming language Java there is, for example, the graphics framework *Swing*, which for Windows is based internally on the GDI operating system interface.

## The WinUI 3 Graphic Framework

With the introduction of the WinUI 3 graphic interface, Microsoft would like to enable all applications under Windows to use a modern graphic interface. WinUI 3 is the technological successor to WinUI 2 [1]. However, it is available for all types of Windows applications and is therefore not limited to use in apps for the UWP. WinUI 3 is part of the new Windows App SDK, which is also provided in parallel with the introduction of Windows 11. The Windows APP SDK bundles new features for the development of Windows applications. It is not only aimed at Windows 11, but can also be used under the current versions of Windows 10. Development of the Windows App SDK is still ongoing, but a first version is available that can already be used in newly created applications.

WinUI 3 is technically and conceptually based on WinUI 2. If you have ever developed an app for the UWP, you will quickly get to grips with it. It is based on the following principles:

- *Separation of code and design:* The user interface is declaratively created in separate files using the XML-based XAML language.
- *UI control elements*: A range of controls are available for designing the user interface. These include basic elements such as buttons and text entry fields, along with more complex and advanced elements such as a calendar control element, a WebView or an element for displaying personal data, which we can use, for example, for user administration. If you would like to explore the range of control elements further go to the Microsoft Store and download the *WinUI 3 Controls Gallery* app from there. This app gives a preview of the available controls for WinUI 3. Their use (Use Case) and their integration in the source code (XAML) are demonstrated. Corresponding links to the documentation can also be found (**Figure 1**).
- *Loose coupling through DataBinding*: The properties and events of the control elements are linked to the source code by means of data binding. In this way, data is exchanged in both directions between the program code and the user interface control. Events of the control elements, such as the click on a button, are also forwarded to the relevant algorithm in the same way.
- *Modern Design*: The WinUI 3 provides a contemporary feel. This includes the use of Microsoft’s design language *Fluent Design System* with the *Mica* material introduced in Windows 11. The Fluent Design System provides the following UI elements: conscious use of geometry and colour, overlapping of surfaces, use of selected materials and the use of specific iconography and typography for visual design using images, symbols and fonts. Motions between UI elements are also supported.



Modifying — Visual Studio Community 2019 — 16.11.4

Workloads | Individual components | Language packs | Installation locations

Web & Cloud (4)

ASP.NET and web development ☐
Build web applications using ASP.NET Core, ASP.NET, HTML/JavaScript, and Containers including Docker supp...

Azure development ☐
Azure SDKs, tools, and projects for developing cloud apps and creating resources using .NET and .NET Framework....

Python development ☐
Editing, debugging, interactive development and source control for Python.

Node.js development ☐
Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

Desktop & Mobile (5)

.NET desktop development ☑
Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F# with .NET and .NET Frame...

Desktop development with C++ ☑
Build modern C++ apps for Windows using tools of your choice, including MSVC, Clang, CMake, or MSBuild.

Universal Windows Platform development ☑
Create applications for the Universal Windows Platform with C#, VB, or optionally C++.

Mobile development with .NET ☐
Build cross-platform applications for iOS, Android or Windows using Xamarin.

Installation details

- Desktop development with C++ *
  - Included
    - ✓ C++ core desktop features
  - Optional
    - ☑ MSVC v142 - VS 2019 C++ x64/x86 build t...
    - ☑ Windows 10 SDK (10.0.19041.0)
    - ☑ Just-In-Time debugger
    - ☑ C++ profiling tools
    - ☑ C++ CMake tools for Windows
    - ☑ C++ ATL for latest v142 build tools (x86 &...
    - ☑ Test Adapter for Boost.Test
    - ☑ Test Adapter for Google Test
    - ☑ Live Share
    - ☑ IntelliCode
    - ☑ C++ AddressSanitizer
    - ☐ MSVC v142 - VS 2019 C++ ARM64 build t...
    - ☐ C++ MFC for latest v142 build tools (x86...
    - ☐ C++/CLI support for v142 build tools (Late...
    - ☐ C++ Modules for v142 build tools (x64/x8...
    - ☐ C++ Clang tools for Windows (12.0.0 - x64...

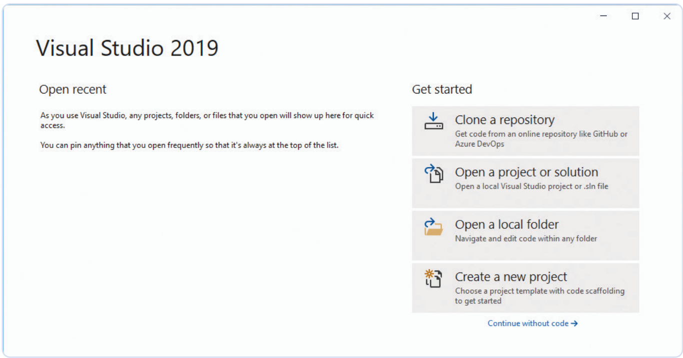
Location
C:\Program Files (x86)\Microsoft Visual Studio\2019\Community

By continuing, you agree to the license for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the 3rd Party Notices or in its accompanying license. By continuing, you also agree to those licenses.

Total space required -1.79 GB

Install while downloading | Modify

*Figure 2: Select the necessary Workloads for Visual Studio.*



Visual Studio 2019

Open recent

As you use Visual Studio, any projects, folders, or files that you open will show up here for quick access.

You can pin anything that you open frequently so that it's always at the top of the list.

Get started

Clone a repository
Get code from an online repository like GitHub or Azure DevOps

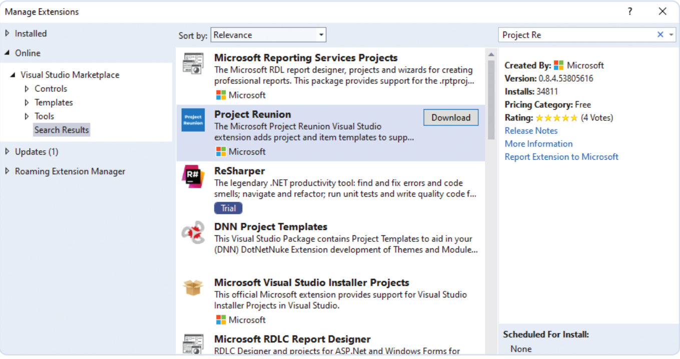
Open a project or solution
Open a local Visual Studio project or .sln file

Open a local folder
Navigate and edit code within any folder

Create a new project
Choose a project template with code scaffolding to get started

Continue without code →

*Figure 3: First Start of Visual Studio.*



Manage Extensions

- Installed
- Online
  - Visual Studio Marketplace
    - Controls
    - Templates
    - Tools
    - Search Results
- Updates (1)
- Roaming Extension Manager

Sort by: Relevance

**Microsoft Reporting Services Projects**
The Microsoft RDL report designer, projects and wizards for creating professional reports. This package provides support for the .rptproj...
Microsoft

**Project Reunion** Download
The Microsoft Project Reunion Visual Studio extension adds project and item templates to supp...
Microsoft

**ReSharper**
The legendary .NET productivity tool: find and fix errors and code smells; navigate and refactor; run unit tests and write quality code f...
Trial

**DNN Project Templates**
This Visual Studio Package contains Project Templates to aid in your (DNN) DotNetNuke Extension development of Themes and Module...

**Microsoft Visual Studio Installer Projects**
This official Microsoft extension provides support for Visual Studio Installer Projects in Visual Studio.
Microsoft

**Microsoft RDLC Report Designer**
RDLC Designer and projects for ASP.Net and Windows Forms for

Project Re

**Created By:** Microsoft
**Version:** 0.8.4.53805616
**Installs:** 34811
**Pricing Category:** Free
**Rating:** ★★★★★ (4 Votes)
Release Notes
More Information
Report Extension to Microsoft

**Scheduled For Install:**
None

*Figure 4: Windows SDK App (Project "Reunion") installation.*

Let's look at the procedure for developing applications with the WinUI 3. First, we will need to set up the development environment.

## The Development Environment and Setup

The current version of Visual Studio 2019 [2] will be used as the development environment. The Community Edition will be sufficient for our needs. By the time you're reading this article, a stable version of Visual Studio 2022 may already be available, in which case you should use this version. It is advisable to install the latest updates for the operating system beforehand. During the installation of Visual Studio, you will be asked to select the installation packages. You can also call up the *Visual Studio Installer* at any point later on via the start menu. Now select the following installation packages (Workloads): *.NET desktop development, Desktop development with C ++* and *Universal Windows Platform development* (**Figure 2**). After installation, start Visual Studio and in the start screen select the option *Continue without code ->* (**Figure 3**).

We now also need to install the template for development with WinUI 3. To do this in Visual Studio, choose the menu option *Extensions | Manage Extensions*. Search here for *Project Reunion* (the development name of the new Windows App SDK) and install the current version (**Figure 4**). In the same way install the *Windows Template Studio* extension. This offers advanced templates for creating a new application. Visual Studio must be restarted after the extensions have been downloaded; the installation will then take place automatically.

## An App for WinUI 3

Let's start by creating a new project. Here in **Figure 5** we select the App *WinUI 3 in Desktop* template (*Windows Template Studio*). In the Windows Template Studio (**Figure 6**), we can configure the project:

- *Project type*: Here we specify the type of navigation, for example with a menu bar or a side navigation bar (Hamburger menu).
- *Design pattern*: Direct installation and configuration of the MVVM toolkit. This couples the elements of the user interface (defined in XAML) with the program logic (programming language C#).
- *Pages*: We can add a number of pages to the project. We can choose from different templates, for example one page for entering program settings.
- *Features*: Here we can select some sample themes or save program settings.

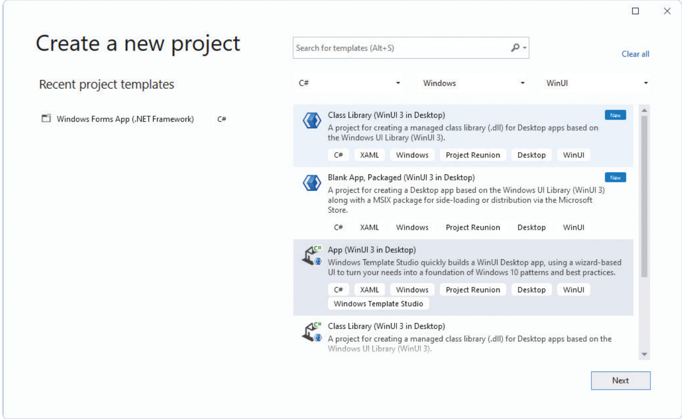
By clicking on the *Create* button, we generate the desktop application that WinUI 3 uses. Windows Template Studio generates a project folder with three projects:

- *App*: This contains the source code for the desktop application. In the subfolder *View*, for example, you will find the XAML files for the pages that were created by Windows Template Studio. The program logic is stored in files in the *ViewModel* program folder.
- *Package*: The project is responsible for providing the desktop application. At the moment, WinUI 3 applications are installed on the target computer as an app package. This format has so far been used for UWP apps. The generated packages can also be distributed by using Store. Future versions of the Windows App SDK should also allow installation without an app package.
- *Core*: This project contains the collection of services and classes that provide service for the app. This project is not mandatory and can be left out.

Start the application directly from Visual Studio using the green arrow on the toolbar. Congratulations - You have created your first application with WinUI 3 (**Figure 7**). I should emphasis again; this is a desktop application with full system access. As already mentioned, this is important for software designed to control external electronics, for example. A side navigation bar, first pages and the possibility to adapt the application design are all possible. You have all options for accessing the system, including communication with the system libraries and drivers. Now we can go ahead and experiment with the design of the user interface.

## A Demo Application

The best way to become familiar with any new system is to try it out. Here we will design a simple user interface for our first application (the source code, for this example, can be found on the web page for the article [3]). The starting point is the XAML file for the relevant page. As an experiment we can create a handy calculation tool for use with the LM317 adjustable linear voltage regulator (**Figure 8**). The output voltage of this device is given by the formula *Vout = 1.25 (1 + R2/R1)*. We can solve this equation for *R2* and thus calculate its value to give the desired output voltage. With the help of this example we can



Create a new project

Search for templates (Alt+S)

Clear all

Recent project templates

Windows Forms App (.NET Framework) C#

C# | Windows | WinUI

Class Library (WinUI 3 in Desktop) New
A project for creating a managed class library (.dll) for Desktop apps based on the Windows UI Library (WinUI 3).
C# XAML Windows Project Reunion Desktop WinUI

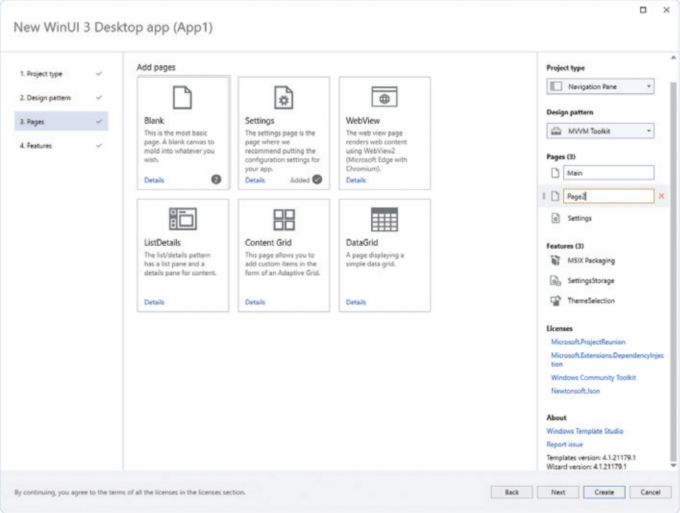
Blank App, Packaged (WinUI 3 in Desktop) New
A project for creating a Desktop app based on the Windows UI Library (WinUI 3) along with a MSIX package for side-loading or distribution via the Microsoft Store.
C# XAML Windows Project Reunion Desktop WinUI

App (WinUI 3 in Desktop)
Windows Template Studio quickly builds a WinUI Desktop app, using a wizard-based UI to turn your needs into a foundation of Windows 10 patterns and best practices.
C# XAML Windows Project Reunion Desktop WinUI Windows Template Studio

Class Library (WinUI 3 in Desktop)
A project for creating a managed class library (.dll) for Desktop apps based on the Windows UI Library (WinUI 3).

Next

Figure 5: Project template WinUI 3 Desktop.



New WinUI 3 Desktop app (App1)

1. Project type ✓
2. Design pattern ✓
3. Pages ✓
4. Features ✓

Add pages

Blank
This is the most basic page. A blank canvas to mold into whatever you wish.
Details 2

Settings
The settings page is the page where we recommend putting the configuration settings for your app.
Details Added ✓

WebView
The web view page renders web content using WebView2 (Microsoft Edge with Chromium).
Details

ListDetails
The list/details pattern has a list pane and a details pane for content.
Details

Content Grid
This page allows you to add custom items in the form of an Adaptive Grid.
Details

DataGrid
A page displaying a simple data grid.
Details

Project type
Navigation Pane

Design pattern
MVVM Toolkit

Pages (3)
Main
Page2
Settings

Features (3)
MSIX Packaging
SettingsStorage
ThemeSelection

Licenses
Microsoft.ProjectReunion
Microsoft.Extensions.DependencyInjection
Windows Community Toolkit
Newtonsoft.Json

About
Windows Template Studio
Report issue
Templates version: 4.1.21179.1
Wizard version: 4.1.21179.1

By continuing, you agree to the terms of all the licenses in the licenses section.

Back Next Create Cancel

Figure 6: Windows Template Studio.

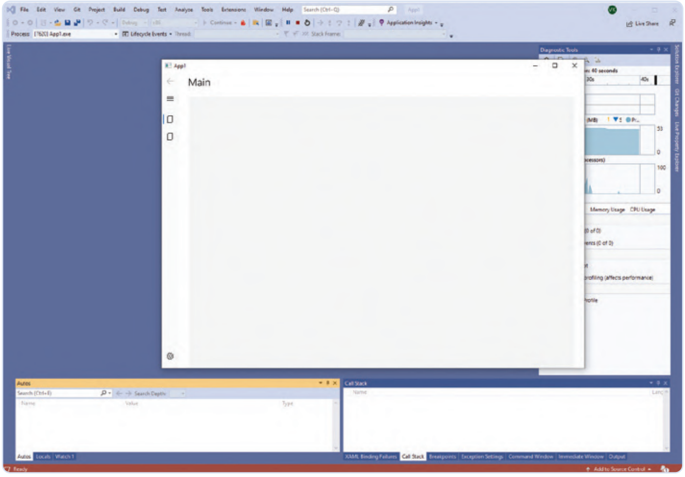


Figure 7: A first desktop application with WinUI 3.

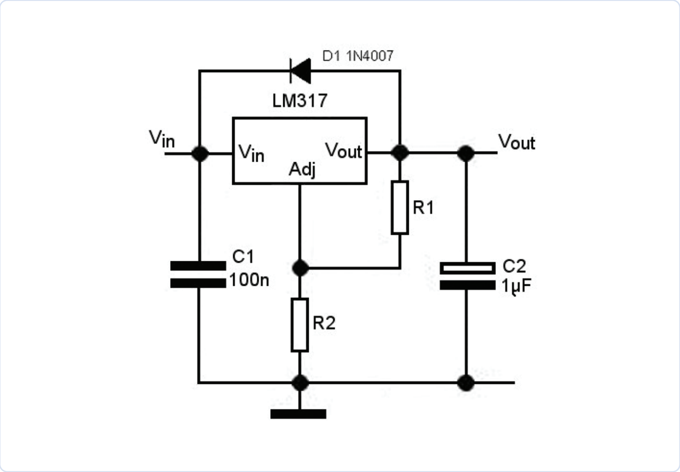


*Figure 8: LM317 voltage regulator circuit diagram.*

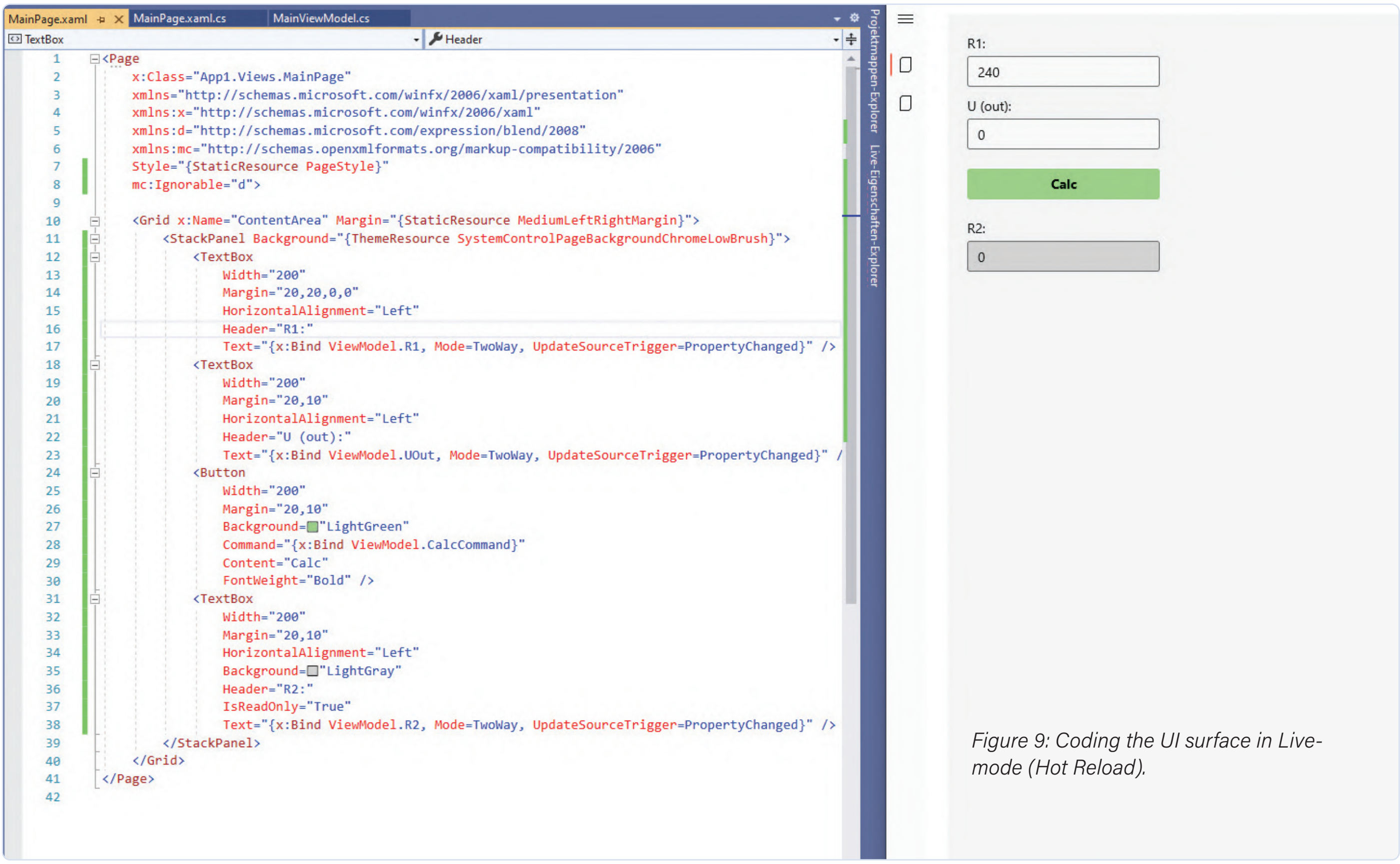
demonstrate the procedure for programming applications with WinUI 3. This process will include the following steps:

- Definition of the User Interface in XAML.
- Coding the Program logic in C#.
- Binding the user interface to the program logic.
- Forwarding the user interaction from the user interface to the program logic.
- Output the data to the form.

Let's start by defining the surface. We need two text fields to record the values of *R1* and $V_{Out}$. We also need a text field for the value of *R2*. A *Button* will be required to trigger the calculation. We use *TextBox* controls for input and output. All the elements should be arranged in the form of a vertical stack one above the other, which is why they are inserted into a layout container of the type *<StackPanel />*. Without

**Table 1: Control Elements**

| Control Element | Property | Value | Description |
|---|---|---|---|
| TextBoxR1 | Width | 200 | Textbox width |
| | Margin | 20, 20, 0, 0 | Margin size: left, top, right, bottom |
| | HorizontalAlignment | Left | Left justified |
| | Header | R1: | Caption |
| | Text | x:Bind ViewModel.R1 | Binding the property to the variable R1 in C#. |
| TextBoxU (out) | Width | 200 | Textbox width |
| | Margin | 20, 10 | Margin size: left, top, right, bottom |
| | HorizontalAlignment | Left | Left justified |
| | Header | U (out): | Caption |
| | Text | x:Bind ViewModel.UOut | Binding the property to the variable UOut in C#. |
| TextBoxR2 | Width | 200 | Textbox width |
| | Background | LightGray | Background colour |
| | Margin | 20, 10 | Margin size: left, top, right, bottom |
| | HorizontalAlignment | Left | Left justified |
| | Header | R2: | Caption |
| | IsReadOnly | True | Read only protection |
| | Text | x:Bind ViewModel.R2 | Binding the property to the variable R2 in C#. |
| Button | Width | 200 | Button area width |
| | Margin | 20, 10 | Margin size: left, top, right, bottom |
| | Background | LightGreen | Background colour |
| | Command | x:Bind ViewModel.CalcCommand | Binding to the CalcCommand Method in C#. |
| | Content | Calc | Caption |
| | FontWeight | Bold | Font properties |



```
<Page
    x:Class="App1.Views.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    Style="{StaticResource PageStyle}"
    mc:Ignorable="d">

    <Grid x:Name="ContentArea" Margin="{StaticResource MediumLeftRightMargin}">
        <StackPanel Background="{ThemeResource SystemControlPageBackgroundChromeLowBrush}">
            <TextBox
                Width="200"
                Margin="20,20,0,0"
                HorizontalAlignment="Left"
                Header="R1:"
                Text="{x:Bind ViewModel.R1, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}" />
            <TextBox
                Width="200"
                Margin="20,10"
                HorizontalAlignment="Left"
                Header="U (out):"
                Text="{x:Bind ViewModel.UOut, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}" /
            <Button
                Width="200"
                Margin="20,10"
                Background=🟩"LightGreen"
                Command="{x:Bind ViewModel.CalcCommand}"
                Content="Calc"
                FontWeight="Bold" />
            <TextBox
                Width="200"
                Margin="20,10"
                HorizontalAlignment="Left"
                Background=⬜"LightGray"
                Header="R2:"
                IsReadOnly="True"
                Text="{x:Bind ViewModel.R2, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}" />
        </StackPanel>
    </Grid>
</Page>
```

R1:
240
U (out):
0
R2:
0

Figure 9: Coding the UI surface in Live-mode (Hot Reload).

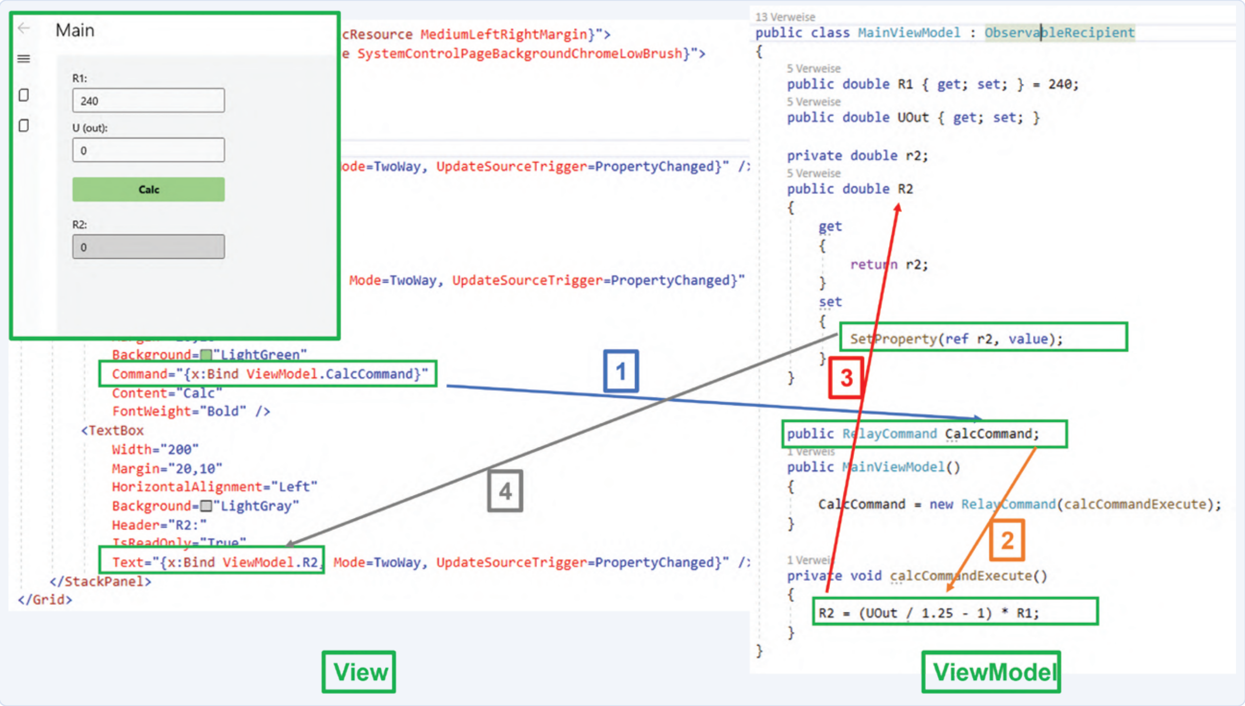
the need for any further configuration, all the elements will be arranged one above the other with *StackPanel*. The controls are configured via the XAML code, with the properties according to **Table 1**.

The associated source code is shown in **Listing 1**. You can code the surface interactively. Start the application and place the relevant XAML file in Visual Studio and the application side by side on the screen (**Figure 9**). Changes in the XAML code are immediately adopted when the application is started - without saving — and produce an updated display. This feature is called *Hot Reload* and is standard when creating graphical user interfaces.

What is interesting here is the control binding of the control elements of *TextBox* type with the properties of *Text*. Here you will find an expression according to the pattern in the XAML code:

### Listing 1. Definition of User Interfaces

```
<Page
    x:Class="App1.Views.MainPage"
    …>

    <Grid x:Name="ContentArea" Margin="">
        <StackPanel Background="">
            <TextBox
                Width="200"
                Margin="20,20,0,0"
                HorizontalAlignment="Left"
                Header="R1:"
                Text="" />
            <TextBox
                Width="200"
                Margin="20,10"
                HorizontalAlignment="Left"
                Header="U (out):"
                Text="" />
            <Button
                Width="200"
                Margin="20,10"
                Background="LightGreen"
                Command=""
                Content="Calc"
                FontWeight="Bold" />
            <TextBox
                Width="200"
                Margin="20,10"
                HorizontalAlignment="Left"
                Background="LightGray"
                Header="R2:"
                IsReadOnly="True"
                Text="" />
        </StackPanel>
    </Grid>
</Page>
```



Main

R1: 240

U (out): 0

Calc

R2: 0

```
cResource MediumLeftRightMargin}">
e SystemControlPageBackgroundChromeLowBrush}">

ode=TwoWay, UpdateSourceTrigger=PropertyChanged}" /:

 Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}"

            Background="LightGreen"
            Command="{x:Bind ViewModel.CalcCommand}"
            Content="Calc"
            FontWeight="Bold" />
        <TextBox
            Width="200"
            Margin="20,10"
            HorizontalAlignment="Left"
            Background="LightGray"
            Header="R2:"
            IsReadOnly="True"
            Text="{x:Bind ViewModel.R2, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}" /:
    </StackPanel>
</Grid>
```

```
13 Verweise
public class MainViewModel : ObservableRecipient
{
    5 Verweise
    public double R1 { get; set; } = 240;
    5 Verweise
    public double UOut { get; set; }

    private double r2;
    5 Verweise
    public double R2
    {
        get
        {
            return r2;
        }
        set
        {
            SetProperty(ref r2, value);
        }
    }

    public RelayCommand CalcCommand;
    1 Verweis
    public MainViewModel()
    {
        CalcCommand = new RelayCommand(calcCommandExecute);
    }

    1 Verweis
    private void calcCommandExecute()
    {
        R2 = (UOut / 1.25 - 1) * R1;
    }
}
```

*Figure 10: Example Relationship between View and ViewModel.*

```
Text="{x:Bind ViewModel.R1, Mode=TwoWay, UpdateSource
   Trigger=PropertyChanged}"
```

This means that the property of *Text* is bound to the variable *R1*. This is defined in the *ViewModel* page and is based on the MVVM concept. UI events are handled in the *View* layer and data is managed in the *Model* layer. The ViewModel represents the connection between the two layers. Thanks to the MVVM concept, all layers are decoupled and can be developed and maintained independently of one another. Information on the MVVM pattern can be found under [4].

## Program Logic

The program logic is implemented using C# (**Listing 2**). For this purpose, a program file (*ViewModel*) is assigned to each window of

**Listing 2. Program logic for the calculation in C#**

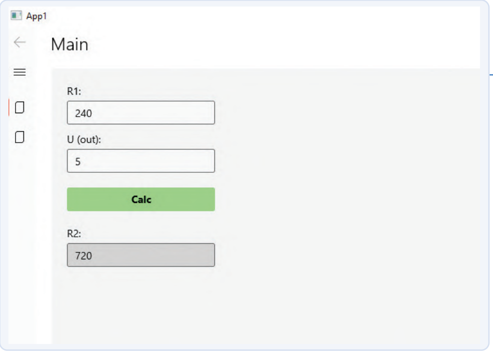
```
using CommunityToolkit.Mvvm.ComponentModel;
using CommunityToolkit.Mvvm.Input;

namespace App1.ViewModels
{
    public class MainViewModel : ObservableRecipient
    {
        public double R1 { get; set; } = 240;
        public double UOut { get; set; }

        private double r2;
        public double R2
        {
            get
            {
                return r2;
            }
            set
            {
                SetProperty(ref r2, value);
            }
        }


        public RelayCommand CalcCommand;
        public MainViewModel()
        {
            CalcCommand = new RelayCommand(calcComm
                andExecute);
        }

        private void calcCommandExecute()
        {
            R2 = (UOut / 1.25 - 1) * R1;
        }
    }
}
```

the user interface (*View*). In our example, it is the *MainViewModel* file assigned to the view *MainPage*. Here are some notes on the program code:

- **Import of libraries**: This is done via the *uses* statement. In our case we need two libraries for the MVVM pattern.
- **Definition of properties**: These must be *public* because we access the properties from outside, in this case from the view.
- **Automatic User Interface updating:** The `MainViewModel` class is derived from the `ObservableRecipient` base class, which was generated by the project wizard when the project was created. This class in turn implements the so-called `OnPropertyChanged` event. This ensures that when a value of a property changes, all bound elements are notified of the change. In our case the property `R2` is of interest. The value of `R2` is calculated in the program code. The so-called Setter of the property is called and the `OnPropertyChanged` event just described is triggered via the `SetProperty (…)` method. The *Text* property of the TextBox R2 is bound to the property `R2` (in the ViewModel). If `R2` is changed, the displayed value is automatically updated in the associated TextBox. This works thanks to data binding.
- **Pass on user actions by means of commands:** If the user presses the button, a command is triggered. The calculation method is linked to this command. Here, too, the user interface and program code are only linked to one another via the data link.
- **Assignment of View and ViewModel:** The program code (file: *MainViewModel.cs*) is assigned to the UI (file: *MainPage.xaml*). This is done in the page's code-behind file (file: *MainPage.xaml.cs*). You can see this if you take a look at the source code.
- **Calculation:** Calculation of value of `R2` takes place in the `calcCommandExecute (...)` method according to the above formula, which is then assigned to `R2`.

The user interface is therefore "loosely" linked to the program code by means of data binding. The connections to data binding just described are visualized using the example in **Figure 10**. Start the application and try it out. The value of the second resistor *R2* is calculated after entering the values of *R1* and $V_{Out}$ (**Figure 11**).

We have thus described the basic development model for desktop applications with the WinUI 3 graphics framework. From today's perspective, it will become a new standard under Windows and can also be used by other development environments and languages. The variety of graphical options for creating modern applications is impressive.



R1:
240

U (out):
5

R2:
720

*Figure 11: The finished app example.*

## Conclusions and Outlook

You can choose from a variety of technologies to create an application for the Windows operating system. The trend — also with a view to Windows 11 — is towards the use of WinUI 3. Using this will enable you to create an attractive and up to date user interface. From this perspective, it is worthwhile considering WinUI 3 when developing any new Windows application and to check migration options for any existing applications. The resulting applications have a contemporary interface and produce a good user experience. There are also no limitations on system access as there are with the UWP application model.

210407-01

## Contributors

Text and images: **Dr. Veikko Krypczyk**
Editor: **Jens Nickel**
Translation: **Martin Cooke**
Layout: **Giel Dols**

## Questions or Comments?

Do you have any technical questions or comments prompted by this article? If so, please contact the editor at editor@elektor.com.

## WEB LINKS

[1] Information for WinUI 3: https://docs.microsoft.com/en-us/windows/apps/winui/
[2] Visual Studio 2019: https://visualstudio.microsoft.com/
[3] Project page for this article: http://www.elektormagazine.com/210407-01
[4] Information for MVVM pattern: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel

Part 04

# Create GUIs with Python: World's worst GUI

Learn good GUI design by doing it all wrong first!

MAKER

Laura Sach

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

@CodeBoom

MAKER

Martin O'Hanlon

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

@martinohanlon

**It's time to really go to town with your GUIs and experiment with different widgets, colours, fonts, and features.** Like most experiments, it's likely that you won't get it right first time! In fact, you are going to explore the wrong way to approach creating your GUI.

## It's hard to read

The right choice of GUI colour and font are important. It's important that the contrast between background and text colour ensure that your GUI is easily readable. What you shouldn't do is use two very similar colours.

Import the widgets at the top of the code:

```
from guizero import App, Text
```

Create an app with a title:

```
app = App("it's all gone wrong")
title = Text(app, text="Some hard to read
text")

app.display()
```

Experiment by changing the colours, font, and text size (see **worst1.py** listing). Our choices are not the best!

```
app = App("it's all gone wrong", bg="dark
green")
title = Text(app, text="Some hard-to-
read text", size="14", font="Comic Sans",
color="green")
```

It's important that text on a GUI also stays around long enough to be read. It certainly shouldn't disappear or start flashing.

All widgets in guizero can be made invisible (or visible again) using the `hide()` and `show()` functions. Using the `repeat` function in guizero to run a function every second, you can make your text hide and show itself and appear to flash.

Create a function which will hide the text if it's visible and show it if it's not:

```
def flash_text():
    if title.visible:
        title.hide()
    else:
        title.show()
```

Before the app is displayed, use `repeat` to make the `flash_text` function run every 1000 milliseconds (1 second).

```
app.repeat(1000, flash_text)

app.display()
```

Your code should now look like **worst2.py**. Test your app: the title text should flash, appearing and disappearing once every second.

## The wrong widget

Using an appropriate widget can be the difference between a great GUI and one which is completely unusable.

Which widget would you use to enter a date? A TextBox? Multiple Combos? A TextBox would be more flexible but would require validation and formatting. Multiple Combos for year, month, and day wouldn't require validating but would be

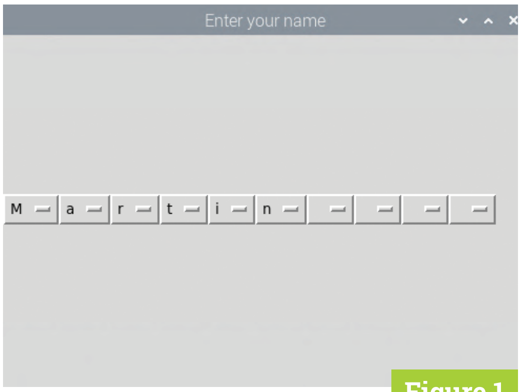


Figure 1

**Figure 1** A slider to set date and time

slower to use.

Using a Slider to set a date and time (**Figure 1**), as in the **worst3.py** code example, is not a great idea, though.

The Slider widget returns a number between 0 and 999,999,999. This is the number of seconds since 1 January 1970. The function `ctime()` is used to turn this number into a date and time.

Getting text from your user is simple: a TextBox or a multi-line TextBox should fulfil all your needs. Is it too simple, though? Does this require too much typing?

What about the user who just wants to use a mouse? Perhaps a series of Combos each containing all the letters in the alphabet would be better (**Figure 2**)? Start by importing the guizero widgets and `ascii_letters`.

```
from guizero import App, Combo
from string import ascii_letters
```

`ascii_letters` is a list containing all the 'printable' ASCII characters which you can use as the options for the Combo.

Create a single Combo which contains all the letters and displays the app.

```
 a_letter = Combo(app, options=" " + ascii_
letters, align="left")

 app.display()
```

Your program should now resemble **worst4.py**. Running it, you will see a single Combo which contains all the letters plus a space and is aligned to the left of the window.

To get a line of letters together, you could continually add Combo widgets to your app, e.g.:

```
 a_letter = Combo(app, options=" " + ascii_
letters, align="left")
 b_letter = Combo(app, options=" " + ascii_
letters, align="left")
 c_letter = Combo(app, options=" " + ascii_
letters, align="left")
```

By aligning each Combo widget to the left, the widgets are displayed next to each other against the left edge.

Alternatively, you could use a `for` loop, create a list of letters, and append each letter to the list, as shown in **worst5.py**.

Try both these approaches and see which you prefer. The `for` loop is more flexible as it allows you to create as many letters as you like.

## Pop-ups

No terrible GUI would be complete without a pop-up

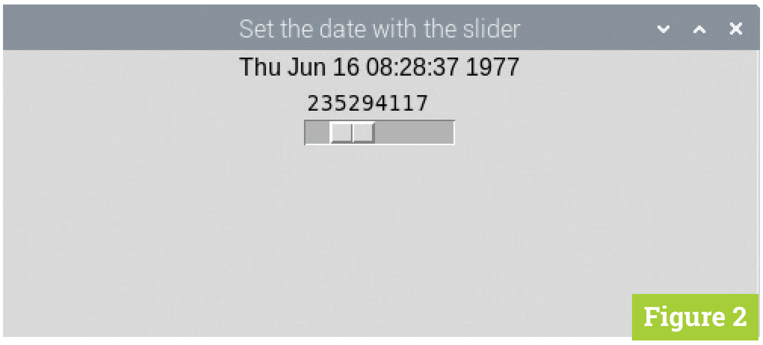
## worst1.py

Language: **Python 3**

DOWNLOAD THE FULL CODE:

magpi.cc/guizerocode

```
# Imports ---------------

from guizero import App, Text


# App -------------------

app = App("it's all gone wrong", bg="dark green")

title = Text(app, text="Hard to read", size="14", font="Comic
Sans", color="green")

app.display()
```



Set the date with the slider

Thu Jun 16 08:28:37 1977

235294117

Figure 2

**Figure 2** Combos to choose letters

## worst2.py

Language: **Python 3**

```
# Imports ---------------

from guizero import App, Text


# Functions -------------

def flash_text():
    if title.visible:
        title.hide()
    else:
        title.show()


# App -------------------

app = App("it's all gone wrong", bg="dark green")

title = Text(app, text="Hard to read", size="14", font="Comic
Sans", color="green")

app.repeat(1000, flash_text)

app.display()
```

## worst3.py

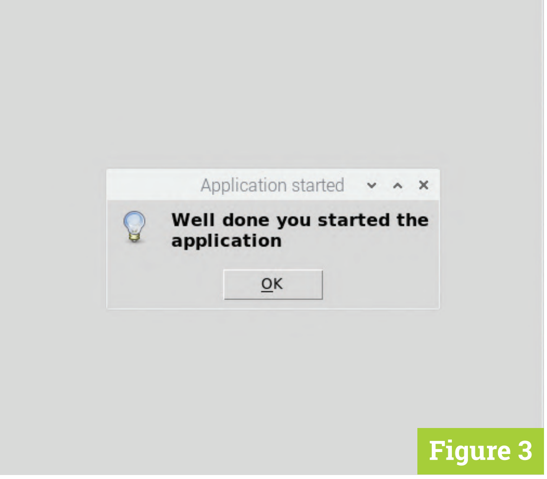
Language: **Python 3**

```python
# Imports ---------------

from guizero import App, Slider, Text
from time import ctime


# Functions -------------

def update_date():
    the_date.value = ctime(date_slider.value)


# App -------------------

app = App("Set the date with the slider")
the_date = Text(app)
date_slider = Slider(app, start=0, end=999999999,
command=update_date)

app.display()
```

## worst4.py

Language: **Python 3**

```python
# Imports ---------------
from guizero import App, Combo
from string import ascii_letters


# App -------------------

app = App("Enter your name")

a_letter = Combo(app, options=" " + ascii_letters, align="left")

app.display()
```



Well done you started the application

**Figure 3**
Pointless pop-up

## Window widget

Pop-up boxes can be used to ask users questions, but they are really simple.

If you want to do show additional information or ask for supplementary data, you could use the Window widget to create multiple windows.

Window is used in a similar way to App and has many of the same functions.

```python
from guizero import App, Window

app = App("Main window")
window = Window(app, "2nd Window")

app.display()
```

You can control whether a Window is on screen using the `show()` and `hide()` methods.

```python
window.show()
window.hide()
```

An app can be made to wait for a window to be closed after it has been shown, by passing True to the `wait` parameter of `show`. For example:

```python
window.show(wait=True)
```

You can find out more about how to use multiple windows in the guizero documentation: **lawsie.github.io/guizero/multiple_windows**.

box. guizero contains a number of pop-up boxes, which can be used to let users know something important or gather useful information. They can also be used to irritate and annoy users!

First, create an application which pops up a pointless box at the start to let you know the application has started.

```python
from guizero import App

app = App(title="pointless pop-ups")

app.info("Application started", "Well done
you started the application")

app.display()
```

Running your application, you will see that an 'info' box appears (**Figure 3**). The first parameter passed to info is the title of the window; the second parameter is the message.

You can change the style of this simple pop-up by using `warn` or `error` instead of `info`.

Pop-up boxes can also be used to get information

## worst5.py

Language: **Python 3**

```python
# Imports ---------------

from guizero import App, Combo
from string import ascii_letters


# App -------------------

app = App("Enter your name")

name_letters = []
for count in range(10):
    a_letter = Combo(app, options=" " + ascii_letters,
align="left")
    name_letters.append(a_letter)

app.display()
```

## 05-worlds-worst-gui.py

Language: **Python 3**

```python
from guizero import App, PushButton

def are_you_sure():
    if app.yesno("Confirmation", "Are you sure?"):
        app.info("Thanks", "Button pressed")
    else:
        app.error("Ok", "Cancelling")

app = App(title="pointless pop-ups")

button = PushButton(app, command=are_you_sure)

app.info("Application started", "Well done you started the
application")

app.display()
```

### Create Graphical User Interfaces with Python

For further tutorials on how to make your own GUIs with guizero, take a look at our book, *Create Graphical User Interfaces with Python*. Its 156 pages are packed with essential info and a range of exciting projects.
**magpi.cc/pythongui**

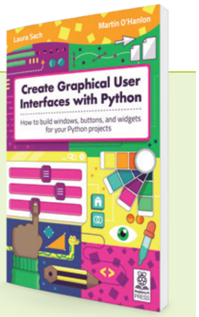
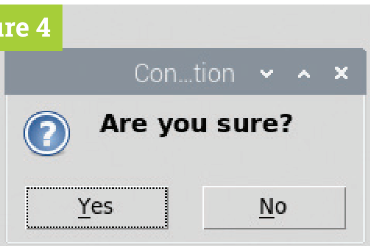

Laura Sach
Martin O'Hanlon
Create Graphical User Interfaces with Python
How to build windows, buttons, and widgets for your Python projects

Figure 4



Are you sure?

**Figure 4** Yes, we're sure!

from the user. The simplest is a `yesno` which will ask the user a question and get a True or False response. This is useful if you want a user to confirm before doing something, such as deleting a file. Perhaps not every time that they press a button, though! Import the PushButton widget into your application:

```python
from guizero import App, PushButton
```

Create a function which uses the `yesno` pop-up to ask for confirmation.

```python
def are_you_sure():
    if app.yesno("Confirmation", "Are you 
sure?"):
        app.info("Thanks", "Button 
pressed")
    else:
        app.error("Ok", "Cancelling")
```

Add the button to your GUI which calls the function when it is pressed.

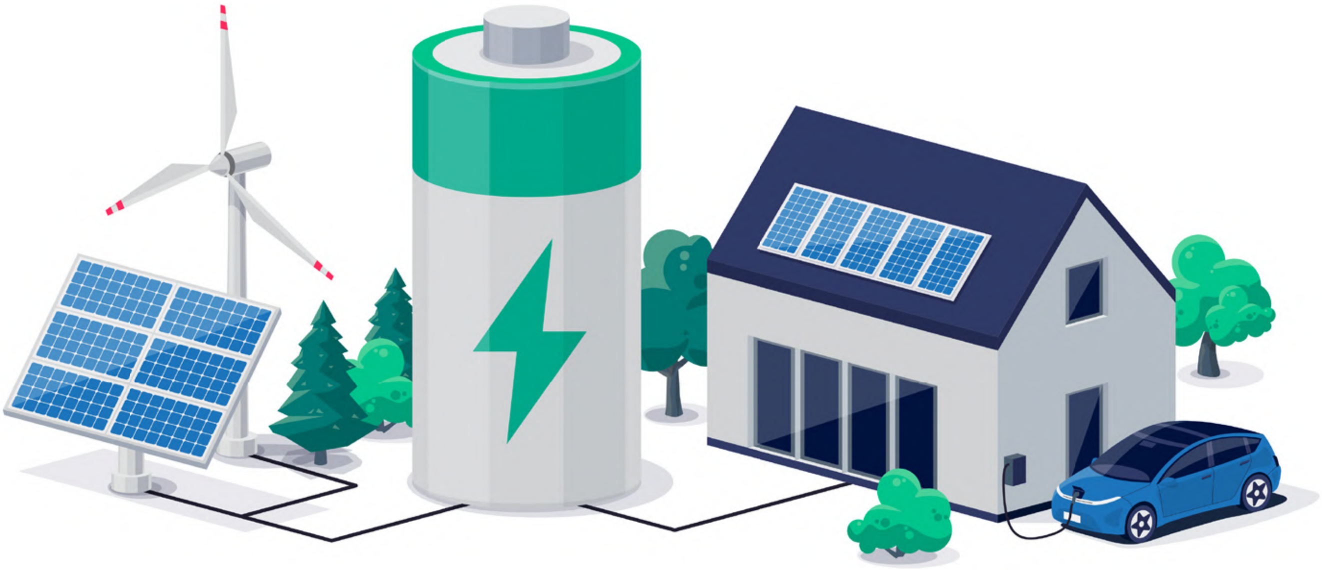
```python
button = PushButton(app, command=are_you_
sure)
```

Your code should now resemble **05-worlds-worst-gui.py**. When you run the application and press the button, you will see a pop-up asking to you confirm with a Yes or No (**Figure 4**).

You can find out more about the pop-up boxes in guizero at **lawsie.github.io/guizero/alerts**.

How about combining all of these 'features' into one great GUI?

Editor's note: This article originally appeared in *MagPi* magazine. Elektor publishes the Dutch, French, and German editions of *MagPi*.



# Off-Grid Solar Systems

## Electrical Energy Independent of the Mains Grid

By Dr. Thomas Scherer (Germany)

What is an off-grid solar system? Where are such installations necessary or practical? What are the most important design considerations? These questions and more will be answered in this article.

In the September/October 2021 edition of *Elektor*, we took a look at a photovoltaic system connected to the mains grid [1]. Here we will consider essentially autonomous solar installations that are isolated from the public grid. These can be used to generate electrical energy where otherwise a grid connection would be too costly, such as in a shed on an allotment, or impossible, such as on a motorboat or sailing boat. In general these are low-power systems which can handle a peak demand in the range of a few watts to a couple of kilowatts. Also, as the levels of feed-in tariffs continue to fall, new and simplified designs for fixed domestic solar installations that store the generated energy locally in rechargeable batteries solely for private use, rather than feeding into the public supply grid, begin to make more sense. These installations typically have a maximum nominal power output of a few kWp ('kilowatts peak'). Let us now look at these small-scale systems in more detail.

### Principle of Operation

An off-grid solar installation requires at least three components: the solar panel itself; some energy storage in the form of a rechargeable battery; and finally a charge controller that ensures that the battery is not overcharged. For smaller systems, typically operating at 12 V, that is in theory all that is needed. If, however, a 230 V AC output is required at 50 Hz or 60 Hz, a fourth component comes into play: an inverter. **Figure 1** shows a typical four-component solution: superficially it looks very simple, but as ever the devil is in the detail. In the following sections we will therefore take a look at these individual components.

Here's a real-life example: Klaus, a good friend of mine, decided to install a 12 V system in his shed because of the low prices and manageable size of the components involved. To design the system and specify the components there are two questions that first need to be answered.

### Energy and Power

The first question to answer is how much total energy needs to be stored by the system. This directly affects the required capacity of the rechargeable battery and hence it is necessary to estimate the average load on the installation. Bearing on this calculation is the number of cloudy days that the system must be able to 'survive.' Klaus would like to use an electric drill in his shed, and brew the occasional cup of tea, but these relatively rare loads do not significantly affect the average load calculation. More significant is the requirement to keep cool beer always at hand: this calls for a 12 V refrigerator operating continuously with an average power draw of 20 W. The system should be able to run for at least one day without sun.

The second question is the peak power requirement. From this we can determine the maximum current that will be drawn from the rechargeable battery and hence also specify the parameters of the charge controller (and inverter, if used). Usually this question is very easy to answer: in the case of my friend's allotment shed the answer was 1 kW, covering the power draw of the water heater, a standard drill and possibly also a water pump, all operating at 230 V.

### The Solar Panel

Over a 24-hour period the refrigerator in Klaus' shed will consume a maximum of 500 Wh. Although he lives in a sunny part of southwest Germany, the roof of his shed is unfortunately in the shadow of a tree and so the panel cannot be installed there. Instead it will have to be mounted vertically on the south-facing wall of the shed, which reduces its power output by some 30 % compared to mounting it at the optimal angle to the sun. The panel will therefore have to be over-specified by some 40 % to compensate for this loss. Fortunately there is plenty of space available and the price of panels has fallen considerably in recent years. An advantage of vertical mounting is that in winter snow will not lie on the panel, and moreover, since the sun is at a lower angle, the output will increase: in the best-case scenario the beer will be kept cool even on sunny winter days.

Now we can calculate the required power output from the panel. In this part of Germany we can reckon on an total incident energy of over 1200 kWh/m$^2$ over the course of a year. Over and above the expected daily energy use we should allow a safety margin of 100 %, and so for 500 Wh/day (from spring to autumn) we should be looking to generate 1 kWh/day. On the basis of 8 hours of sunshine per day we arrive at a required power output from the panel of around 125 Wp. On top of that we add the compensation for vertical mounting and arrive at 175 Wp. That means we need a 180 W panel, which will fit comfortably on the shed wall: see **Figure 2**.

### The Rechargeable Battery or Batteries

The energy required to provide one day of reserve power is at least 500 Wh. At a nominal voltage of 12 V we will need a battery with a capacity of at least 40 Ah. Since our inverter is specified with an output power of 1 kW, we must also keep in mind that at maximum load it will be drawing a current of at least 85 A at its input. This is an important consideration in selecting a battery. First we must decide on the battery chemistry. A lithium battery pack rated at 40 Ah can comfortably deal with this current (about '2C,' or twice the current that the battery can deliver for one hour) because of its low internal resistance. However, such a pack can easily cost over €250/US$280/£210

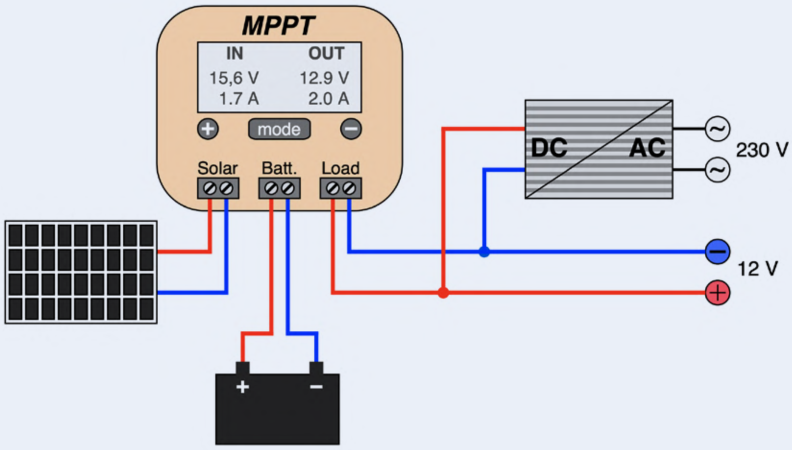


*Figure 1: The standard wiring of the four components normally found in an off-grid solar installation. The inverter on the right is only required if it is desired to run equipment designed for 230 V operation.*

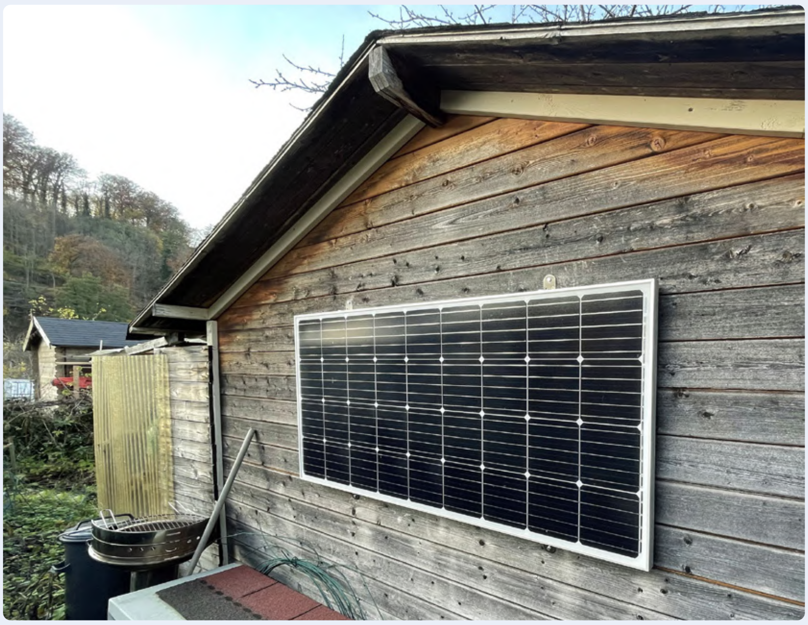


*Figure 2: The 12 V solar panel mounted vertically on the wall of Klaus' shed. It has an output rating of 180 Wp.*

and requires careful management. Instead, Klaus plumped for a simple lead-based battery costing a fraction of that price. An obvious choice was a car battery, since these are designed for high peak currents. However, they have disadvantages: low efficiency, low cycle life and high self-discharge rate. For that reason he compromised on a gel battery: this type does not like high discharge currents, and so two batteries, each rated at 36 Ah, were connected in parallel. This combination offers a nominal 864 Wh of stored energy and cost a little under €150/US$170/£130.

In direct sunshine the selected solar panel delivers so much power that it can easily fully charge the batteries in a single day, and their capacity is enough to cover 1.5 days without sunshine. I had some misgivings about the high current draw at the input of the inverter, but Klaus decided to give it a shot and, if two batteries proved not up to the job, to buy another and add it in parallel. With the system installed and the batteries fully charged, a first test was fun using a 1 kW water heater. While running the voltage at the battery terminals fell to 11.7 V, but nevertheless it was easy enough to bring half a litre of water to the boil. At a rough estimate the efficiency of the battery (output energy divided by input energy) at these high currents is at most 50 % and the current does not do the battery's health any favours. A further 36 Ah battery was therefore ordered and wired in: see **Figure 3**. Now the initial terminal voltage when discharging at 85 A is a more acceptable 12.6 V, and the total capacity has been increased to nearly 1.3 kWh, guaranteeing over two days of reserve.

### The Charge Controller

A search of eBay or of specialist distributors will turn up a wide range of charge controllers. Controllers rated at 10 A go for as little as €15/US$17/£13. However, a 180 Wp panel will deliver up to 15 A at 12 V, and so we need a controller rated at at least 20 A: these cost around €20/US$23/£17. If the charge controller is to be connected as shown in Figure 1, then it is better to choose a 100 A version, costing perhaps a little under €50/US$55/£45. Now let's go into the details.

The charge controller has the job of charging the battery using the power delivered from the panel, and terminating the charging process when a threshold voltage is reached. This

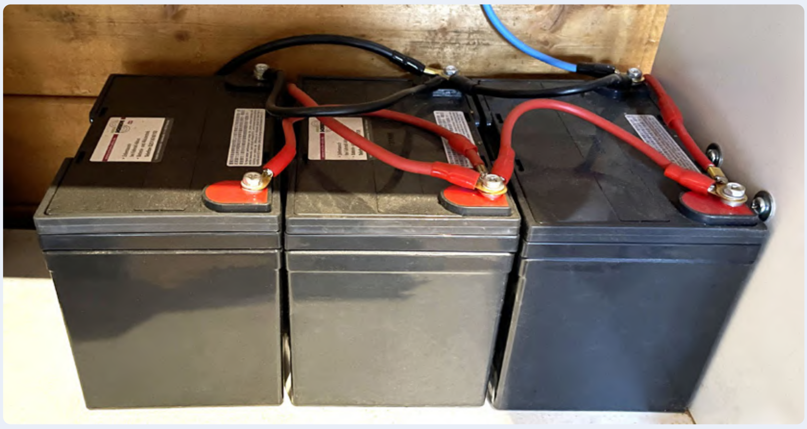


*Figure 3: Three 12 V lead gel batteries, each rated at 36 Ah, are wired in parallel to act as energy storage in Klaus' shed.*

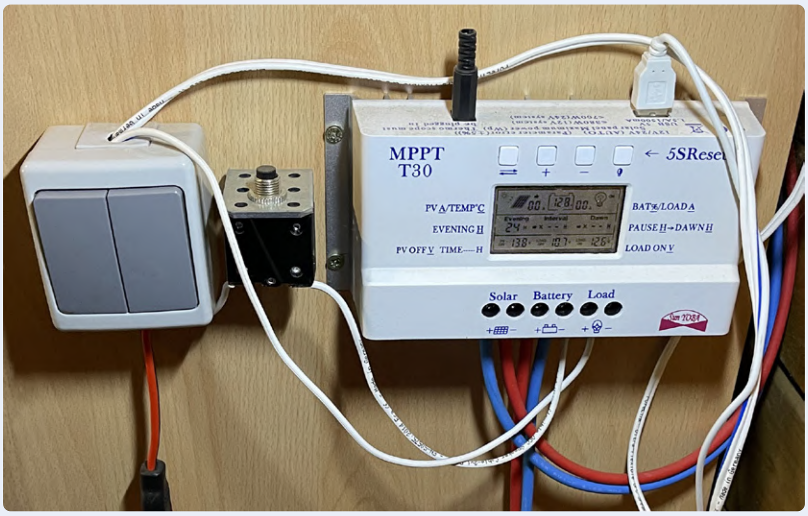


*Figure 4: From left to right: light switch, 30 A electromagnetic circuit breaker, and MPPT charge controller.*

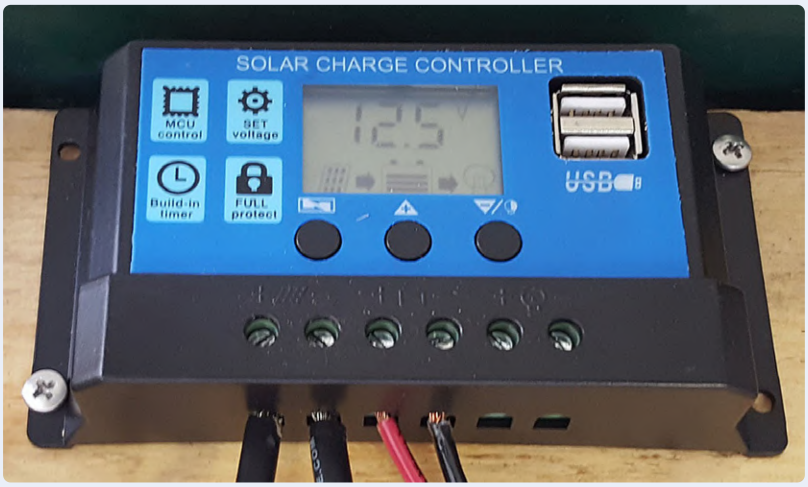


*Figure 5: Charge controllers that look like this one certainly do not have MPPT functionality (even if they carry a sticker bearing those letters!). (Source: United States Department of Energy)*

ensures that the connected battery is not overcharged and hence damaged. Almost all charge controllers also control the load connection and ensure that the load is disconnected when a lower voltage threshold is reached, this time protecting the battery from deep discharge. They invariably employ a microcontroller and so most can be configured to work with different battery types, including lead-acid, lead gel and lithium chemistries. They also automatically adapt to a nominal terminal voltage of either 12 V or 24 V. Often it is also possible to configure the undervoltage and overvoltage thresholds manually.

The next consideration is the charger topology. All low-cost examples use PWM control, even if it says 'MPPT' on the device: labels are cheap, but a 'real' MPPT charger is better, more complicated and therefore rather more expensive.

In a PWM controller the charge current is adjusted so that the output voltage of the panel is just above the current battery terminal voltage. The battery is therefore charged at the maximum possible current given the amount of incident light on the panel and its size, and the state of charge of the battery over a wide range of conditions. The circuit to do this requires just a simple microcontroller and a power MOSFET: a low-cost solution, but not optimal.

Now, the output power of a panel is given by the product of its output voltage and output current. For any given panel and level of illumination there is a point where this product is maximized; almost invariably at this maximum power point the output voltage of the panel is above the battery voltage. An MPPT (maximum power point tracking) controller continuously determines where this 'sweet spot' is and drives a step-down voltage regulator circuit such that it draws the optimal current and delivers the maximum possible output power. In the best case the output power from an MPPT controller can be 30% higher than that from a PWM controller. However, this comes at a cost : even a low-cost MPPT controller will set you back over €50/US$55/£45, and a name-brand unit will be at least €100/US$110/£85. The 30 A charge controller shown in **Figure 4** is a low-cost MPPT type, and costs about €60/US$70/£50, but Klaus felt that the extra output power was worth the money. If you are looking for an MPPT controller, avoid ones like that shown in **Figure 5**: this type is available in a range of colours and with different markings.

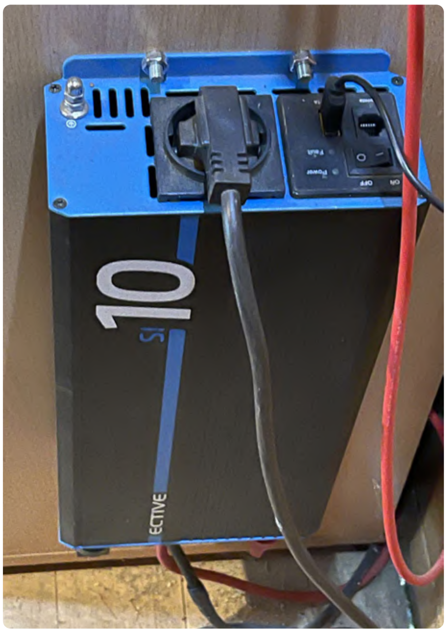
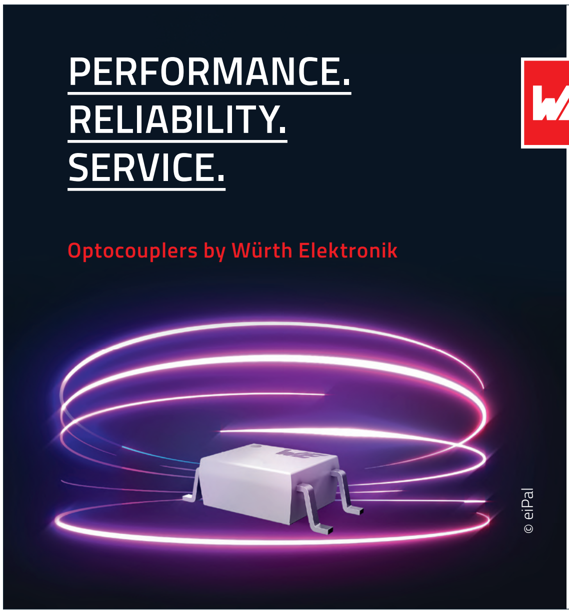


*Figure 6: This 1 kW inverter made by Ective has proved very stable and reliable over time.*

### The Inverter

If it is a requirement to generate a 230 V AC output, then an inverter is essential. Low-cost examples with implausible power specifications and output voltage waveforms at best distantly related to a sinusoid are best given a wide berth. An important point to note is that the specification for maximum continuous



power is given for an ohmic load. Klaus' 1 kW inverter is perfectly suitable for driving a 1 kW 500 ml water heater; but the situation with an inductive or, less commonly, capacitive load is completely different. In this case we must also check the reactive power: note that the apparent power is always at least as great as the true power. Among the most problematic devices are electric motors, which can exhibit high inrush currents that will trigger the built-in overload protection circuitry on an inadequately-rated inverter. An allowance of 100 % when running motors is not excessive even for a high-quality inverter. Klaus' 1 kW inverter, shown in **Figure 6**, can comfortably handle an electric drill and a water pump rated at 450 W. It cost over €200/US$225/£170.

### Wiring

As you may have surmised from the images so far, the wiring between the solar panel, charge controller, and battery is done using multi-stranded wire with a 6 mm² cross-section; the connections to the 12 V cigarette lighter sockets are not shown. The parallel connections between the batteries themselves are made using 16 mm² wire. The thicknesses of wire used need to be appropriate for the currents to be carried: this is not a good place to try to save money.

The inverter is connected directly to the battery using 16 mm² wire to minimize losses. A direct connection is only possible when the inverter (like the one here) offers undervoltage protection, switching off to protect the battery from deep discharge. The inverter is only enabled when the 230 V output is actually required: its quiescent current consumption, in the tens of milliamps, would otherwise be an unnecessary waste of energy. The final set-up is thus as shown in **Figure 7**.

## Other Off-Grid Systems

The electricity supply in Klaus' shed is a typical example of an off-grid solar installation. Various suppliers offer ready-to-go packages comprising a solar panel, charge controller and inverter, with various nominal power levels. If you decide to opt for wind energy rather than solar, then again suitable generators and charge controllers are available based on broadly the same principles. For my part, last year I modified my robot lawn mower for autonomous power [2]. This needed just a 50 W panel and a simple PWM charge controller; no inverter was neces-

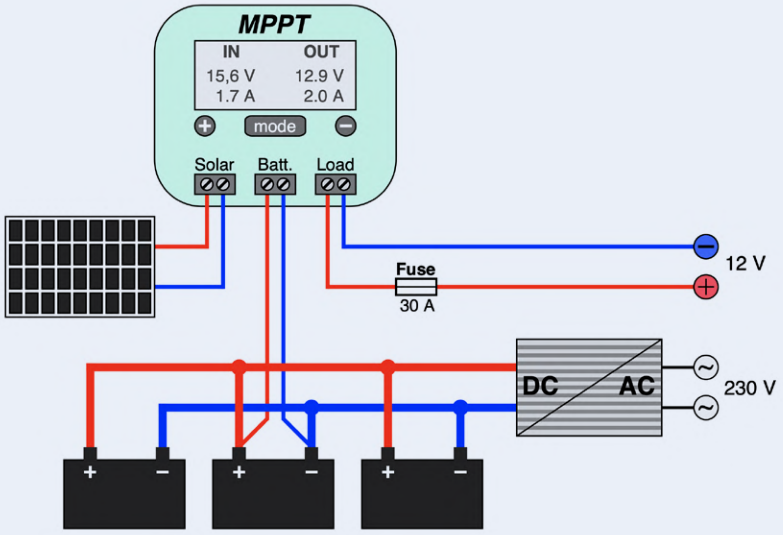


*Figure 7: In Klaus' installation the inverter is connected directly to the rechargeable battery, and the 12 V output of the charge controller is provided with extra protection.*

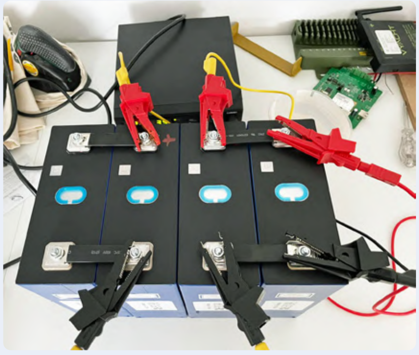


*Figure 8: LiFePO$_4$ batteries under test before installation in Martin's boat. (Source: Martin Jepkens)*

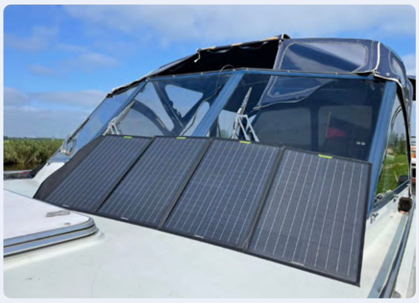


*Figure 9: The foldable solar panel chosen by both Martin and Detlev can be stored below deck when underway. It is rated at 120 Wp. (Source: Martin Jepkens)*

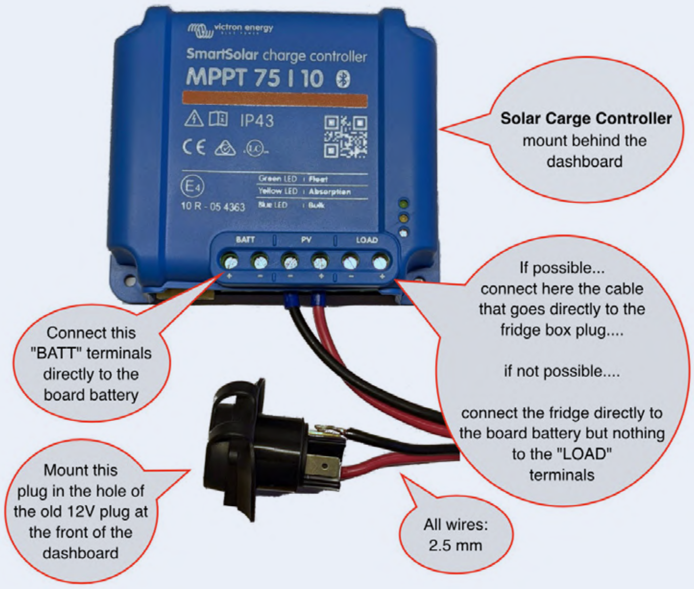

victron energy
SmartSolar charge controller
MPPT 75 | 10
IP43
BATT
PV
LOAD
Solar Carge Controller
mount behind the dashboard
If possible...
connect here the cable that goes directly to the fridge box plug....
if not possible....
connect the fridge directly to the board battery but nothing to the "LOAD" terminals
Connect this "BATT" terminals directly to the board battery
Mount this plug in the hole of the old 12V plug at the front of the dashboard
All wires: 2.5 mm

*Figure 10: Guide to installing the charge controller in Detlev's boat.*

sary. Since then I upped the battery capacity from 12 Ah to 30 Ah to help cover the periods of rainy weather we have had. I have also recently replaced the PWM charge controller with a better (and more expensive) MPPT controller, and the system can now generate enough electricity to mow the lawn even late into autumn.

There are of course many other applications for off-grid power. Two of my other friends have boats: Martin navigates his large steel-hulled boat through the riverscapes of the Netherlands, while Detlev makes mischief in the Med in his sportsboat with a planing hull. Both often spend days away from a harbour or other mooring where electricity is available and therefore would like to have more independence, especially as far as refrigeration is concerned: not in this instance just for beer, but for other sustenance as well. It would be ecologically unfriendly, not to mention inefficient, to run the engine frequently in order to charge the on-board battery, and so both have installed solar systems.

Now Martin is a smart engineer and doesn't need to rely on any advice from me. Nevertheless he discusses his ideas with me from time to time. He was wondering whether the generator on his boat could be overloaded if he happened to connect a huge 200 Ah $LiFePO_4$ battery across it. The dangers of such a course of action are explained in a YouTube video [3]. **Figure 8** shows how he set up his batteries for capacity testing: he decided on a battery with a $LiFePO_4$ chemistry mainly because of their long cycle life, but also because of their compactness compared to lead-based batteries. In Martin's boat the on-board battery is separate from the starter battery. In order to ease the burden on the alternator, the batteries are each charged via their own charge controller when the engine is running. A 120 Wp foldable solar panel and charge controller are also fitted for charging when underway (see **Figure 9**).

Because of the lack of available space, a fixed solar panel installation is not a practical proposition on Detlev's sportsboat. He therefore decided to use the same type of solar panel as Martin, although neither knew what the other had chosen! Detlev is not an electronics specialist, and at first he wanted to use his extra 120 A on-board battery because it was still rather new. I did some calculations for him and advised him that using the cigarette lighter socket on the 'bridge' of his vessel to connect the solar panel was not a great idea from a reliability point of view: I suggested the use a a waterproof Neutrik connector instead. I pre-wired the connector and drew up an installation guide (**Figure 10**) for his boatbuilder, so that the whole system could be set up in harbour in Istria. The combination of

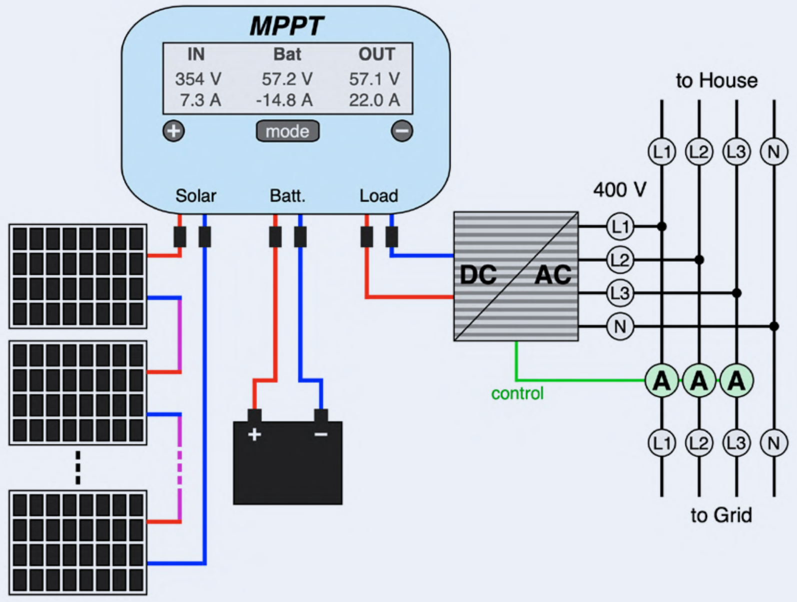


*Figure 11: Semi-off-grid solar installation for a house. The three-phase current measurement controls the inverter in such a way that no electrical energy is driven into the grid.*

foldable 120 Wp panel plus a Victron Energy MPPT charge controller came to a total of just under €500/US$550/£450. The charge controller has a Bluetooth connection and all parameters and graphs can be monitored using a smartphone app.

## The Semi-Off-Grid House

In these times of dwindling feed-in tariffs interest has grown in a version of the fixed solar installation that as far as possible dedicates all of the generated energy to the demand of the house. An array of, for example, 10 modern solar panels will generate some 3.75 kWp; a suitable MPPT solar charge controller could then charge a $LiFePO_4$ battery with a capacity of say 6.5 kWh; and then a three-phase inverter could be controlled using a current monitoring circuit (the three ammeters towards the bottom right of **Figure 11**) to ensure that under no circumstances is any electrical energy driven into the grid. All the 'current' is therefore locally used. With electricity in Europe in 2022 costing around €0.35/US$0.40/£0.30 per kWh this is a very attractive option: not only does it avoid having a complex and also expensive grid-tied inverter with integrated charging electronics for the battery, it also avoids a large amount of bureaucracy: not a negligible consideration! (Note that such an arrangement may not be legal in all countries.)

In the arrangement in Figure 11 the savings from using the simplified design add up to €1000/US$1150/£850 to €2000/US$2300/£1700. It would take a few years to make up for that using a feed-in tariff. The most costly part of the system is the battery: a 6.5 kWh $LiFePO_4$ battery costs over €3000/US$3400/£2600. With a guaranteed 6000 charging cycles at 90 % discharge depth that means that around 36 MWh of energy will have flowed via the battery, making the battery cost around €0.09/US$0.10/£0.08 per kWh. Furthermore, at that point the battery is still not completely dead and so the effective cost per kWh will be even lower. Over the life of the battery a solution like this can save costs of very roughly €13000/US$15000/£11000 using energy one has generated locally. If an electric vehicle is also charged (at a low charge rate) then an installation of this kind can pay for itself within a few years.

210644-01

### Contributors

Text and figures (unless otherwise stated):
**Dr. Thomas Scherer**
Editor: **Jens Nickel**
Translation: **Mark Owen**
Layout: **Harmen Heida**

### Questions or Comments?

Feel free to send technical questions to the Elektor editorial team by e-mail to editor@elektor.com.

### RELATED PRODUCTS

- **PeakTech 4350 Clamp Meter (SKU 18161)**
  www.elektor.com/18161
- **Pokit Meter: Portable Multimeter, Oscilloscope and Logger (SKU 19854)**
  www.elektor.com/19854
- **PeakTech 3445 True RMS Digital Multimeter with Bluetooth (SKU 18774)**
  www.elektor.com/18774

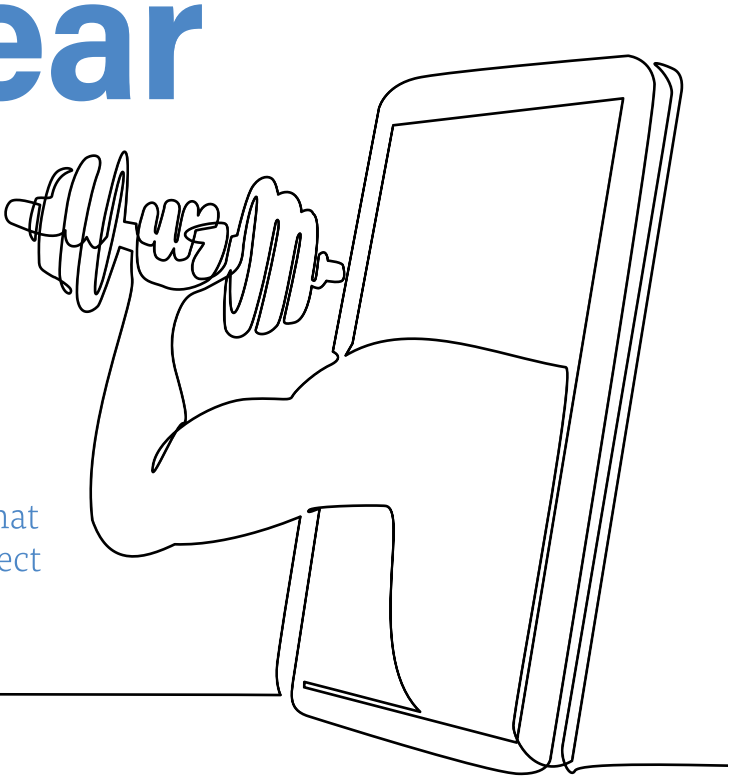
### WEB LINKS

[1] T. Scherer, "Balcony Power Plant," Elektor Magazine September/October 2021: www.elektormagazine.com/210326-01
[2] T. Scherer, "Solar Power for Mowing Robots," Elektor Magazine July/August 2021: www.elektormagazine.com/200553-01
[3] Victron Energy, "How to not blow up your alternator when charging lithium," YouTube: www.youtube.com/watch?v=jgolocPgOug

# The 10-Year Smartphone

## Renew Your Expectations

By Priscilla Haring-Kuipers (The Netherlands)



What we expect from our devices drives what we accept from our devices. We do not expect a lot of time from our smartphone.

Maybe you've seen the ads online for a new smartphone that will last you 10 years. Maybe you too clicked on one, only to be disappointed that this phone does not exist ... yet. The "10-year phone" is an initiative that has been trying to get your attention, and the EU's legislative attention, by tempting us with a future in which it becomes the norm for a smartphone to last a decade. In order for this to actually work, there would have to be legislation on repairability, continued software support and availability of parts — especially the battery. Information should be readily available on how to repair your phone yourself along with an official repairability score [1].

### Timing

Somewhere along the journey of increasingly amazing mobile phones, we appear to have accepted that these expensive lifelines are only going to last two to four years (in full working order). I guess we don't often stop to consider how much time we actually expect from our devices. How long should your 4K flat screen last? What about your washing machine, your wake-up light, or your reflow oven?

Giving you two to four years doesn't seem so bad for most devices as you will probably want something new in this timeframe anyway. Which is a strange notion. Where did this drive to continuously upgrade come from and isn't it time we got ourselves rid of it? In light of our planet drowning in our stuff, the resources and suffering involved in making many of our devices and our own appreciation of what we have. How about we adjust our devices and our mental models to last longer?

### New Reflex

My smartphone is now around five years old, and I desperately need to have the battery replaced. It drops from 42% to dead in an instant. I found myself browsing for a new phone until I realised that I am still very happy with my phone beyond the suicidal battery. I am not alone in this new reflex; only 11% of people in the EU will repair their phone when it fails them. Would I also immediately browse for a new washing machine if it suddenly refused to spin cycle? I don't think so. I think my reflex would be to google the problem to see if maybe the internet knows that this happens all the time and I should just unclog the something-nozzle. Maintaining and repairing it myself. Should that fail, I would probably get a repair professional IF the washing machine is not older than 10 years.

Washing machines and smart phones are in the same price range. But we don't expect our phones to last a decade. It's time that we smarten up.

*The 10-Year Smartphone is an initiative of the European Right to Repair campaign, a coalition of over 80 organisations from across Europe, pushing for longer-lasting and more repairable products.*

210714-01

### WEB LINK

[1] "10 Year Smartphone": https://10yearphone.com/

elektor store

www.elektor.com

# 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**!

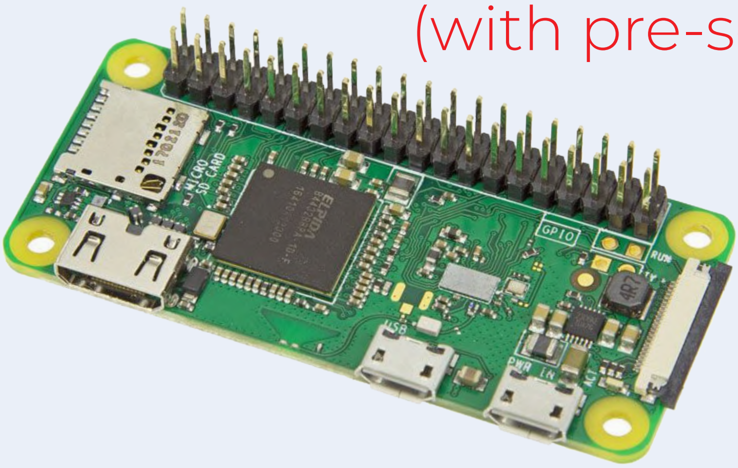
## puhui T-962 Infrared Reflow Oven

Price: €229.00

**Member Price: €206.10**

www.elektor.com/20135

## Raspberry Pi Zero 2 WH (with pre-soldered GPIO Header)



Price: €34.95

**Special Price: €29.95**

www.elektor.com/20157

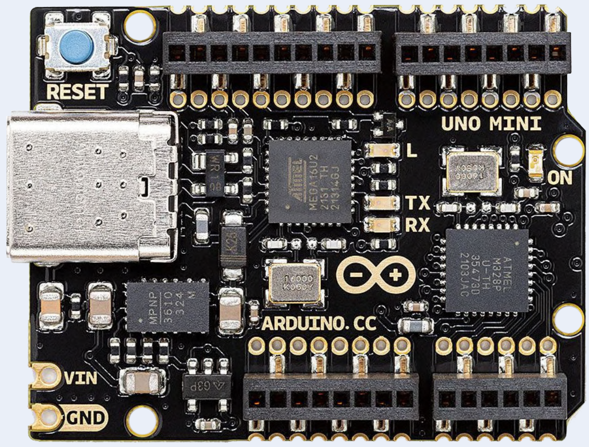
## MakePython ESP32 Development Kit

Price: €74.95

**Special Price: €64.95**

🛒 www.elektor.com/20137

## Arduino Uno Mini (Limited Edition)

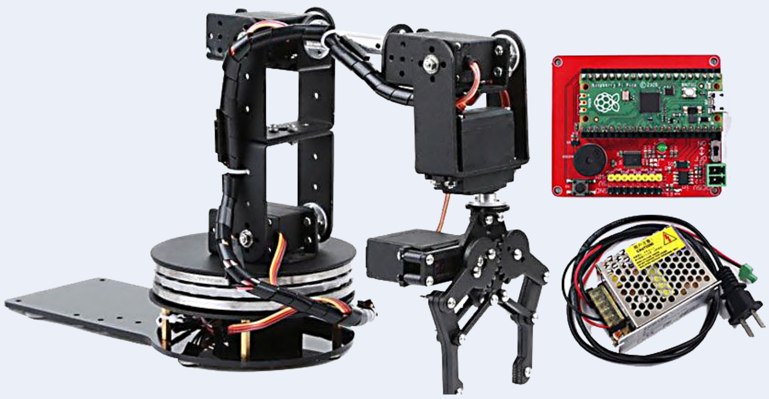


Price: €54.95

**Member Price: €49.46**

🛒 www.elektor.com/20098

## Makerfabs 6 DOF Robot Arm with Raspberry Pi Pico

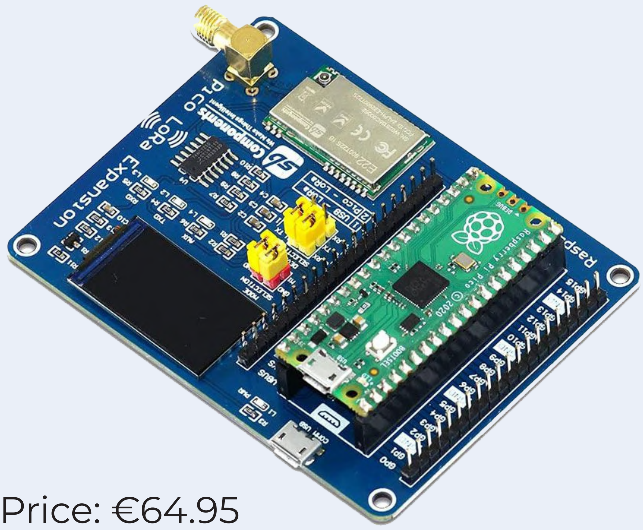


Price: €139.95

**Member Price: €125.96**

🛒 www.elektor.com/20130

## Raspberry Pi Pico LoRa Expansion + FREE Raspberry Pi Pico



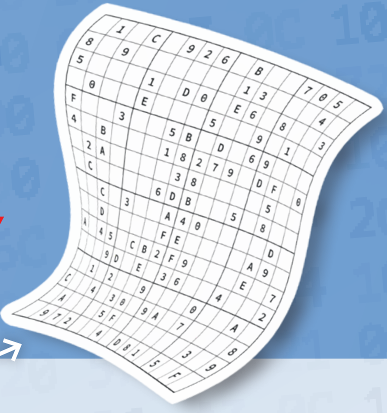
Price: €64.95

**Member Price: €58.46**

🛒 www.elektor.com/20096

# 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 June 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 02-03/2022 (March & April) is: **C73B8**.
Solutions submitted to us before April 15th were entered in a prize draw for 5 Elektor Store Vouchers.
The winners are posted at www.elektormagazine.com/hexadoku.

**Congratulations everyone!**

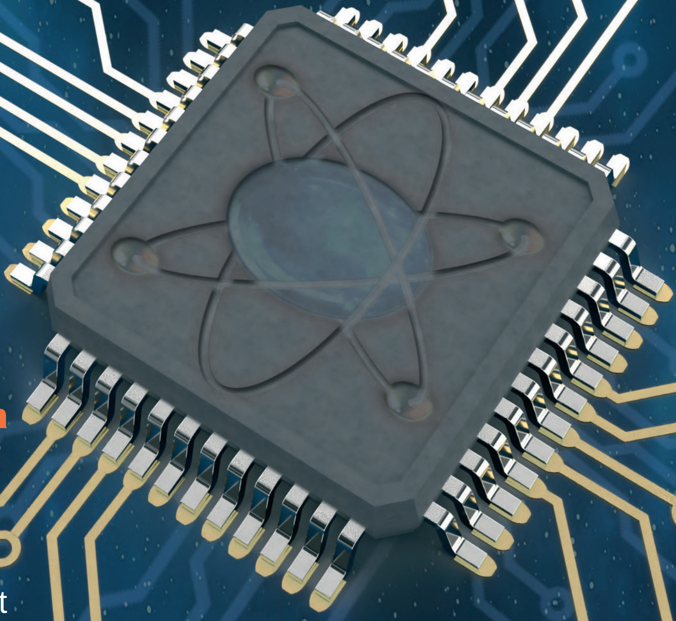
| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | D | 7 | | 1 | 6 | | | | | | | B | | 8 | 0 |
| 4 | 6 | | | D | 5 | | | 0 | | | | | C | F | |
| 2 | | | | F | 4 | | | | | B | | | | E | 3 |
| | | | 3 | 8 | | 2 | | A | 9 | | D | | | 5 | |
| D | E | B | 5 | 2 | | | 6 | | | | 8 | | | 3 | C |
| 8 | 0 | 3 | | | | 1 | | | 2 | F | C | D | | | |
| | | | 4 | | 8 | | 0 | | | 7 | | F | | | |
| | | | | 3 | | 4 | | | A | | 5 | | | | |
| | 1 | | E | | | | | | | | B | | 6 | | F |
| | | | C | | D | | 8 | | 7 | | | E | | 9 | B |
| | | 2 | | | E | 0 | | | | | F | 5 | 3 | 7 | |
| | | | 7 | B | F | | 3 | E | | 1 | 4 | | | | |
| 6 | | | | | 1 | A | | | C | D | | 3 | 9 | | 5 |
| | A | | | | | | | 1 | | 5 | | 4 | | | 8 |
| 9 | C | 5 | D | 7 | | | | | 4 | 2 | | | | | 1 |
| B | | 4 | | 6 | | | | 9 | E | | | C | 7 | A | |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E | 2 | 6 | 9 | 0 | 3 | C | A | F | 5 | 7 | B | 1 | 4 | 8 | D |
| 4 | C | 7 | 3 | B | 8 | 5 | D | E | 1 | 9 | 6 | A | F | 2 | 0 |
| 5 | B | 0 | D | E | F | 4 | 1 | C | 2 | 8 | A | 3 | 6 | 7 | 9 |
| 8 | F | 1 | A | 2 | 6 | 7 | 9 | 3 | D | 0 | 4 | 5 | C | B | E |
| A | 3 | 2 | 0 | 5 | C | F | 8 | 4 | 6 | D | 9 | B | 7 | E | 1 |
| 1 | D | B | 8 | 4 | 2 | E | 6 | 0 | 7 | A | 5 | C | 3 | 9 | F |
| 7 | 5 | C | 6 | 1 | 9 | A | B | 2 | 3 | E | F | 4 | 0 | D | 8 |
| F | E | 9 | 4 | 3 | 7 | D | 0 | 1 | 8 | B | C | 2 | A | 5 | 6 |
| 3 | 0 | D | E | 6 | A | 9 | F | B | C | 1 | 2 | 7 | 8 | 4 | 5 |
| 2 | 6 | F | 1 | C | B | 8 | E | 5 | 4 | 3 | 7 | 9 | D | 0 | A |
| 9 | 4 | 5 | 7 | D | 0 | 1 | 2 | 8 | A | F | E | 6 | B | 3 | C |
| B | 8 | A | C | 7 | 5 | 3 | 4 | 9 | 0 | 6 | D | E | 1 | F | 2 |
| 0 | 7 | 3 | 2 | 8 | 4 | 6 | 5 | D | E | C | 1 | F | 9 | A | B |
| 6 | A | 4 | B | F | E | 0 | C | 7 | 9 | 5 | 8 | D | 2 | 1 | 3 |
| C | 1 | E | F | 9 | D | 2 | 3 | A | B | 4 | 0 | 8 | 5 | 6 | 7 |
| D | 9 | 8 | 5 | A | 1 | B | 7 | 6 | F | 2 | 3 | 0 | E | C | 4 |

*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

### 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

### Pre-Production Checklist

- Set of board tests before Gerber Output
- Includes placement, connectivity and clearance testing
- Completely independant code for clearance checks

labcenter Electronics

www.labcenter.com

info@labcenter.com

☎ +44 (0)1756 753440