

FOCUS ON

Embedded & AI

The Connected Autonomous Vehicle and Its Environment

An Introduction to Real and Reduced-Scale Autonomous Vehicles



Virtual Assistant with ChatGPT & Raspberry Pi

Unleash the Power of AI in an Easy Way



The Intel 8279 Keyboard/Display Interface

Infographics

Embedded & AI



Makerfabs SenseLoRa

Plug and Play IoT for Greenhouses

Join the Elektor Community



Take out a
membership!



- ✓ The Elektor web archive from 1974!
- ✓ 8x Elektor Magazine (print)
- ✓ 8x Elektor Magazine (digital)
- ✓ 10% discount in our web shop and exclusive offers
- ✓ Access to more than 5,000 Gerber files



Also available

The Digital
membership!



- ✓ The Elektor web archive from 1974!
- ✓ 8x Elektor Magazine (digital)
- ✓ 10% discount in our web shop and exclusive offers
- ✓ Access to more than 5,000 Gerber files



www.elektormagazine.com/member



CONTENTS

3 Colophon

4 Virtual Assistant with ChatGPT and Raspberry Pi

Unleash the Power of AI in an Easy Way



12 The Connected Autonomous Vehicle and Its Environment

An Introduction to Real and Reduced-Scale Autonomous Vehicles

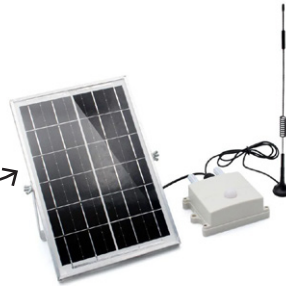
20 Infographics: Embedded and AI

22 Peculiar Parts

The Intel 8279 Keyboard/Display Interface

24 Makerfabs SenseLoRa

Plug and Play IoT for Greenhouses



The March/
April 2025 edition of
ElektorMag is available
at newsstands and in
the Elektor Store.



C. J. Abate

Content Director, Elektor

Embedded & AI Today and Tomorrow

Embedded systems are at the heart of modern technology, powering everything from predictive maintenance systems in factories to life-saving medical devices. As AI integration advances, embedded solutions are becoming smarter, more efficient, and capable of instantaneous decision-making. In this Bonus Edition of ElektorMag, we share helpful insights into these rapidly evolving fields.

Read on for a hands-on guide to building a Virtual Assistant with ChatGPT and Raspberry Pi, a plug-and-play IoT solution designed for smart greenhouse management, and more. These projects and insights highlight the limitless potential of embedded systems when combined with AI and IoT.

We're also excited for embedded world 2025 in Nuremberg, Germany (March 11-13, 2025). If you attend the event, head over to booth 5-181 to meet our editors and engineers. Together, we can explore how embedded AI is shaping the future. If you can't attend in person, look out for our online coverage.

Enjoy this Bonus edition. We look forward to seeing your projects on the Elektor Labs online platform!

The Team

International Editor-in-Chief: Jens Nickel | **Content Director:** C. J. Abate | **International Editorial Staff:** Asma Adhimi, Roberto Armani, Eric Bogers, Jan Buiting, Rolf Gerstendorf (RG), Ton Giesberts, Saad Imtiaz, Alina Neacsu, Dr. Thomas Scherer, Jean-Francois Simon, Clemens Valens, Brian Tristram Williams | **Regular Contributors:** David Ashton, Stuart Cording, Tam Hanna, Ilse Joostens, Prof. Dr. Martin Ossmann, Alfred Rosenkränzer | **Graphic Design & Prepress:** Harmen Heida, Sylvia Sopamena, Patrick Wielders | **Publisher:** Erik Jansen | **Technical questions:** editor@elektor.com

COLOPHON

Elektor March/April 2025 Bonus Edition

Elektor Magazine is published 8 times a year by
Elektor International Media b.v.
PO Box 11, 6114 ZG Susteren, The Netherlands

Phone: +31 46 4389444

elektor.com | elektormagazine.com

For all your questions
service@elektor.com

Become a Member
elektormagazine.com/membership

Advertising & Sponsoring
Büsa Kas
Tel. +49 (0)241 95509178
busa.kas@elektor.com
elektormagazine.com/advertising

Copyright Notice

© Elektor International Media b.v. 2025

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

responsibility for the safe and proper function of reader-assembled projects based upon or from schematics, descriptions or information published in or in relation with Elektor magazine.

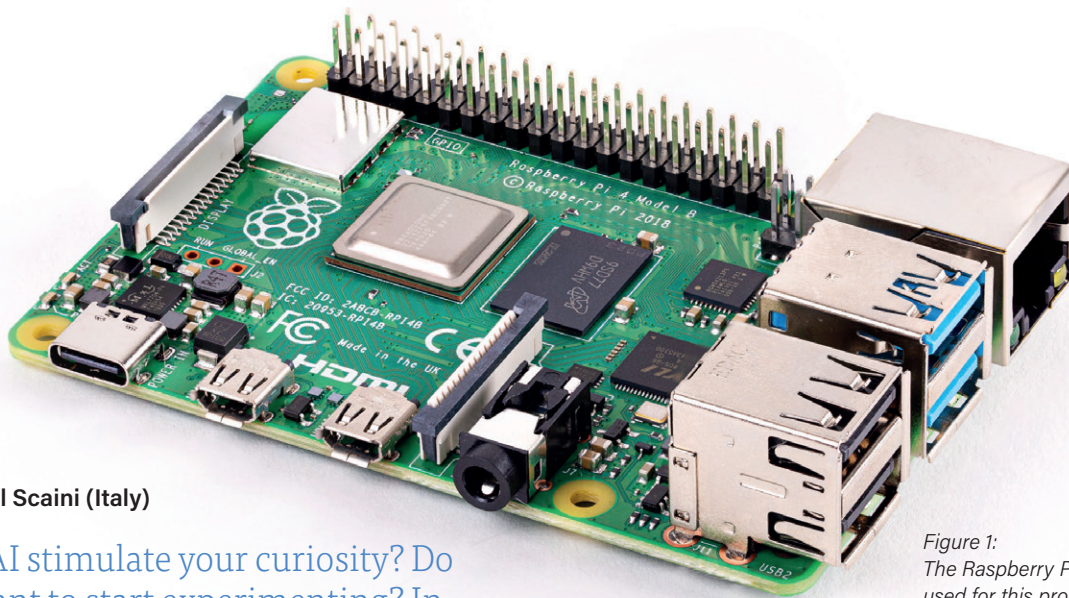
International Editor-in-Chief
Jens Nickel

Content Director
C. J. Abate

Publisher
Erik Jansen

Virtual Assistant with ChatGPT and Raspberry Pi

Unleash the Power of AI in an Easy Way



By Daniel Scaini (Italy)

Does AI stimulate your curiosity? Do you want to start experimenting? In this article we turn a Raspberry Pi into a powerful intelligent system with Python and ChatGPT. Follow this step-by-step guide to create an advanced virtual assistant that's ready to interact in a natural and surprising way.

One of the most debated and topical issues recently is, no doubt, artificial intelligence (AI) and its developments within our society. The technical scenario of artificial intelligence has evolved considerably since 1950, when mathematician Alan Turing was the first to ask whether data processors were capable of thinking.

The term AI was created in 1956, and since then, artificial intelligence has evolved through several stages, starting with symbolic artificial intelligence, based on logical systems built by humans, moving through a period of stalemate in the 1970s (referred to as the "winter" of AI), and then arriving in the 1990s at the chess-playing Deep Blue computer.

Since 2011, advances in machine learning - a subgroup of AI that uses a statistical approach — have improved the ability of computers to

make predictions, based on historical data. The maturity of so-called "neural networks" — a machine learning (ML) modeling technique — along with the availability of large amounts of data and increased computing power, is behind the expansion of AI development.

Today, we find its application within numerous domains, we are all familiar with the conversational assistants deployed by major manufacturers: IBM's Watson, Amazon's Alexa, Google's Google Home, Apple's Siri and many others. In this case, we are talking about software that can conduct a competent conversation as similar as possible to that which takes place between human beings, and it does so through Natural Language Processing (NLP) techniques.

These are tools that can provide a great deal of information and provide support for everyday actions: in medicine, automotive, and so many other areas. To date, Generative Pre-Trained Transformer entities have been developed, such as ChatGPT, which we will discuss extensively in one of the following chapters.

With these, it is possible to interact directly, and they can understand, summarize, generate and predict content. Based on this premise, the idea of the project proposed in the article is to be able to interact with artificial intelligence from a platform such as the Raspberry Pi.

Figure 1:
The Raspberry Pi 4 (4 GB) board
used for this project.

The latter, equipped only with a microphone, headphones or PC speakers, always connected to the Internet, will act as a bridge to AI.

Hardware

The hardware for this project is simple and is divided into three main components. The first, which is also the most important, is the base on which we will build the entire system: the Raspberry Pi 4. This model has 4 GB of RAM and an updated system capable of supporting advanced audio and video processing.

This represents the fourth generation produced by the Raspberry Foundation and is not only suitable for those who want to learn to program, develop, and play games, but thanks to numerous pins it is also particularly suitable for complex IOT projects.

Equipped with an ARM processor, it offers performance comparable to that of an entry-level PC, but is also capable of handling 4K video streams. Now considered a cornerstone for those embarking on this type of project, it offers numerous devices that can be combined such as camera modules, sensors, GPS, LCDs and many more.

Furthermore, it makes available numerous pins, both 3.3 V power supply and to be able to interface peripherals, relays or sensors of different nature and effortlessly manageable at the Python or C++ code level through a set of libraries made available by the manufacturer (**Figure 1**).

As an operating system, we definitely recommend Raspbian OS 64 bit so that no surprises can be expected after installing the various packages. Along with this platform, we have provided a USB microphone, which is optimal for our system as well as for PCs and MACs. This microphone offers a frequency response of 100 Hz to 16 kHz and a sensitivity of -67 dBV/ μ Bar, -47 dBV/Pascal (± 4 dB).

Two of the main advantages of this choice are the suppression of unwanted background noise and simple installation, thanks to plug and play functionality that eliminates the need for external drivers.

Last, but not least, as a third and final component, we have provided a pair of PC speakers or simple headphones, which we will connect to the audio jack on our Raspberry Pi.

Language

The programming language used is Python: object-oriented, it can be used for many types of software development precisely because of its ease and integration with other types of languages. Clear and powerful, Python is comparable to Java or Perl, which, like them, in fact, uses an elegant syntax allowing programs to be read easily and thus create complex projects without sacrificing maintainability.

Today, according to several surveys, it is used by as many as 84% of developers as their primary programming language. Among the most widely used areas, we find Data analysis and Web development in the lead but not only, in fact, other areas such as machine learning, computer graphics and game development are beginning to emerge.

Python — although not as optimized as C++, mainly in terms of RAM and processor usage — was chosen because it is open source, with a huge number of libraries also available at no extra cost, and is portable to all kinds of OS platforms (Linux, Mac, Windows).

In our case, in fact, it will run on Raspbian, an OS based on the official Debian releases adapted for an ARM architecture, but it could easily be run on Windows without having to make too many changes. The most widely used version of Python is 3.6 and is often used with IDEs/Editors such as PyCharm Professional/Community Edition or VS Code.

Given the small number of lines we are going to write, it is recommended to install an editor only if you are a novice; otherwise, you can safely use Nano, VI or Text Editor preinstalled on Raspberry Pi.

ChatGPT

But what exactly is ChatGPT? ChatGPT, where GPT stands for Generative Pre-trained Transformer, is an enhanced digital assistant that uses artificial intelligence (AI) to provide a natural conversational experience. Technically, it is a Large Language Model (LLM), a generative artificial intelligence algorithm that uses deep learning techniques and huge datasets to understand, summarize, generate and predict content.

OpenAI's product, developed in 2015 in San Francisco, has been at the center of widespread media hype for several weeks, both because of the blocking of the content creation software in Italy, which was later reactivated after it was brought into compliance with European privacy regulations, and because of the exit and subsequent return of CEO Sam Altman, events that further fueled attention and discussion around OpenAI and its strategic decisions.

But what exactly does generative artificial intelligence and the large language model mean? Generative artificial intelligence is a branch of AI that focuses on creating models that can autonomously generate text, images and other types of data. GPT represents an example of a transformer-based generative AI model that uses supervised learning on a large corpus of text to generate coherent and meaningful content. It can write content that can be used in numerous contexts, perform automatic translation, provide information and generate real-time responses that can help those working in communications and marketing, but not only that.

The reach of ChatGPT is much broader. Unlike a search engine or voice assistant, this tool is based on artificial intelligence that uses a process called pre-training to understand the context of a conversation and generate responses. The software has been "trained" by feeding it a massive amount of text from books, articles, websites, and human-to-human dialogues.

Significant advantages with this approach certainly include, given the premise, broad knowledge, speed of response, and ease of access and use through a user-friendly interface. ChatGPT can be used in several areas to improve user interaction and automate certain tasks.

Some of these may include the following:

- **Financial services:** can provide virtual assistance in the financial sector, such as answering questions about accounts, providing basic investment advice, or helping with personal financial planning;
- **Health care:** can support health care professionals by providing basic information on symptoms, answers to common questions, advice on managing certain medical conditions, or suggestions for a healthy lifestyle;
- **Education and training:** can be used as a virtual tutor to provide detailed explanations of academic or professional topics, answer student questions, offer practice exercises, or create interactive learning scenarios;
- **Customer support services:** can automate and improve customer support services by providing immediate answers to frequently asked questions, support in solving technical problems, or providing information about products or services;
- **Travel and tourism sector:** can help with travel planning by providing information on flights, hotels, tourist attractions and restaurants, suggesting customized itineraries or assisting in booking tickets and accommodations.

Some significant companies such as Mastercard, KLM, or Spotify have already entered into agreements with this product and its company, for example, to be able to implement Customer Care service or in the case of Mastercard to provide support in handling financial transactions and answering questions related to credit card security.

Services and APIs

For ChatGPT, in addition to the classic interfacing from a Web UI (User Interface), there is also a mode served through REST API, which is more widely used by companies such as those mentioned above. An API, or Application Programming Interface, consists of a set of rules that determines how applications or devices can connect and communicate with each other. A REST API is an API that conforms to REST, or representational state transfer, architectural style design principles. For this reason, REST APIs are sometimes referred to as RESTful APIs.

First introduced in 2000 by computer scientist Roy Fielding in his doctoral dissertation, REST provides developers with a relatively high level of flexibility and freedom. This flexibility is just one reason why REST APIs turn out to be the most popular method for connecting components and applications in a microservice architecture.

Basically, an API is a procedure that allows one application or service to access a resource within another application or service. The application or service making the access is called *client*, and the application or service that contains the resource is called *server*.

REST APIs communicate via HTTP requests to perform standard database functions such as creating, reading, updating and deleting records (a set of operations also known as CRUD, which stands for "create, read, update and delete") within a resource. For example, a

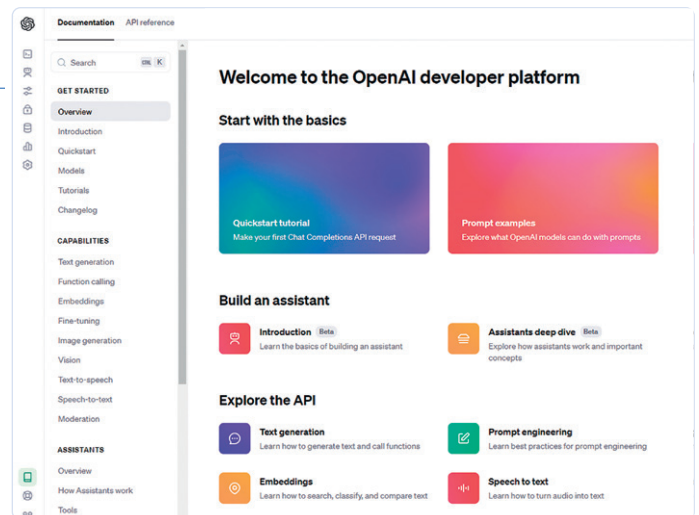


Figure 2: The OpenAI web page dedicated to developers.

REST API will use a GET request to retrieve a record, a POST request to create it, a PUT request to update it, and a DELETE request to delete it. All HTTP methods can be used in API calls.

An optimally designed API is similar to a website running on a web browser with built-in HTTP capabilities. They can be divided between GET and POST, whose main difference lies in the fact that in the former a request is sent where the data is visible and entirely in the called URL, while in the latter it is in a body and often hidden to the untrained eye.

To interface with our service, we will use exactly such a POST method. By connecting to the web page [1], after login, we can get an overview of the services offered, capabilities, through this interface (Figure 2). As you can see, there are APIs for text generation, prompt engineering, embeddings, speech-to-text, image generation, fine-tuning, text-to-speech and for vision.

The first step to be able to use their services is definitely to create a new key. From the *Overview* [1], you can click on *Dashboard*, top right, and then again on *API Keys* in the side menu that will present itself. Next, *Create new secret key* (Figure 3).

We leave all the settings as they are and put then the permissions on *All*, add the name we want to give to the API key and copy the string it will give us once we press *Create*. This string will no longer be visible, so it is imperative to save it immediately to one of our notepads or similar (Figure 4).

Now, the sore point: OpenAI initially gave a kind of free premium period in which it made \$5 available, but recently this policy was changed. For this reason, it will be necessary to enter the *Usage* section, again, from the menu on the left. Here on the right side of the page we find *Increase limit* and follow the directions to link your credit card and buy some credit (Figure 5). Even as little as €5 is sufficient for educational purposes, since each request requires very little in economic terms.

OS Configuration

We will now install the operating system inside our Raspberry Pi. Two things are needed to accomplish this step:

- MicroSD card with at least 16 to 32 GB
- Raspberry Pi Imager

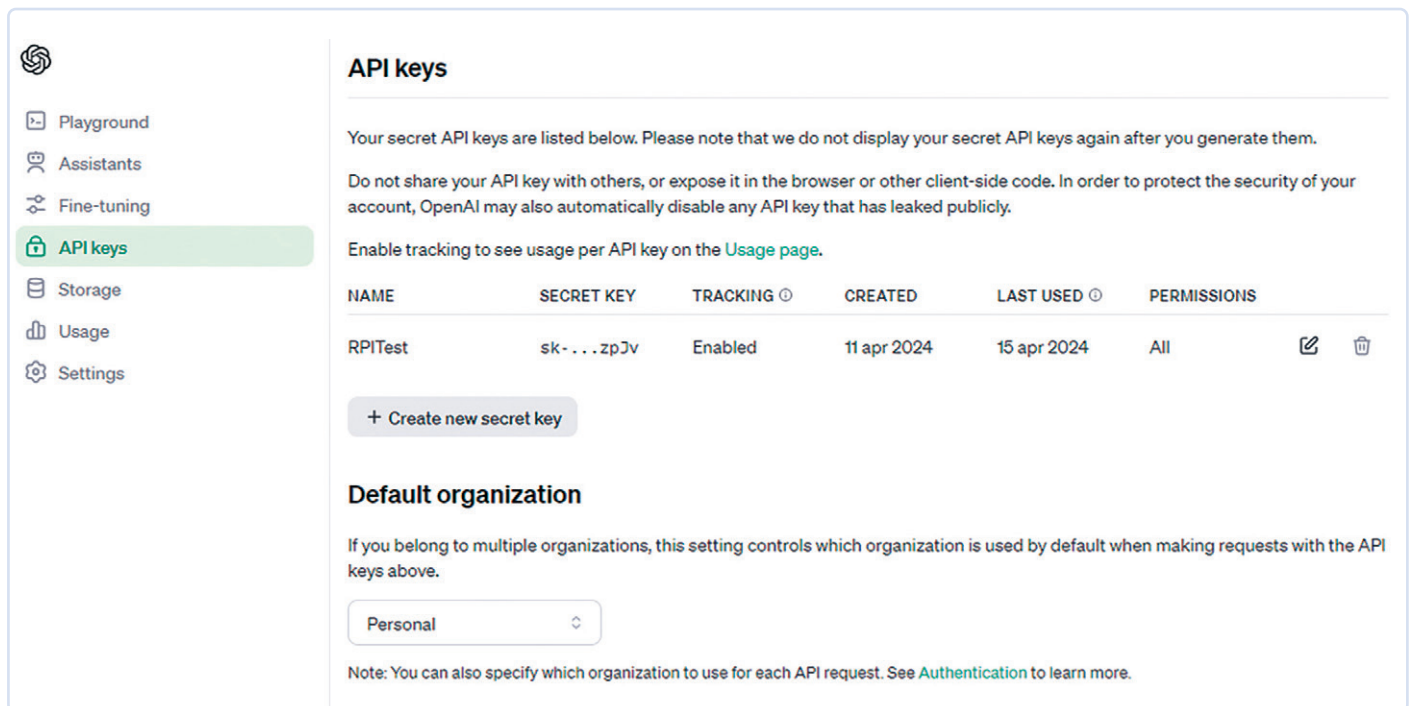


Figure 3: Page for creating the key, in order to interact.

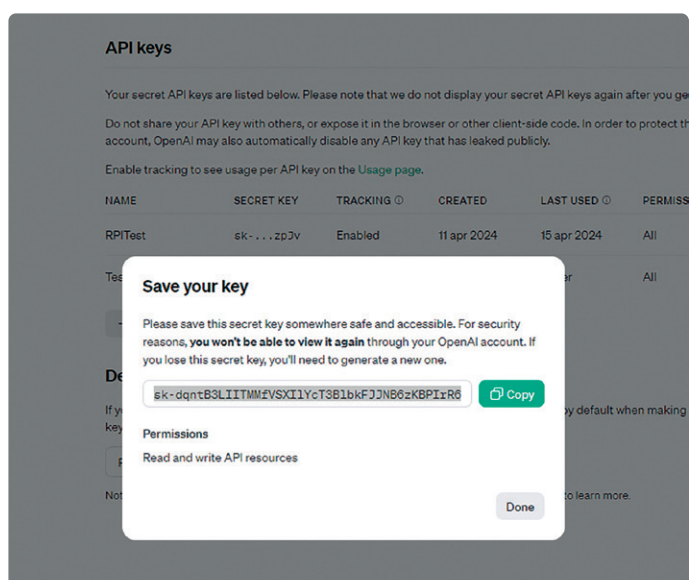


Figure 4: Save your key for future use.

For the last requirement, the reference website is at [2]. In the first section, we see *Imager*, which is the tool needed to create a bootable SD card, capable of booting the device. Further on, still within the site, we find the zipped file that we are going to upload. This is the direct link to download the package [3]; be aware that's a 1.2 GB file that might take time with not-so-fast internet connections.

Once Imager is installed, we open it, insert the SD card, and click **CHOOSE OS (Figure 6)**. Here we select the last item, i.e. *Custom OS* and then choose the zipped file downloaded earlier.

Next, we click on **CHOOSE SD CARD** and select the drive on which we wish to restore the OS and finally **WRITE**. We just need to remember to change the default settings so that we can log in for the first time with custom credentials.

In our case, as can be seen from the screenshot in **Figure 7** we chose username *daniel* and password *12345678*. After a few moments, we have our SD card and are ready to go with our device.

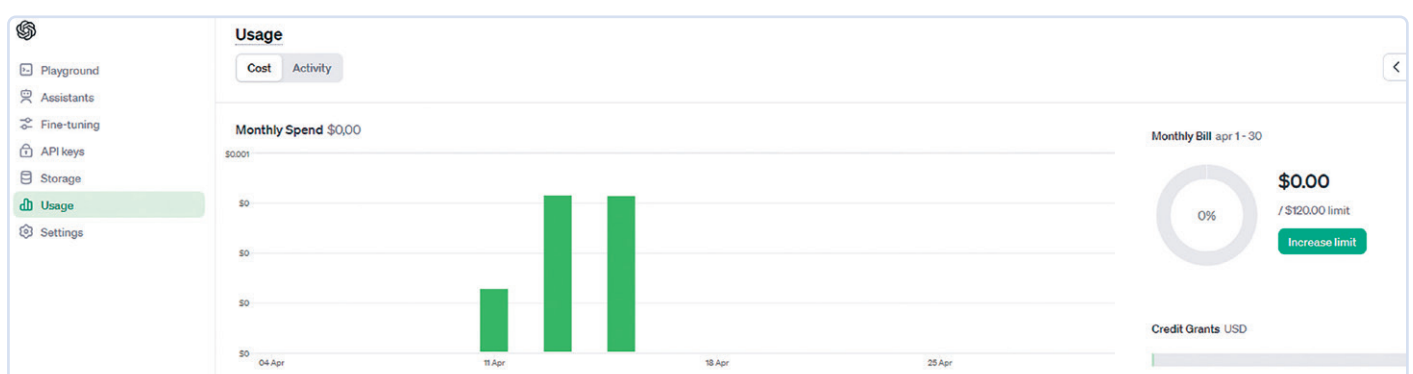


Figure 5: Credit is required to use the service.



Figure 6: The Raspberry Pi Imager screenshot.

Let's insert the card inside our Raspberry Pi, connect it to a network with Internet access and turn it on. It will be very useful to be able to install a remote desktop so that we can use our RPI without necessarily getting an additional monitor. Then, we install the remote desktop by following these steps:

```
sudo apt-get install xrdp -y
sudo apt-get update
sudo apt-get upgrade
```

Now we have all the tools to communicate with the machine remotely, and so we can forget about the monitor and act directly from our pc, your choice. We install what are the fundamental libraries for development: *python* to be able to compile the project and *libsox* that allows us to interface with the microphone:

```
sudo apt install python3 python3-pip python3-venv
sudo apt-get install sox libsox-fmt-mp3
```

We create a working folder in which we will install a virtual environment, which we will call *env* so that we can install the necessary Python libraries without compromising the entire system.

```
mkdir openai
cd openai
python -m venv env
```

We now access the environment through the command

```
source env/bin/activate
```

Bear in mind that *env* is a name we set up, we are not bound to it at all, at the previous step we could have also called it *env_openai*. Once activated, we install the Python libraries, in detail then:

Google Speech to be able to interact with the Google cloud and succeed in a high-level Speech to text that will allow us to extract text from speech:

```
pip install google-speech
```

SpeechRecognition, to be able to access the USB microphone, without having to deal with handling bits, calibration of it or other low-level subtleties:

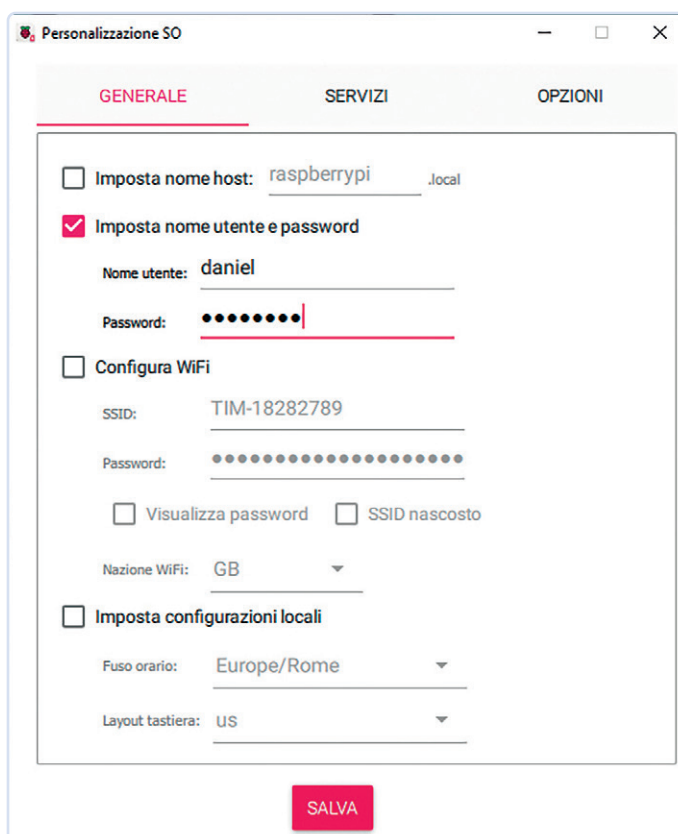


Figure 7: You will need to change your credentials the first time you log in.

```
pip install SpeechRecognition
pip install PyAudio
```

OpenAI to be able to interact with the API described above and get answers from our artificial intelligence:

```
python3 -m pip install openai
```

We do not close the console at this point, since it will be essential later to be able to execute the code. In case it is closed, to re-enter the properly configured space we will simply enter the correct folder (first command below) and type the following:

```
cd openai
python -m venv env
source env/bin/activate
```

Code

Top-down or class-based programming? The first option, although faster and more intuitive, may not be the correct choice for several reasons. In fact, object-oriented programming (OOP) is based on the idea of building a set of objects that can interact with each other by exchanging messages, but each maintains its own state and data just as it does in the real world.

"Message exchange" refers to the ability of objects to call the public methods of other objects, for example by passing them data to process and receiving the result of their processing. Unlike procedural programming, which is a programming style that consists of creating blocks of code (called subprograms), in object-oriented programming it is possible to centralize multiple features within a single class that encapsulates variables and functions.

Other advantages of object-oriented programming include:

- natural support for software modeling of real-world objects or the abstract model to be reproduced
- easier management and maintenance of large projects
- the organization of code in the form of classes encourages modularity and code reuse

That being said, let us move on to writing the actual code, and create the file that we will implement with the instructions to be able to interact with the AI. To achieve this, we type from the console we are in:

```
touch gpt.py
```

Next a file is created; we open it with the default editor of the operating system, which in our case, if we are accessing with GUI or remote desktop, is Geany Programmer's editor.

As in all Python projects, references to the libraries we will use must be inserted:

```
import speech_recognition as sr
from openai import OpenAI
from google_speech import Speech
```

For teaching purposes, we will keep all the classes, in this case only one, within the same file as the main and the code that will do the work. We then define the `OpenAICustom` class, which will have a few constants: the AI model we are going to invoke, the language in which we communicate and want responses, and a last one that allows us to control the output:

```
class OpenAICustom:
    GPT_MODEL = "gpt-3.5-turbo"
    GPT_LANG = "en"
    isDebug = True
```

The constructor takes as input the pointer to the microphone and the string containing the key to the API and the ChatGPT world. Within this, we are going to initialize the entities involved and the message that we will then exchange and add the various requests to:

```
def __init__(self, sr, keys):
    self.recognizer_instance = sr.Recognizer()
    self.key = keys
    self.client = OpenAI(
        api_key = keys
    )
    self.messages = [
        {
            "role": "system",
            "content":
                "You are a helpful assistant"
        }
    ]
```

We add a method to export one of the instances created outside the class. In detail, we discuss the entity that handles the microphone:

```
def getReco(self):
    return self.recognizer_instance
```

Certainly among the indispensable functions to be had in this class we have the one responsible for communicating with the ChatGPT API, which we have already mentioned extensively.

This one will have to take message input and leverage the objects already initialized in the constructor. As a first action, we are going to add to our message array a new element that will contain a `json` expression consisting of the user's role and content.

Next we create the real request to the OpenAI servers which will give us an object in response that we will save in the `chat` variable. This will have the following tree:

```
{
  "choices": [
    {
      "finish_reason": "stop",
      "index": 0,
      "message": {
        "content": "RISPOSTA DI CHATGPT",
        "role": "assistant"
      },
      "logprobs": null
    }
  ],
  "created": 1677664795,
  "id": "chatcmpl-7QyqpwdfhqwajicIEznoc6Q47XAyW",
  "model": "gpt-3.5-turbo",
  "object": "chat.completion",
  "usage": {
    "completion_tokens": 17,
    "prompt_tokens": 57,
    "total_tokens": 74
  }
}
```

As you can see, the message is contained in the first object of the `choices` array. We may or may not print it within our debug console and then feed it to the function responsible for text to speech:

```
def sendToAI(self, message):
    self.messages.append(
        {
            "role": "user",
            "content": message
        }
    )
    chat = self.client.chat.completions.create(
        messages=self.messages,
```

```

        model=self.GPT_MODEL
    )
    reply = chat.choices[0].message
    if self.isDebug:
        print("Assistant: ", reply.content)
    self.messages.append(reply)
    speech = Speech(reply.content, self.GPT_LANG)
    speech.play()

```

Once we have properly constructed the class on which all the code will pivot, we go on to create the `main`, which is what will actually call and interact with the user. We define, in the `SYS_LANG` variable, the language for which we intend to recognize speech.

Next we take the OpenAI key as input and then put it in a continuous loop where we continue to listen, understand the user and respond accordingly:

```

SYS_LANG = "it-IT"
in_key = input("Enter your OpenAI key: ")
myObj = OpenAICustom(sr, in_key)
while True:
    with sr.Microphone() as source:
        myObj.getReco().adjust_for_ambient_noise(source)
        print("I am listening... go ahead and talk!")
        audio = myObj.getReco().listen(source)
        print("Ok! I am now processing the message!")
        try:
            text = myObj.getReco().
                recognize_google(audio, language=SYS_LANG)
            print("Google understood: \n", text)
            myObj.sendToAI(text)
        except Exception as e:
            print(e)

```

Before running the code, let's remember the premise made at the end of the operating system configuration chapter, that is, let's make sure we are inside the virtual space created with all the libraries. Finally, to launch the program inside the console, type the command:

```
python3 gpt.py
```

Future Developments

Certainly, one of the closest future developments to the proposed project is to miniaturize it to further contain cost and power consumption. In fact, portability to a device such as the ESP32 could add value and have numerous applications.

With this in mind, there could be two types of approaches: a first, much more lite, which sees the ESP32 only as a gateway bridge to convey information while all processing is delegated to a server or an app. In this first case, certainly feasible, it involves little computational load but dual development and a non-autonomous chip.

A second approach, on the other hand, which is much more complete, involves putting all parts on chip, a first receiving part through a microphone, a speech processing part to be able to do a Speech to text properly, and finally direct interaction with the ChatGPT API.

The biggest issue we may encounter will definitely be with the speech-to-text processing. This, in fact, could saturate the memory of the chip, since there will need to be a conversion to base64 or similar. Interaction with the ChatGPT engine, according to some rumors leaked on the official site, could also be possible through IFTTT, thus completely IoT-oriented.

The code to implement this voice assistant can be downloaded at the Elektor Labs page for this project at [4]. ◀

240619-01

Questions or Comments?

Do you have technical questions or comments about this article? Please write to the editorial team of Elektor at editor@elektor.com.



Related Product

➤ **Raspberry Pi 4 (4 GB) Official Starter Kit**
www.elektor.com/20556

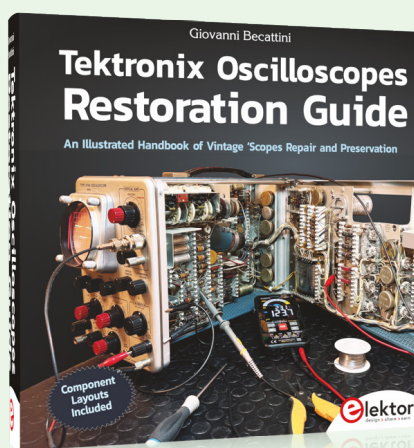


WEB LINKS

- [1] OpenAI platform developer page: <https://platform.openai.com/docs/overview>
- [2] Raspberry Pi download of Imager tool: <http://www.raspberrypi.org/downloads>
- [3] Raspberry Pi OS download: http://downloads.raspberrypi.org/raspbian_latest
- [4] Elektor Labs webpage for this project: <https://tinyurl.com/rsvj5ken>

Tektronix Oscilloscopes Restoration Guide

Tektronix oscilloscopes are true masterpieces of electronics and have helped mankind advance in every field of science, wherever a physical phenomenon needed to be observed and studied. They helped man reach the moon, find the cause of plane crashes, and paved the way for thousands of other discoveries.



Price: €59.95

Member Price: €53.96

www.elektor.com/21051

SunFounder GalaxyRVR Mars Rover Kit for Arduino

The SunFounder GalaxyRVR Mars Rover Kit was designed to mimic the functionality of real Mars rovers, it offers a hands-on experience that's both educational and exciting. Compatible with Uno R3, the GalaxyRVR is equipped to navigate diverse terrains with ease.

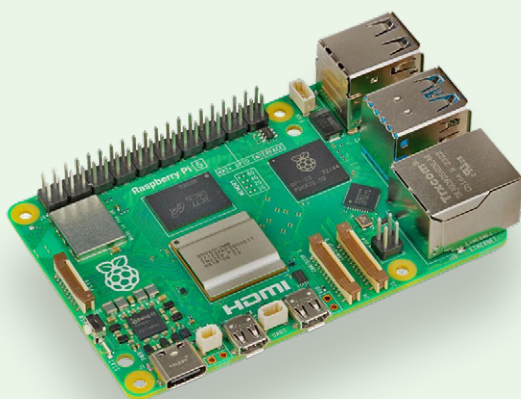


Price: €134.95

Member Price: €121.46

www.elektor.com/21061

Raspberry Pi 5 (16 GB RAM)



Price: €139.95

www.elektor.com/21080

LILYGO T-Panel S3 Development Board



Price: €89.95

Member Price: €80.96

www.elektor.com/21026

The Connected Autonomous Vehicle and Its Environment

An Introduction to Real and Reduced-Scale Autonomous Vehicles

Source: Adobe Stock

By Jacques Ehrlich (France)

With global connectivity on the rise and network latencies dropping to a few milliseconds, self-driving vehicles are rapidly becoming CAVs (connected autonomous vehicles) in the universal ecosystem of the Internet of Things. In this introduction to his forthcoming Elektor book, Jacques Ehrlich introduces us to the terminology related to autonomous driving and explains how the technology is evolving in these vehicles, in parallel with the development of the infrastructure that hosts them.

Editor's Note: This article is an excerpt from the Elektor book, *The Connected Autonomous Vehicle and Its Environment*, which will be published by Elektor in 2025. It was formatted and lightly edited to match Elektor Mag's conventions and page layout. The author and editor are happy to help with queries. Contact details are in the **Questions or Comments?** box.

The autonomous vehicle is a complex and ever-evolving technology, and the aim of this book is to delve into its many facets. For too long, the vehicle was seen as

an isolated entity — disconnected from other vehicles and from its surrounding environment — namely, the infrastructure.

This isolationist view is entirely outdated. Modern vehicles, equipped with advanced driving assistance systems (ADAS), and autonomous vehicles, are becoming highly cooperative systems. In the near future, they will exchange information with each other (V2V) and the infrastructure (V2I) continuously via various communication channels, integrating into the intelligent transport system (ITS) seamlessly.

Intelligent Transportation System

ITS can be defined as a system integrating all communication, control and information processing technologies to provide users of the transport system with safe and efficient mobility services and road operators with decision-making tools to optimize road network operation. The expected benefits are to save lives, time, energy, and money, and preserve the environment.

Figure 1 illustrates an architectural framework inspired by USA Architecture. Many countries have developed similar frameworks, such as KAREN in Europe and ACTIF in France. Typically, these architectures consist of four main subsystems: traffic management centers (TMC), vehicles, road infrastructure, and users.

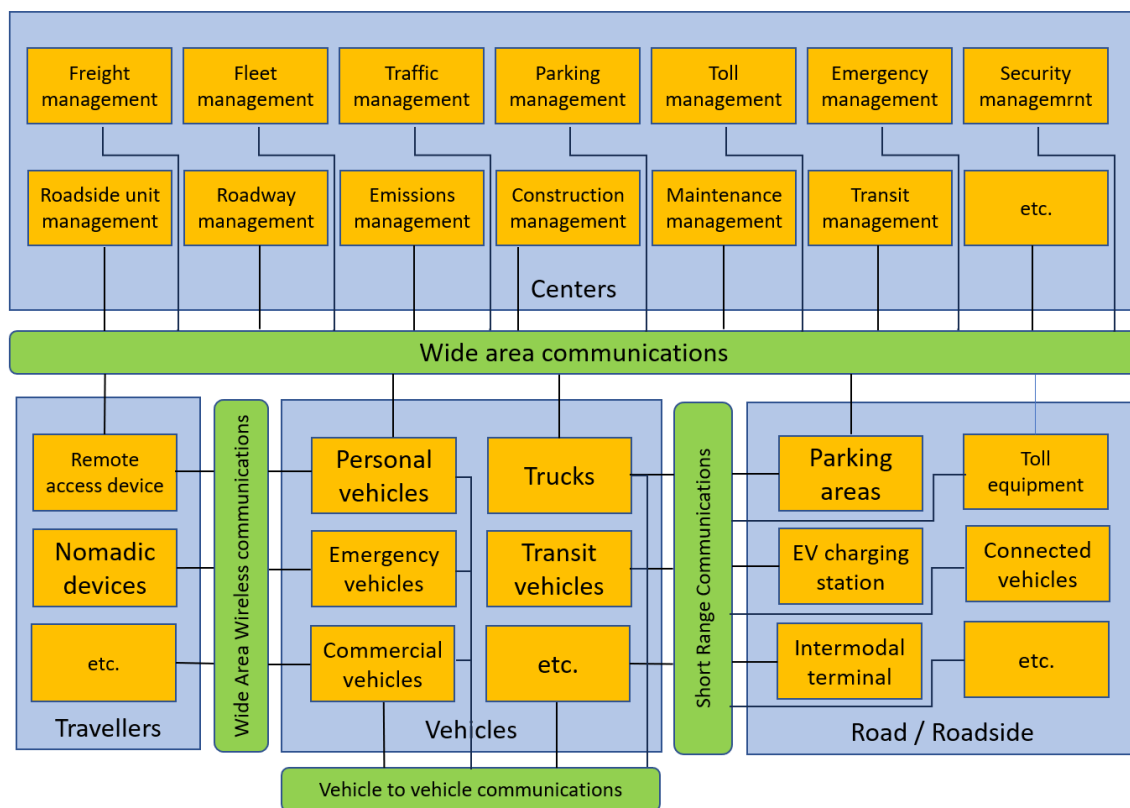


Figure 1: Framework architecture for intelligent transportation systems (inspired from the USA Arc-IT 9.2 architecture).

Each subsystem is made up of a wide range of services. These subsystems are cooperative, exchanging data through various communication methods — whether short, medium, long-range, wireless, or wired.

The overarching architecture is designed to foster the interoperable integration of services and optimize the sharing of common resources. The autonomous vehicle is a crucial component within this architectural framework.

Driving Assistance and Autonomous Vehicles

The autonomous vehicle is not a sudden revolution, but rather the natural outcome of a long evolution. It began with driving assistance systems such as ABS and ESP, and, more notably, advanced driver assistance systems (ADAS), many of which are now widely available. These ADAS serve as the foundational building blocks of connected and autonomous vehicles (CAVs), with higher-level autonomous functions built on top of them. Before delving into autonomous vehicles, this book will offer a brief overview of ADAS, whose evolutionary path is illustrated in **Figure 2**.

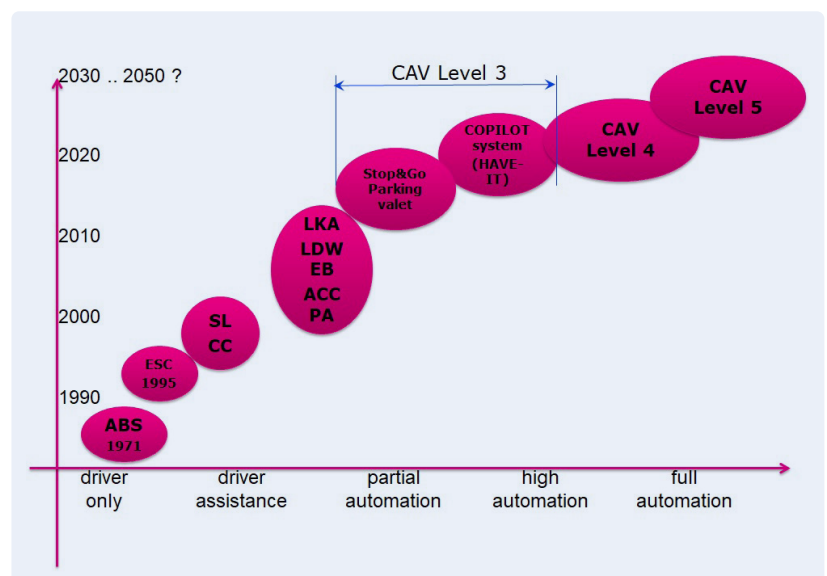
However, it would be a mistake to assume that an autonomous vehicle is merely the sum of its ADAS components. What truly sets the autonomous vehicle apart is its ability to plan a route from origin to destina-

tion, handling all obstacles along the way while, ideally, performing more efficiently than a human driver. This is an extraordinarily complex challenge — one that not all autonomous vehicles have yet fully mastered.

Classification of Autonomous Vehicles

As a general definition, we can say that in an autonomous vehicle, all or part of the driving task is delegated to onboard automation systems. In certain situations,

Figure 2: Driving assistance from the 1990s to the present.



Level	Mode Definition	Vehicle Control	Environment Monitoring	Fallback Performance	Contexts
0	No Automation	Human	Human	Human	N/A
1	Driving assistance: actions on steering system, brakes or accelerator in some circumstances	Human and System	Human	Human	Limited
2	Partial automation: actions on steering system, brakes or accelerator in some circumstances	System	Human	Human	Limited
3	Conditional automation: automatic driving, but the driver must be ready to take control at any time	System	System	Human	Limited
4	High automation: as above, even if the driver is not ready to regain control	System	System	System	Limited
5	Full automation: as above, but in all situations	System	System	System	All



Figure 3: The five levels of autonomy.

driver intervention may still be required, sometimes for extended periods. However, this definition remains too vague, which led to a more formalized framework proposed in 2014 through the SAE J3016 standard, which has since gained widespread acceptance.

The table in **Figure 3** illustrates the different automation levels, where definitions and terms were derived from a SAE International document, downloadable for free at [1].

In this work, we will explore five levels of autonomy, each defined by the vehicle's capacity to replace the human

driver in key tasks: lateral and longitudinal control, environmental monitoring, and failure management, within both limited and unrestricted road contexts.

Driving Task Analysis

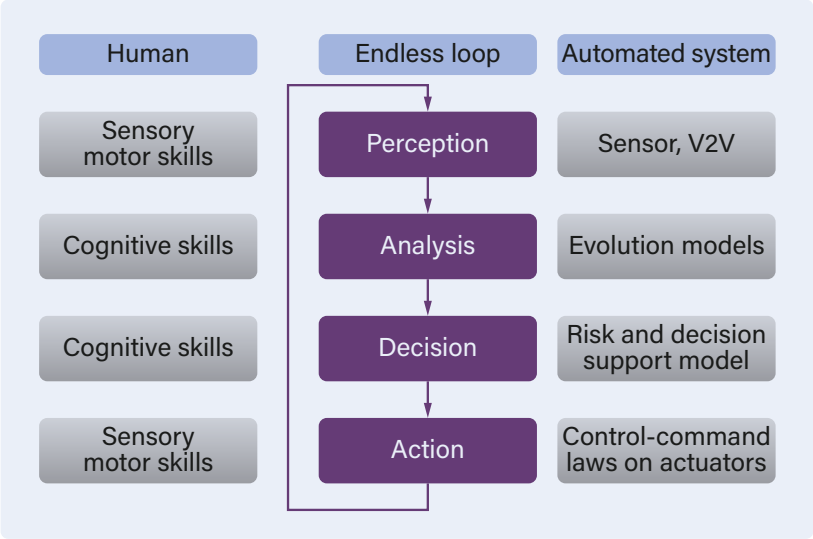
Understanding the basics of CAV means first understanding how humans drive because autonomous vehicles, as well as the driving assistance systems, are inspired by humans. We will start with a small case study: a vehicle on the highway facing a potential accident situation. The book will highlight the duality between the human model and the vehicle model.

On the one hand, the driver uses his cognitive and sensory motor skills, while on the other, the autonomy of the vehicle is based on algorithms, sensors, and actuators. All these capabilities revolve around a four-step model: perception, analysis, decision, and action. **Figure 4** shows the similarities between the actions performed by the driver and the vehicle's automation. Depending on the levels of automation, all or part of these actions can be supported by one or the other, or can be shared between the two.

The Seven Key Functions of the Autonomous Vehicle

The autonomous vehicle is based on - what we call in this book — the seven key functions of the autonomous vehicle. This is not an official taxonomy, but a proposal by the author.

Figure 4: Modeling the driving task.



1. Macroscopic localization and mapping

Macroscopic localization is of metric precision. It allows the vehicle to be positioned on the road or even on a lane in the road. It is based mainly on GNSS satellite systems (e.g., GPS, GLONASS, GALILEO, etc.). However, satellite localization may be faulty in tunnels or any areas where the signals emitted by satellites are masked. This is why macroscopic localization is the result of the fusion between satellite localization and trajectory estimation based on data delivered by inertial and distance sensors (**Figure 5**). The precision obtained is not sufficient to locate the vehicle on a map. We will examine the *map-matching* technique, consisting of projecting the estimated position into a cartographic reference, made up of segments representing portions of the road (**Figure 6**).

2. Microscopic localization

Microscopic localization is essential for lateral trajectory control in both ADAS and autonomous vehicles. Much like how trams and trains are guided by rails, autonomous vehicles rely on their own “rails:” the road markings. These markings are detected by a front-facing camera, which — combined with image processing techniques and road-curvature modeling — enables the vehicle to position itself within the lane, with centimeter-level accuracy (see **Figure 7**).

3. Obstacle detection

ADAS such as collision avoidance or mitigation, emergency braking, or distance headway control are based on obstacle detection. Obstacle detection relies mainly on three types of sensors: radar, laser scanner (lidar), and stereo vision using two cameras, and data fusion between these three sources. Stereo vision and data fusion techniques will be explored as well as a focus on different kinds of radar (long-, medium-, and short-range).

4. Vehicle dynamics

Knowledge of the dynamic state of the vehicle is necessary to ensure that it remains within its controllability domain. In the case of ADAS, this will make it possible to initiate corrective actions on the trajectory when the vehicle approaches the limits of this domain; in the case of the CAV, it will be an issue of defining control-command laws that remain within the controllability domain. The study of dynamics is based on models, the simplest of which is commonly used by researchers is the “bicycle” model (**Figure 8**).

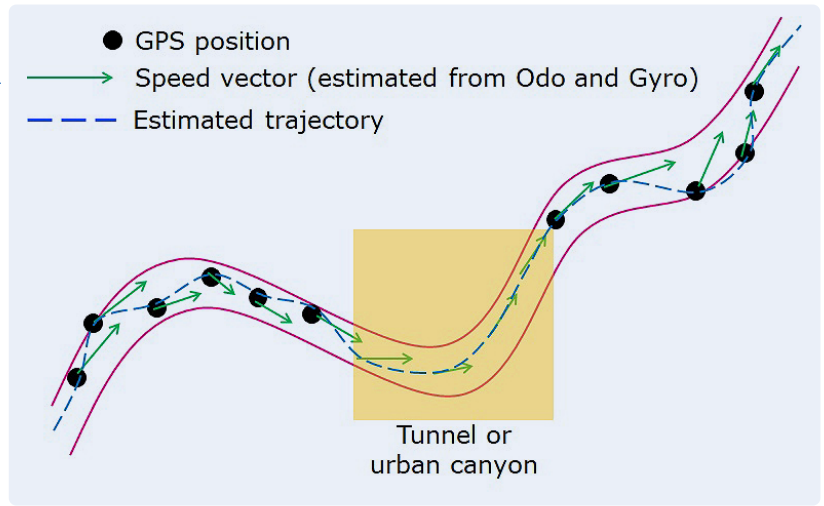


Figure 5: Positioning thanks to GNSS and dead-reckoning techniques.

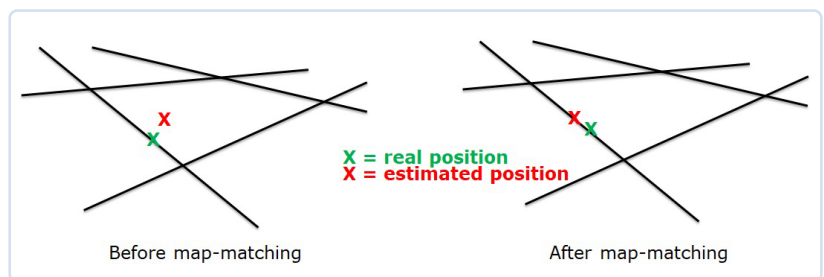


Figure 6: Map matching.

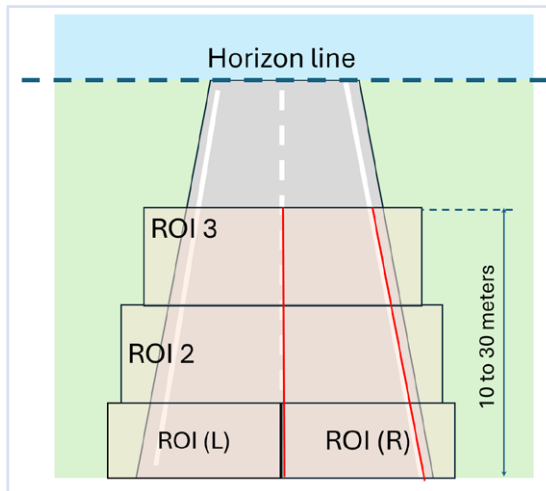


Figure 7: Lane marking detection in the regions of interest (ROI).

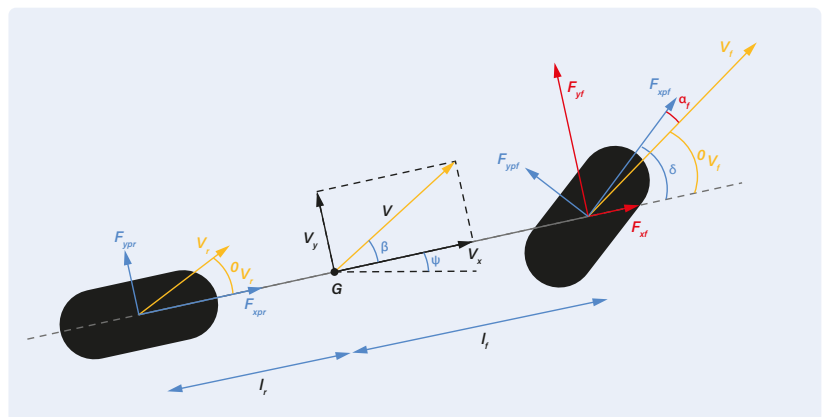


Figure 8: The bike model.

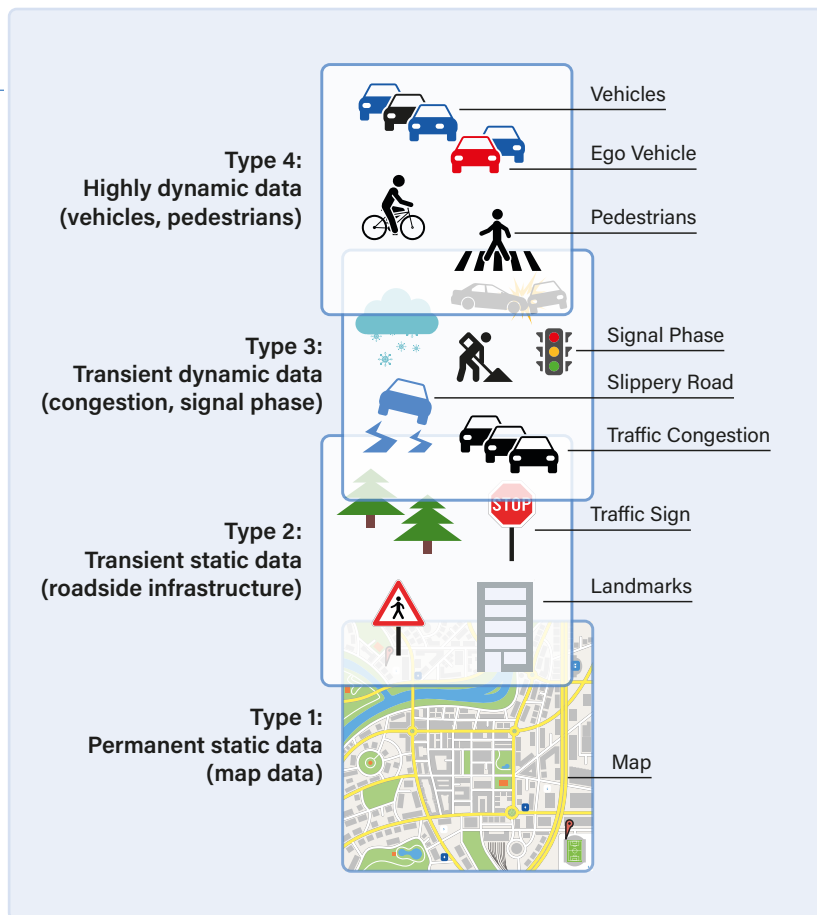


Figure 9: Local dynamic map (LDM) representation.

5. Situational awareness

To operate safely, the autonomous vehicle must have exhaustive knowledge of its environment. This is done using a local dynamic map (LDM). It is a digital map information system in which all road attributes and events are geo-referenced, whether slowly or rapidly evolving (**Figure 9**). This LDM allows the vehicle's situational awareness to be built. Connectivity plays a major role as information comes from various sources located close to or remotely from the vehicle.

6. Path planning

There are three levels of path planning that are commonly considered: the *strategic level*, which covers the entire journey from its origin to its destination, the *tactical level*, whose time span is between ten seconds and a few minutes, and the *operational level*, which concerns the immediate trajectory control thanks to action on the vehicle actuators.

7. Driver monitoring

One of the critical points of the CAV in levels 2, 3, and 4 is its interaction with the driver. In fact, in case of system failure or unexpected events, drivers may be mobilized for the manual recovery of the vehicle. Therefore, it is required that the driver is neither asleep nor distracted. The driver monitoring function provides real-time information on the driver's status, such as to safely manage the transitions from

automated to manual driving. We will also address the issue of minimal risk maneuvers (MRM) in case of no response from the driver.

Connectivity

To allow the autonomous vehicle to anticipate road difficulties, it is necessary to extend the perception of the environment beyond the range of its own perception sensors. This extended perception is achieved through bidirectional V2V, V2I, and I2V communications.

The communication architecture developed at the European level will also be considered in the book. V2X communications are based on the concept of C-ITS stations (**Figure 10**) which are the nodes of a communication network carried by vehicles, roadside units, traffic management centers, and users (pedestrians, cyclists, etc.). Each C-ITS platform can communicate via several communication vectors: 3G, 4G, 5G, 802.11p, Ethernet, etc.

Implementation

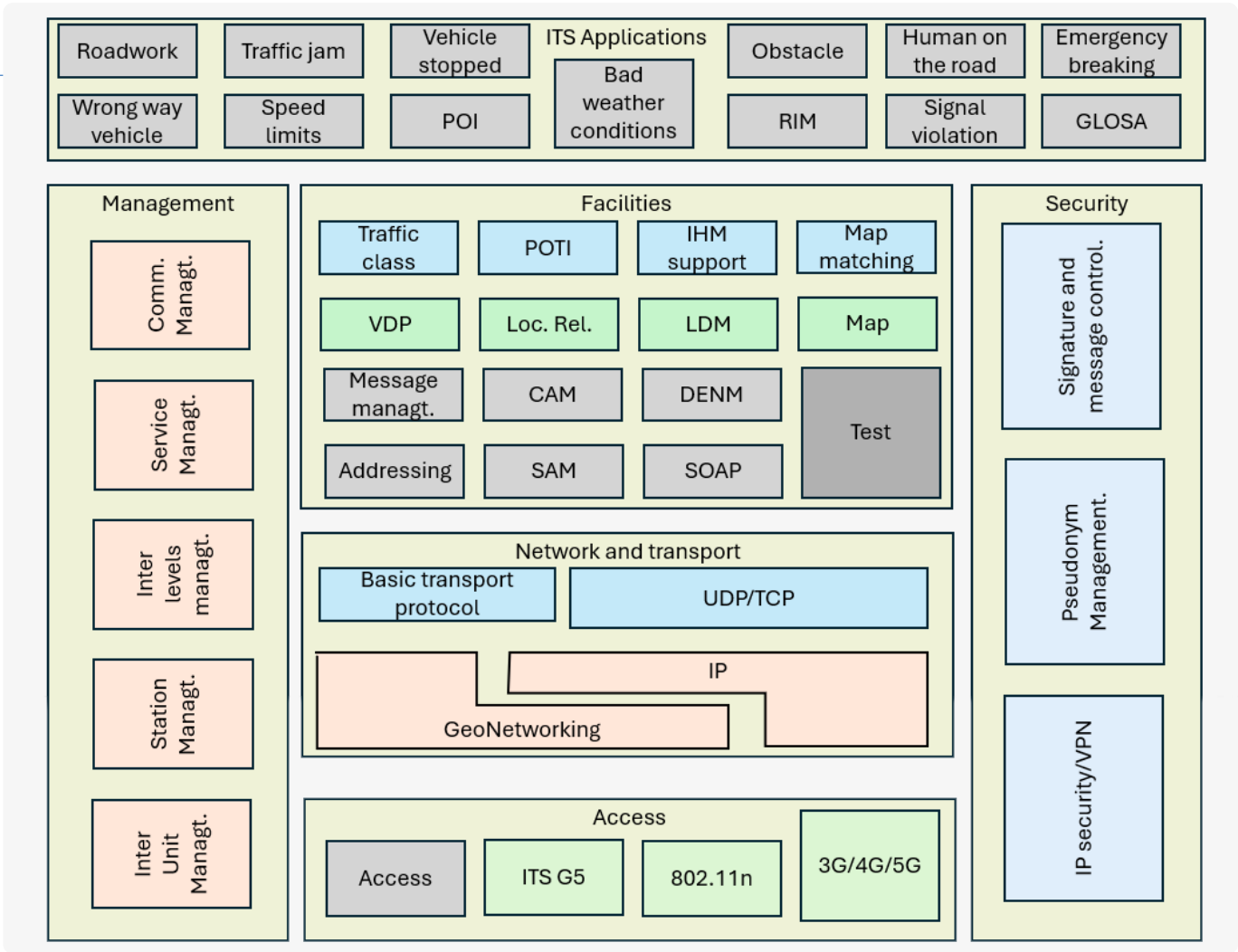
The key functions mentioned above are supported by on-board electronics and computing. In this book, we will deal with the electronic and computer architecture of the vehicle. The components of this architecture are actuators, sensors, computers (aka electronic control units, ECUs), and communication buses.

A major focus will be placed on computers and their execution framework, as well as the different communication buses, such as CAN, FlexRay, and LIN. Also, various architecture topologies will be investigated, from the simplest centralized ones to more complex, distributed architectures (**Figure 11**).

An important chapter of the book will be dedicated to the architecture design methodology. It is based on a top-down approach, starting from user needs and ending with a physical architecture through various intermediate stages. The method will be illustrated by the case study of a simple autonomous vehicle building block: the speed limiter.

Preliminary Hazard Analysis (PHA)

Automotive systems are safety-critical. Even a small hardware or software failure can lead to accidents with fatal consequences. We will investigate functional safety standard ISO26262, which almost all car manufacturers adhere to. The first step is the PHA, which consists of an analysis of the system behavior and its interaction with the environment under all possible circumstances.



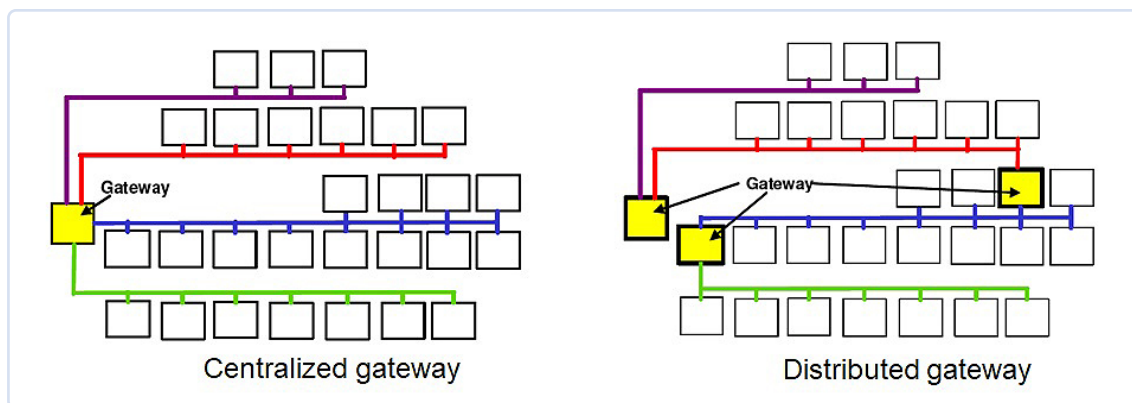
▲
Figure 10: Typical architecture of an ITS station.

The second step consists of estimating the risk level based on three features: severity, risk exposure and controllability, and then estimating an ASIL index (automotive software integrity level). Finally, we will see how “safety requirements” lead to the avoidance or minimization of risk down to acceptable levels.

The Infrastructure

For decades, the worlds of the automobile industry and road operators have been relatively compartmentalized, but with the development of telecom-

munications and the emergence of autonomous vehicles, car manufacturers and road network operators have understood the need for close cooperation. Nowadays, there is win-win cooperation, which allows for improvement of, on the one hand, the car efficiency and safety and, on the other, the road network operation. This cooperation relates to the concept of “floating car data” (FCD) also called probe vehicles. We will also introduce the recent concept of operational design domain (ODD) and infrastructure support for autonomous driving (ISAD).



◀
Figure 11: Centralized vs. distributed architecture topology.

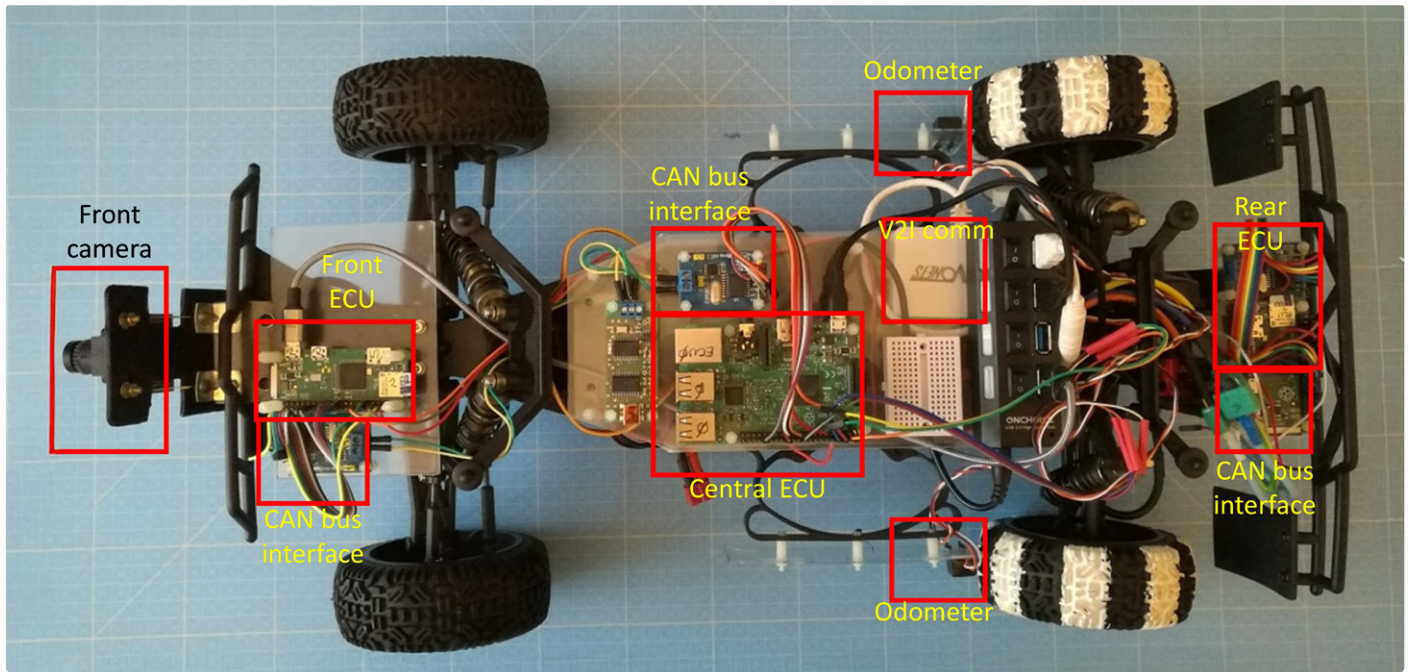



Figure 12: RS-CAV developed by the author. (Sponsored by University Gustave Eiffel, picture by J. Ehrlich).

RS-CAV: The Reduced-Scale Connected and Autonomous Vehicle

Have you ever dreamed of building your own CAV? A reduced-scale connected autonomous vehicle (RS-CAV), may be the answer to your wish. Indeed, in the last part of this book, you will find proposals to build a vehicle on a scale of 1/10, as the author of this book did (Figure 12).

However, you will not have a turnkey solution. Building an autonomous vehicle requires many skills in signal and image processing, control-command, telecommunications, etc. This project could be the start of a challenge between electronics and computer enthusiasts who could work as a team to achieve this goal: a fully operational, small-scale autonomous vehicle. It will be up to you to play... 

240537-01

Questions or Comments?

Do you have technical questions or comments about this article? You may write to Jacques Ehrlich at je-cav-book@orange.fr or Elektor's editorial team at editor@elektor.com.

About the Author

Jacques Ehrlich holds a PhD in electronics and telecommunications from Télécom ParisTech. At IFSTTAR, which later became Gustave Eiffel University, he co-directed, and later directed, a research laboratory specializing in advanced driving assistance systems and autonomous vehicles until his retirement in 2014. From 2014 to 2024, he was Research Director Emeritus. From 2012 to 2019, he also served as the chair of the international Technical Committee "Road Network Operation and ITS" at the World Road Association (PIARC). Currently, he coordinates a course on autonomous vehicles as part of the Smart Mobility Specialized Master's program (MS-SMOB) at Institut Polytechnique de Paris. A passionate enthusiast of electronics, computers, and glider flying, Jacques is also a certified flying instructor. He has authored several projects published in Elektor Labs, including GONOGO, CHADECHE, COBALT, and OUAH!.



Related Products

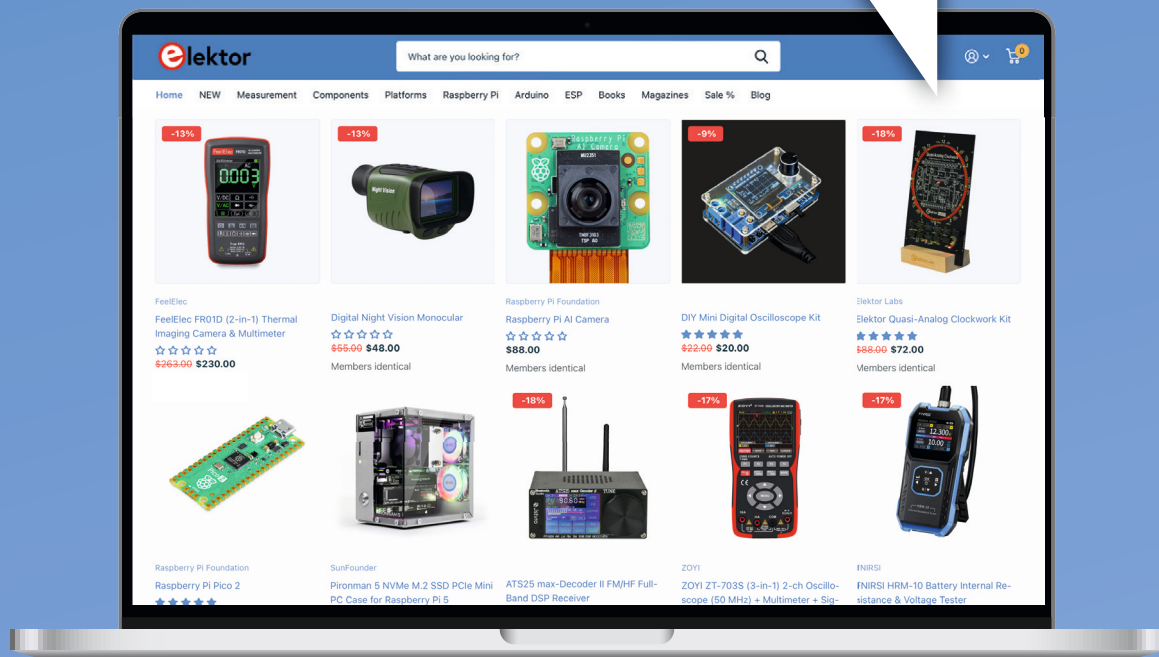
- > **Elektor Ultimate Sensor Kit**
www.elektor.com/19104
- > **Raspberry Pi 5 (4 GB RAM)**
www.elektor.com/20598



WEB LINKS

- [1] SAE International, "Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles," Revised 2021-04: https://sae.org/standards/content/j3016_202104/preview/
- [2] Yue Wang, Eam Khwang Teoh, Dinggang Shen, "Lane detection and tracking using B-Snake," Elsevier: <https://sciencedirect.com/science/article/abs/pii/S0262885603002105>
- [3] Intelligent Transport Systems (ITS) Communication Architecture, ETSI EN 302 665, ETSI [PDF]: <https://tinyurl.com/itsyspdf>

What's Your Opinion?



At Elektor, we offer more than just electronics – we create an experience of quality products and exceptional customer care, backed by the passion of our community.

Share your opinion on
www.elektor.com/pages/customer-reviews



★★★★★

Thanks for the great educational value.

★★★★★

You have been doing great. Keep it up.

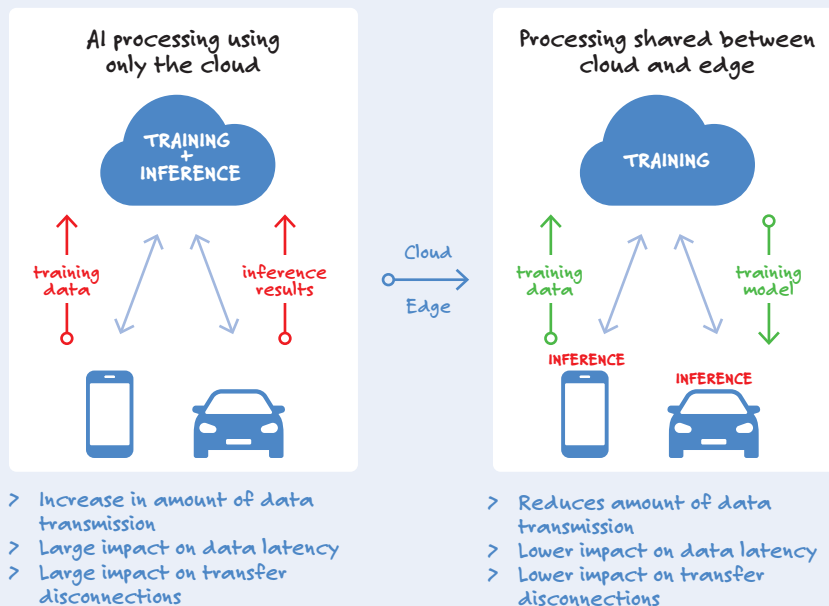
★★★★★

I Love Elektor!

Cloud AI vs Edge AI

Cloud AI is ideal for complex, resource-intensive tasks, but it faces latency and security challenges due to data transmission [1]. Edge AI is preferred for real-time feedback applications, despite its limitations in computational power [1]. Hybrid AI models combine cloud and edge strengths, handling intensive tasks in the cloud while enabling real-time inference at the edge [2]. This hybrid approach optimizes efficiency, reduces bandwidth consumption, and enhances privacy by keeping sensitive data on the device [2].

Source: Lionbridge AI [3]

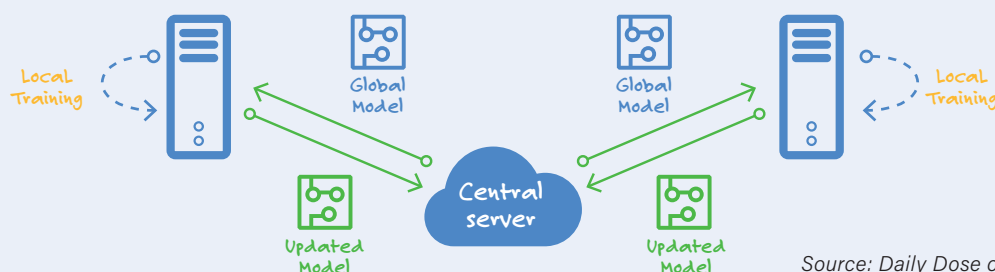


Edge AI & Distributed AI

Distributed AI represents the next stage in AI evolution, offering a solution to the scalability challenges often encountered in deploying edge AI across multiple locations. By adopting a *hub-and-spoke model*, a central hub manages AI capabilities, and spoke locations send data to be processed. The hub acts as a control plane, deploying applications and managing data lifecycle. This model helps ease the strain of large data transfers by implementing intelligent data collection, ensuring that only necessary data is processed [1].

Challenges like "data gravity" (the cost and network bandwidth of transferring large amounts of data) and "heterogeneity" (the need to adapt AI models for various use cases across different locations) are found in Distributed AI [4]. Solutions like automating data and AI lifecycle management and adapting AI pipelines for diverse applications are important to overcoming these obstacles.

Federated Learning



Source: Daily Dose of Data Science [6]

Federated learning is a machine learning (ML) technique that involves training an algorithm through several decentralized edge devices or servers that carry local data samples without

sharing them [5]. This method differs from conventional centralized machine learning methods, which require all local datasets to be submitted to a single server. Data protection,

telecommunications, IoT, and pharmaceuticals are among the sectors where it's used.

Generative AI: Economic Potential

**\$2.6 to \$4.4
Trillion**

Potential value added by generative AI [2].
The estimated global profit impact across industries like R&D, product development, and simulation in life sciences and manufacturing.

**0.1 to
0.6%**

Annual productivity growth [7]. Generative AI could enable labor productivity growth through 2040, with the potential for 0.5 to 3.4% of productivity growth when combined with all other technologies.

**2030 to
2060**

The estimated timeframe during which half of today's work activities could be automated, with a midpoint in 2045 [8].

Generative AI in 2024: Adoption and Impact

As generative AI adoption grows, survey respondents [8] report clear benefits and a reduced risk of inaccuracy.

65%

Organizations that regularly use generative AI — nearly double the adoption rate compared to a survey taken 10 months prior, signaling rapid integration into business operations.

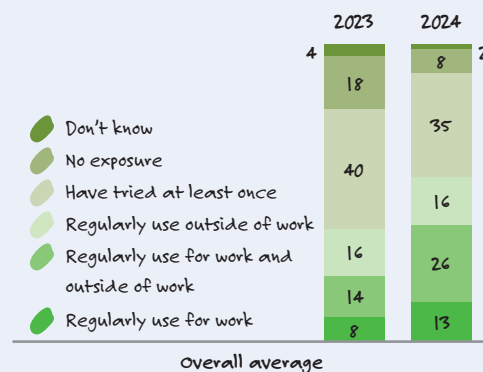
75%

Predict that generative AI will be disruptive. Respondents believe GenAI will lead to major changes in their industries in the coming years.

72%

Overall AI adoption rate. A significant increase from 50% in previous years, with widespread interest across industries and regions.

A significant increase in GenAI usage compared to 2023



(Source: McKinsey Global Survey on AI [8])

240692-01

WEB LINKS

- [1] IBM, "What Is Edge AI?": <https://www.ibm.com/think/topics/edge-ai>
- [2] Edge AI Foundation, "Edge AI Technology Report," Wevolver 2024: <https://www.wevolver.com/article/the-guide-to-generative-ai-at-the-edge>
- [3] Lionbridge AI, "What Is Edge AI Computing?," Medium, DataDrivenInvestor: <https://medium.com/datadriveninvestor/what-is-edge-ai-computing-61ece58c76d0>
- [4] M. Keerthi, "Edge AI vs Distributed AI," Medium 2024: <https://mukul04-sk.medium.com/edge-ai-vs-distributed-ai-154060456f1b>
- [5] Dr. J. Kaur Gill, "Edge AI vs Federated Learning," Xenonstack, December 2024: <https://tinyurl.com/2brehfn8>
- [6] A. Chawla, "Federated Learning: A Critical Step Towards Privacy-Preserving Machine Learning," Daily Dose of Data Science 2023: <https://tinyurl.com/dailydoseofds>
- [7] M. Chui et al., "The Economic Potential of Generative AI," McKinsey Insights 2023: <https://tinyurl.com/ecogenai>
- [8] A. Singla et al., "The State of AI in Early 2024," QuantumBlack, AI by McKinsey, McKinsey Insights 2024: <https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai/>

The Intel 8279 Keyboard/Display Interface

By David Ashton (Australia)

These days, you can easily interface your microcontroller with humans. Plug in an HDMI monitor and a USB keyboard, and off you go. In the (good?) old microprocessor days, it wasn't so easy, and interfacing was quite an art.

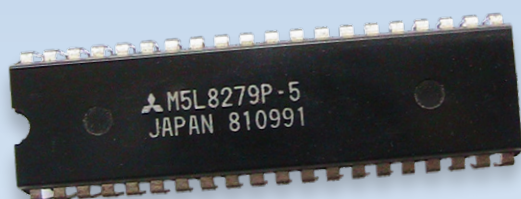


Figure 1:
A second-sourced
(Mitsubishi) 8279 IC.
(Image by JWBE - Own
work, CC BY-SA 3.0,
[https://commons.
wikimedia.org/w/index.
php?curid=20019181](https://commons.wikimedia.org/w/index.php?curid=20019181))

There were many peripheral chip offerings from the major microprocessor companies to make the embedded designer's life easier. One of these was the Intel 8279 Keyboard/Display interface. It enabled an Intel 8080 or 8085 microprocessor system to talk to a 64 key keyboard and two 16 digit numerical displays (7-segment) or one 16 character alphanumeric display. **Figure 1** shows a second-sourced (Mitsubishi) 8279 IC.

Multiplexing

Figure 2 shows a basic 8279 system diagram. Key to the operation were the four scan output lines. They could address eight lines of an 8×8 keyboard matrix or 16 digits of a (alpha)numerical display. To do so, you needed additionally a 3-8 line decoder (for the keyboard) or a 4-16 line decoder (for the display). For the keyboard matrix, eight return lines indicated which of up to 64 keys had been pressed. There were two additional inputs for *Ctrl* and *Shift* keys. The 8279 was then returning a code to the microcontroller — corresponding to the key pressed.

There were many refinements that made the chip very versatile. Usually, the keyboard was an 8×8 matrix and the 8279 returned a code corresponding to the key pressed, but you could also use what was called *Sensor Matrix* mode, where the keyboard scan lines selected a sensor which then input 8 bits of data back to the 8279 (for instance DIP switch settings or ADC output data). Keyboard or sensor data were stored in a FIFO buffer for later retrieval.

The display could be set to left entry mode (data is entered at the leftmost digit and moves right) or right entry mode, which means the multiplexed drive starts addressing the individual digits from L to R or in R to L direction. The eight display data lines could be used directly (for 7-segments plus decimal point, for example) or decoded for alphanumeric displays. There was also the option to split up the data into 2×4 line buses, controlling a 2×16 digit display.

Parallel Interface

In common with most Intel peripheral chips, many functions were controlled and specified via control registers. Apart from the eight data lines to the microcontroller, there were six or seven control lines. These days, if you want to interface to 7-segment displays you can get ICs like the ICM7218 or MAX7219 to drive up to 16 displays, but compared to the 8279 they're not nearly as versatile. ◀

240691-01

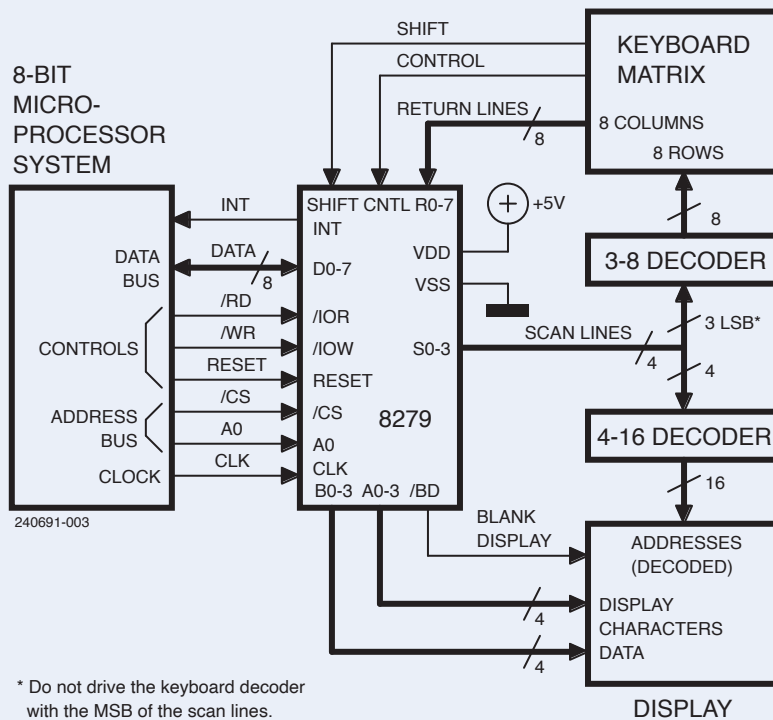


Figure 2:
Basic 8279 block
diagram. (Source:
Intel 8279 Datasheet)

They trust us, do you?

elektor.com

Great service.
Nice store, fast shipping, good packing and documentation.
Jun 27, 2024

Excellent product at a good price
Excellent product at a good price with fast trouble free delivery.
Date of experience: November 25, 2023

Easy ordering process(online)
Easy ordering process(online), good communication of delivery steps.
Overall smooth process
Date of experience: June 01, 2024

Great service
Great service!
Easy to purchase and delivered just in time as promised.
Date of experience: February 12, 2024

We love electronics and projects, and we do our utmost to fulfill the needs of our customers.
The Elektor Store: **'Never Expensive, Always Surprising!'**

Elektor Store
Reviews 365 • Excellent
★★★★★ 4.3
VERIFIED COMPANY

Check out more reviews on our Trustpilot page: www.elektor.com/TP/en

Or make up your own mind by visiting our Elektor Store, www.elektor.com



Makerfabs SenseLoRa

Plug and Play IoT for Greenhouses

By Clemens Valens (Elektor)

If you've ever wanted to set up a wireless monitoring system in your vegetable garden but were discouraged by the complexity of creating a remote, solar-powered, water-resistant setup, the SenseLoRa system from Makerfabs could be just what you need. It offers all those features while being completely plug-and-play, requiring no configuration effort. In this article, we test the SenseLoRa Industrial-grade Air Monitor and the corresponding LoRa Receiver to see if they truly deliver on their promises.

It was a warm and sunny day when I decided to try out the SenseLoRa system from Makerfabs. My setup consisted of a SenseLoRa LoRa Receiver and the Industrial-grade Air Monitor. A third unit exists too, the SenseLoRa Industrial-grade Soil Remote Monitor, but I didn't have one.

A quick look through the online documentation showed me that I was dealing with some kind of plug-and-play environmental monitoring system that didn't need any configuration. So, I connected the receiver through a USB hub to a laptop computer. I put its antenna outside, stuck with its magnet to the metal roof of my workspace. Then I opened the Air Monitor box to switch it on (the power switch is inside), closed it again and put it in the front yard (hidden by the house, so no line of sight), about 25 m away from the receiver. When I returned to the receiver, it already displayed (JSON) data that was sent by the Air Monitor (**Figure 1**):

```
Num:1 | -80dbm
{"ID":"AirM01",
"COUNT":2,
"SLEEP":3600,
"bat":3.90,
"Temp":28.38,
"humi":62.62,
"eco2":400.00,
"lux":189.17}
```

Plug and play indeed!

Note the low light level (**lux**). This data was probably collected when I was preparing the unit or holding it in my hand.

Air Monitor

As you can gather from the received data (temperature, humidity, eCO₂, and light intensity), the Air Monitor is intended for use in greenhouses. However, it might be useful too in classrooms and other spaces where people meet, as it can tell you when to open a window or when to switch on or off the lights.

Figure 1: The receiver has a small display that shows received data in JSON format.



In case you like the concept, but not the sensors, know that these can be replaced by other sensors with an I²C port. Of course, you will have to adapt the software (open source) to accommodate other sensors.

ESP32-S3 with RF92

The Air Monitor consists of an ESP32-S3 — connected to an RF92 LoRa module from HopeRF — and an I²C bus to which three sensors are connected:

- SGP30 for air quality (equivalent CO₂, i.e., eCO₂, 0 to 1000 ppm)
- BH1750 for ambient light (1 to 65535 lx)
- AHT10 for temperature (−40°C to 80°C, ±0.3) and humidity (0 to 100%)

The device can be configured over Wi-Fi when you put it in AP mode. This lets you change its ID (practical when the system comprises several Air Monitors) and the transmission period.

The Air Monitor has a 1000 mAh rechargeable battery inside that is charged by the kit's solar panel and that keeps it running during the night.

Industrial Grade?

The Air Monitor is labelled Industrial Grade (**Figure 2**), which probably refers to its IP68-rated enclosure. It comes with a 6-V, 6-W solar panel with a mounting bracket, nuts and bolts, and an antenna (1-m cable). The antenna looks like an indoor device and only has a magnet in its base for sticking it to some metal object. According to the user manual, it is suitable for outdoor use. The mounting bracket made of painted (enameled?) iron will probably not last long outside. But then again, the Air Monitor is intended for use inside greenhouses (where iron doesn't rust, that's a well-known fact).

The LoRa Receiver

The LoRa Receiver comes as a small module consisting of a red PCB sandwiched between two transparent acrylic plates. It is based on an RP2040 microcontroller and has an OLED display, microSD-card slot (a 16 GB microSD card is included), a HopeRF RF96 module and a USB connector. This is the part I like the least. The USB connector is a male A-type, turning the module into a large dongle that you are supposed to stick into a computer. This severely limits its placement unless you add a USB hub. Furthermore, the module is quite wide (34 mm) and high (18 mm) for a USB stick, and so it will probably block access to other ports of the host device. **Figure 3** shows how I ended up using the module.



Figure 2: The SenseLoRa Industrial-grade Air Monitor mounted outdoors on a wooden structure at a height of almost two meters.

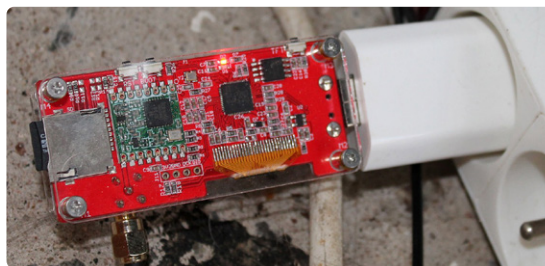


Figure 3: This is how the SenseLoRa receiver ended up in a real-life logging application.

To keep the display readable for most mounting situations, a pushbutton (TFT) is available to rotate it by 180°. An option to switch it off would have been welcome.

The antenna is of the same type as the one for the Air Monitor. Its long cable (5 m) is practical, and so is its magnet if you have a metal surface to stick it on (**Figure 4**). If you haven't, you'll have to use adhesive tape or tie-wraps or something to fix the antenna.



Figure 4: The receiver antenna stuck to the metal roof of the shed.

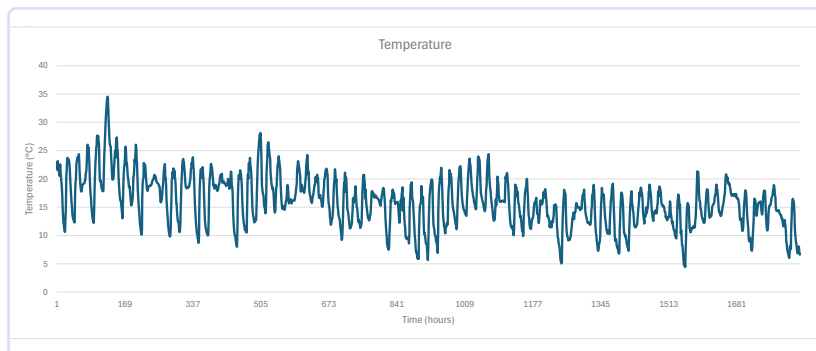


Figure 5: Almost eleven weeks of continuous temperature logging, °C, one sample per hour. I had hoped for a warmer summer :-).

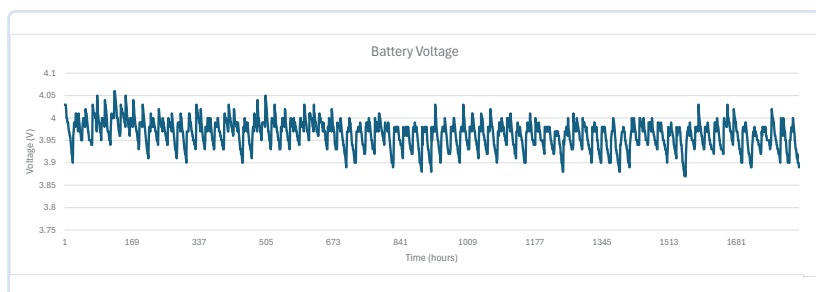
Endless Data Logging

Besides displaying the incoming data, the receiver also transmits it verbatim over its USB serial port and stores it on its microSD card (if one is inserted). Interesting is, at least that's what I thought, that it keeps appending to the same log file on the microSD card, even after a power cycle or reboot. The file is not overwritten or recreated at boot time. Therefore, any existing data is not lost unless you delete it deliberately (Figure 5 and Figure 6). Note that when recording the serial port data, you may also capture boot and status messages when the TFT button is pressed. These may break a simple data converter script. A Python script for recording and decoding data is available at [1].

Very Hackable

Both the Air Monitor and LoRa Receiver are very hackable devices. The schematics, board design files (Eagle) and source code are all published on GitHub together with the user manual [1][2]. The LoRa Receiver is built around an RP2040 microcontroller while the Air Monitor is equipped with an ESP32-S3 module. Both are well-known microcontrollers in maker land. The software is Arduino-based, making it easy to adapt it to your needs and desires.

Figure 6: The battery remains charged even when there is no sun.



LoRa Easy to Use

With the SenseLoRa concept, Makerfabs has tried to make LoRa easy to use for point-to-point connections. Configuration of complicated LoRaWAN and cloud services is avoided this way. The system is plug and play and works out of the box simply by switching it on. Extending the system is possible with minimal configuration (only the device ID must be set). Its simplicity makes SenseLoRa practical for quickly setting up a greenhouse monitoring system. However, what will happen when your neighbor decides to do the same is unclear.

Even though some parts are said to be industrial grade, this must be taken with a grain of salt. True, the Air Monitor has an IP68-rated enclosure, but running two cables (antenna and power) through the same cable gland probably turns the "8" into a "3". Also, the receiver in the shape of a large USB dongle with an open enclosure lacks robustness for tough industrial environments. Yet, I like the SenseLoRa concept, as it really does make things easy at low cost. Also, the fact that it is open source and hackable makes this an attractive system for makers and small businesses. ◀

240407-01

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.



Related Products

> **Makerfabs SenseLoRa LoRa Receiver (EU868)**
www.elektor.com/20883

> **Makerfabs SenseLoRa Industrial-grade Air Monitor (EU868)**
www.elektor.com/20763



WEB LINKS

[1] SenseLoRa LoRa Receiver user manual: <https://github.com/Makerfabs/SenseLoRa-LoRa-Receiver>

[2] SenseLoRa Air Monitor user manual: <https://github.com/Makerfabs/SenseLoRa-Industrial-grade-Air-Monitor>