

OBD2 SENSOR DASHBOARD

Real Vehicle Data on an Adaptable Display

FOCUS ON

IoT & Sensors



OBD2 Rev Counter and Gear Shift Indicator

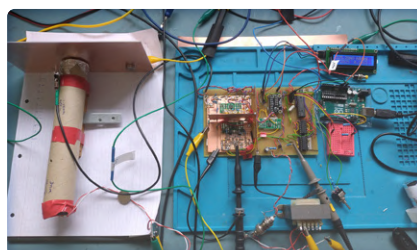
Save Fuel and Keep Emissions to a Minimum

LiDAR and Vision Sensors

For Robotics and More



Solar Charge Controller with MPP Tracking
The Circuit in Detail



Contact-Free E-Field Measurements Assessing DC Voltages or Static Fields



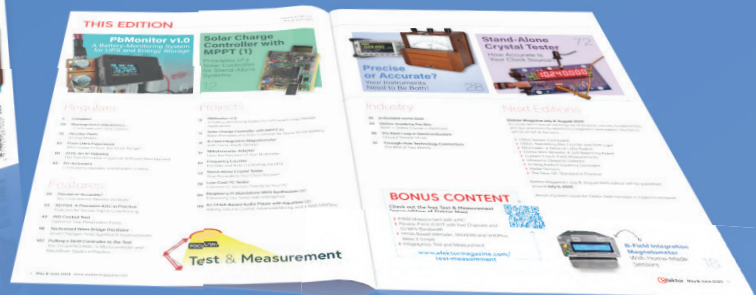
Analog Audio Frequency Generator
Adjustable-Frequency Sine Wave



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

Volume 51, No. 538
 July & August 2025
 ISSN 1757-0875

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
www.elektor.com | www.elektormagazine.com

Content Director: C. J. Abate
Editor-in-Chief: Jens Nickel

For all your questions
service@elektor.com

Become a Member
www.elektormagazine.com/membership

Advertising & Sponsoring
 Büsra Kas
 Tel. +49 (0)241 95509178
busra.kas@elektor.com
www.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.

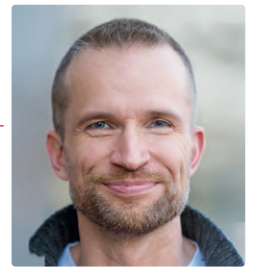
Print
 Senefelder Misset, Mercuriusstraat 35,
 7006 RK Doetinchem, The Netherlands

Distribution
 IPS Group, Carl-Zeiss-Straße 5
 53340 Meckenheim, Germany
 Phone: +49 2225 88010



Jens Nickel

International Editor-in-Chief, Elektor Magazine



Sensors and Wrenches

The IoT is everywhere right now — including in our editions. We've featured remote control and remote monitoring projects in past issues, and we have LoRa and 4G projects planned for the Wireless edition in September. So for this issue, we've decided to focus a bit more on sensors. It's an interesting field — not just for someone like me who studied physics.

This year, I visited the Sensor+Test fair in Nuremberg once again, and I was impressed by many new developments, including sound intensity scanning and antenna arrays for tracking thousands of Bluetooth devices (see page 26).

In this edition, my colleague Jean-François Simon takes a closer look at a radar sensor you can use for presence detection and many other applications (page 50). Saad Imtiaz from our lab reports on some of the best LiDAR sensors currently available (page 19). As you'll see, there are major differences in both specifications and cost. When I have time, I plan to buy one suitable for beginners and run my own experiments with it. A kind of 3D map of audio equipment — combined with sound direction measurements — could make for a fun, visionary project. Who knows? Maybe you'll read about it in July 2026 (or 2027).

Our main cover projects — yes, we have two this time — are aimed at a different kind of enthusiast: one who likes to solder and isn't afraid to pick up a wrench (pages 6 and 14). For people like that, a faint smell of gasoline and a free Saturday are the foundation for happiness. And I know we have plenty of them among our readers.

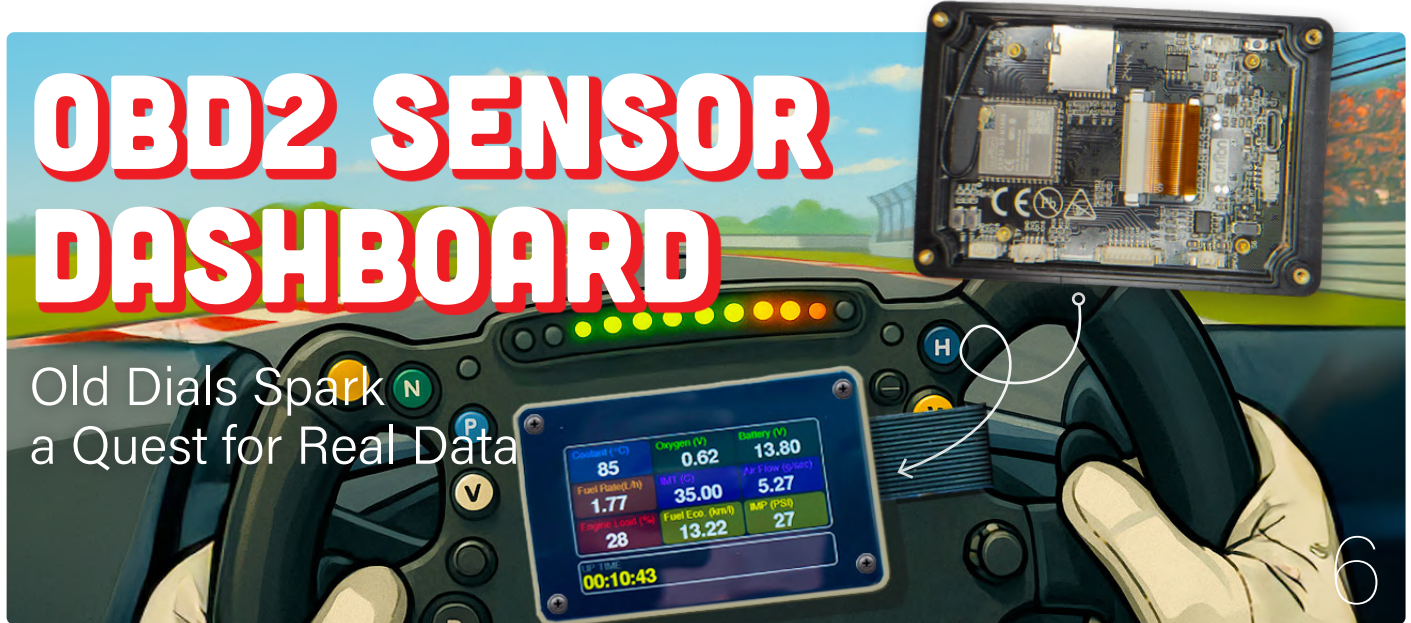
About 15 years ago, our "OBD2 Mini Simulator" was a big hit (<https://elektormagazine.com/magazine/elektor-201006/19323>). It even generated interest beyond our regular readership — if I remember correctly, even some German car websites picked it up.

I hope our two new OBD projects bring just as much joy to auto enthusiasts. As always, the articles don't just invite you to build them — they're also an invitation to modify and extend them to suit your own needs.

Whether you're mapping with LiDAR or decoding engine diagnostic codes, I hope this issue gives you plenty to tinker with!

The Team

International Editor-in-Chief: Jens Nickel | **Content Director:** C. J. Abate | **International Editorial Staff:** Hans Adams, Asma Adhimi, Mahy Arafa, Roberto Armani, Jan Buiting, Rolf Gerstendorf (RG), Ton Giesberts, Saad Imtiaz, Alina Neacsu, Dr. Thomas Scherer, Jean-François 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



Regulars

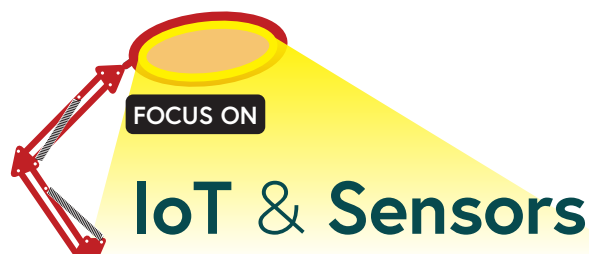
- 3 Colophon**
- 48 Peculiar Parts**
Solar Cells
- 56 From Life's Experience**
Paper Factory
- 70 Err-lectronics**
Corrections, Updates, and Readers' Letters
- 73 Starting Out in Electronics...**
...Concludes the Topic of Opamps
- 92 2025: An AI Odyssey**
Mid-Year Review

Features

- 19 LiDAR and Vision Sensors for Robotics**
- 50 Getting Started with a Modern Radar Sensor**
Is Your Radar Accurate?
- 66 Exploring Wireless Communication with the BeagleY-AI**
- 76 A Powerful AI Code Assistant**
Speed Up Your Development with Continue and Visual Studio Code

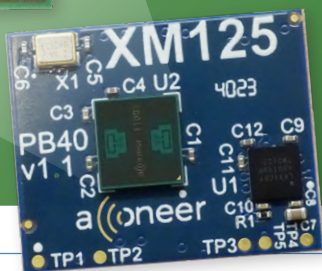
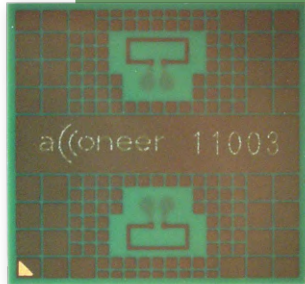
Projects

- 6 OBD2 Sensor Dashboard**
Old Dials Spark a Quest for Real Data
- 14 OBD2: Rev Counter and Gear Shift Indicator**
Retro, But Super Useful
- 28 Contact-Free E-Field Measurements (1)**
A Vibrating Membrane for Assessing DC Voltages or Static Electric Fields
- 36 Wireless Mailbox Notifier**
From Optical Sensors to Radar, Exploring a Few Options
- 42 Elektor Mini-Wheelie**
A Self-Balancing Robot
- 82 Solar Charge Controller with MPPT (2)**
The Circuit
- 88 Ultrasonic Obstacle Detector**
A Simple Project to Help Those with Impaired Vision
- 94 Raspberry Pi Standalone MIDI Synthesizer (3)**
Making It Smarter and Adding a User Interface
- 102 Meshtastic: A Demo Project**
An Intelligent Mesh of LoRa Radios
- 110 Analog Audio Frequency Generator**
High-Quality Adjustable-Frequency Sine Wave Generator



Getting Started with a Modern Radar Sensor

Is Your Radar
Accurate?



50



Solar Charge Controller with MPPT

The Circuit

82

Industry

- 26 Sensor+Test 2025 and PCIM 2025**
- 58 Cybersecurity**
Tough Times for Hackers
- 62 Siglent Presents Next-Gen Multichannel Oscilloscopes**
High-Performance Solutions for Modern Power and Embedded Systems
- 64 Bluetooth 6.0 Brings Enhanced Distance-Ranging Applications**
New Version Offers Improved Device Positioning and Location Services

BONUS CONTENT

Check out the free IoT & Sensors
bonus edition of Elektor Mag!

- Full-Range Analog Input for the ESP32
- Review: Topdon TC004 Lite Thermal Imaging Camera
- Secure Firmware for Microcontrollers
- Infographics: IoT and Sensors



www.elektormagazine.com/iot-sensors

Next Editions

Elektor Magazine Circuit Special 2025

In the tradition of our Summer Circuits, the next edition will be extra thick, filled with dozens of DIY projects, retro circuits, tips and tricks, and much more! Circuit Special 2025 will be published around **August 13, 2025**.

Elektor Magazine September & October 2025

As usual, we'll have an exciting mix of projects, circuits, fundamentals, and tips and tricks for electronics engineers and makers. Our focus will be on Wireless & Communication.

- Navigating Wireless Protocols
- Accurate Positioning with Bluetooth
- Satellite Tracking Using LoRa
- Wireless Audio Transmission
- AM Transmitter
- Performance Tests with the RP2350
- Model Car Remote Control with an ESP32
- Raspberry Pi Zero Web Streaming Camera

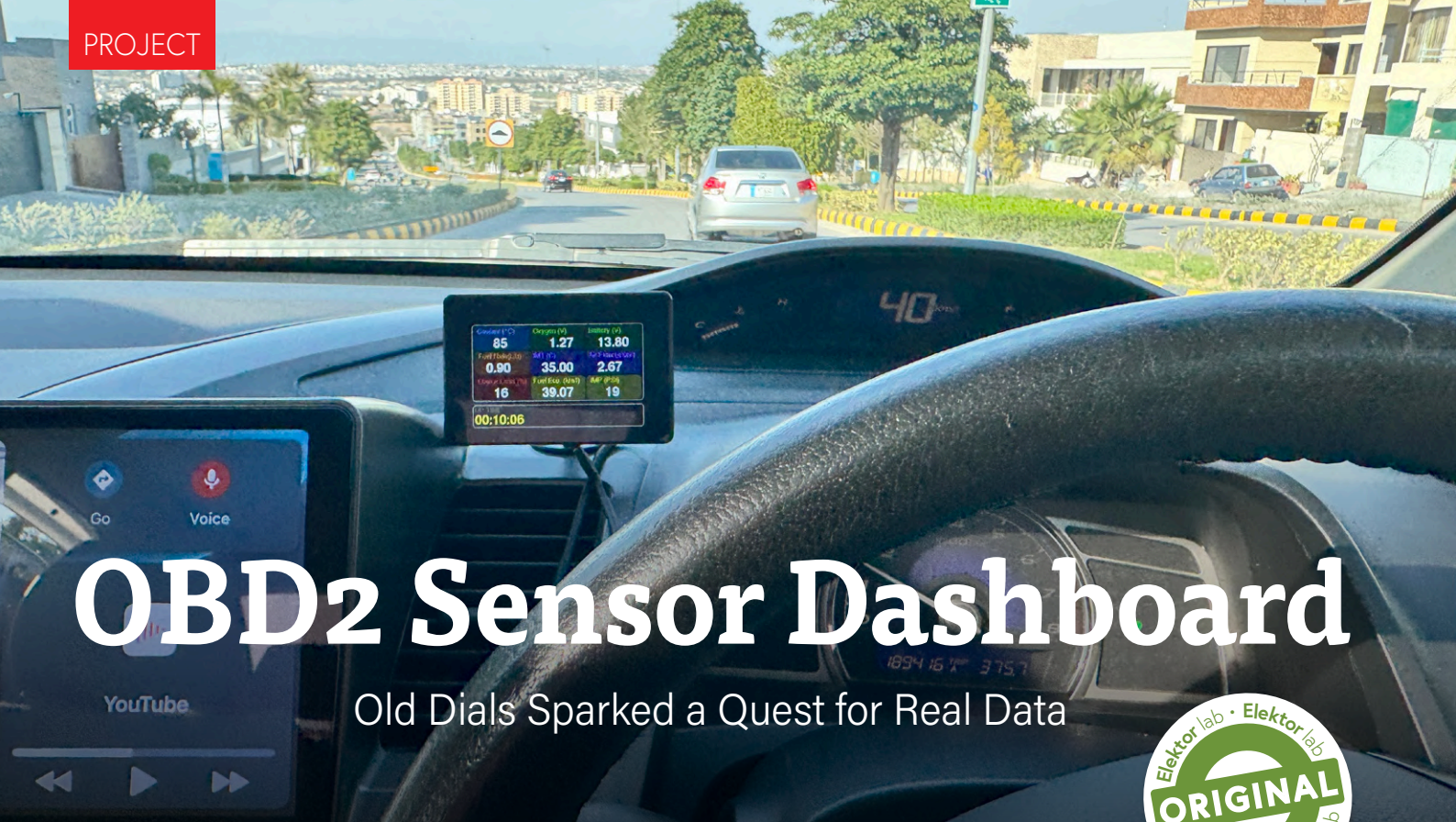
Elektor Magazine's September & October 2025 edition will be published around **September 10, 2025**.

Arrival of printed copies for Elektor Gold members is dependent on shipping times.



**OBD2: Add a Rev Counter
and Gear Shift Indicator
to Your Car**
Retro, But Super Useful

14



OBD2 Sensor Dashboard

Old Dials Sparked a Quest for Real Data



By Saad Imtiaz (Elektor)

This article details the development of a real-time automotive sensor dashboard by using an ELM327 OBD2 module and an ESP32-S3 touch display. The dashboard shows essential sensor data — including coolant temperature, battery voltage, oxygen sensor voltage, fuel economy, and engine load — retrieved directly from the vehicle's ECU, offering instant, reliable diagnostics, and opportunities for future enhancements.

Understanding OBD2

Since about 2002, all vehicles sold in most markets have been equipped with a standardized 16-pin On-Board Diagnostics II (OBD2) interface. This interface replaces the manufacturer-specific diagnostic connectors that were common before its introduction. Governed by ISO and SAE standards (such as ISO 9141-2, ISO 14230-4, and ISO 15765-4), the OBD2 system allows access to a vehicle's ECU for reading diagnostic trouble codes (DTCs), emission system statuses, and live sensor data.

Initially, several communication protocols coexisted — ISO 9141-2 and KWP2000 were used widely in the early years. However, modern vehicles predominantly use the Controller Area Network (CAN) protocol (ISO 15765-4), which is faster and more reliable.

It all began one early morning, sitting in my 2001 Toyota Land Cruiser, waiting patiently as the engine warmed up. Watching the analog dials slowly climb, I wondered: Why rely on vague needles for crucial parameters like coolant temperature, battery voltage, or oil pressure (**Figure 1**)? Wouldn't precise numeric values directly from the vehicle's ECU be far more insightful? This simple thought sparked the idea of developing a custom OBD2 dashboard, directly tapping into the vehicle's real-time sensor data via an ELM327 module and visualizing it clearly on an ESP32-S3-based touch display.

In this article, we'll explore the entire development journey, covering OBD2 communication, hardware selection, UI design, troubleshooting challenges, and firmware implementation. The resulting system provides accurate, real-time data, and serves as a flexible template — easy to customize and extend by integrating any sensor data supported by your vehicle's Electronic Control Unit (ECU).



Figure 1: Analog dashboard view of the 2001 Land Cruiser, displaying traditional gauge dials.

Beyond real-time values, OBD2 also offers access to fault memory and the system's readiness status. One of its main purposes remains emissions control monitoring. If an error affects emissions, the system triggers the Malfunction Indicator Lamp (MIL), alerting the driver via the instrument cluster. While features like airbag status or braking condition are not within the OBD2 scope, the system ensures the exhaust and fuel systems operate within permissible limits. During periodic vehicle inspections, these readiness codes are checked to confirm a well-functioning emission system.

Physically, the diagnostic port is typically located under the dashboard, often in the driver's footwell or near the center console. Earlier models may have hidden it behind covers or panels, but modern cars follow the easy accessibility requirement.

The ELM327: Bridging the Vehicle ECU and the Display

To tap into the OBD2 system, the widely available ELM327 interface was preferred. The ELM327 acts as a translator between the car's OBD2 system and external devices. It supports multiple protocols including CAN, ISO 9141-2, and KWP2000, automatically detecting the vehicle's protocol and establishing communication. Inside, it parses OBD2 requests and responses, presenting them over a UART, Bluetooth, or USB interface.

In **Figure 2**, the block diagram of the project is shown. To visualize data, you can use for example a smartphone. Various smartphone apps utilize ELM327-based adapters to display vehicle data. However, my focus was to use an ESP32 and a display, rather than a smartphone.

Initial Testing and Troubleshooting

Initially, the plan was to use a Bluetooth-based ELM327 module to wirelessly connect the ESP32 with my vehicle's OBD2 port. The first testing began with my 2001 Toyota Land Cruiser, where I connected the ELM327 Bluetooth module and attempted to communicate via an

Android application on the smartphone. Despite correct wiring and initialization, the module consistently failed to establish communication or retrieve sensor data. Suspecting faulty hardware, multiple ELM327 adapters were acquired from different suppliers. After opening and examining their PCB, it turns out they were identical from the inside, having the same PCB with different plastic enclosures (**Figure 3**), eliminating module defects as a potential cause.

Further investigation led to inspect vehicle's OBD2 port closely. It was discovered that the 2001 Land Cruiser lacked CAN High (ISO 15765-4) connections and only minimally supported the older K-Line (ISO 9141-2/ISO 14230-4) protocol — restricted mainly to basic diagnostic trouble codes (DTCs). Consequently, real-time data access (such as RPM, coolant temperature, and fuel economy) wasn't available, a common limitation of early generation OBD2 vehicles.

Undeterred, I continued testing with a newer vehicle — a 2007 Honda Civic equipped with complete CAN protocol support. On this vehicle, communication via Bluetooth was immediately successful, confirming that the issue was protocol-related rather than hardware-specific.



Figure 3: External view of three visually different ELM327 clone modules, which share identical internal circuitry.

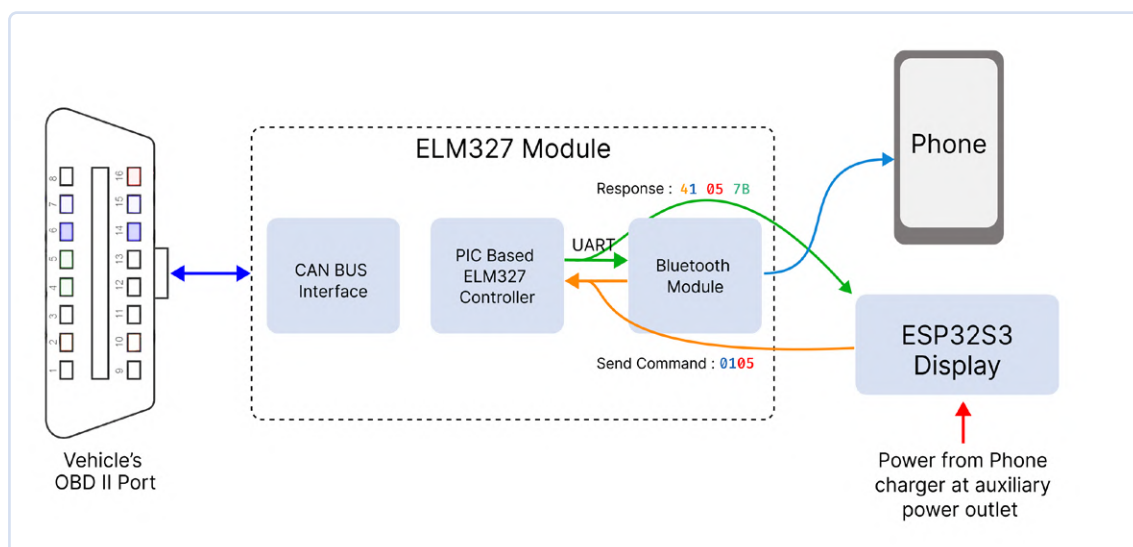


Figure 2: The block diagram.



Figure 4: The JC3248W535 ESP32-S3-based LCD development board used for the sensor dashboard.

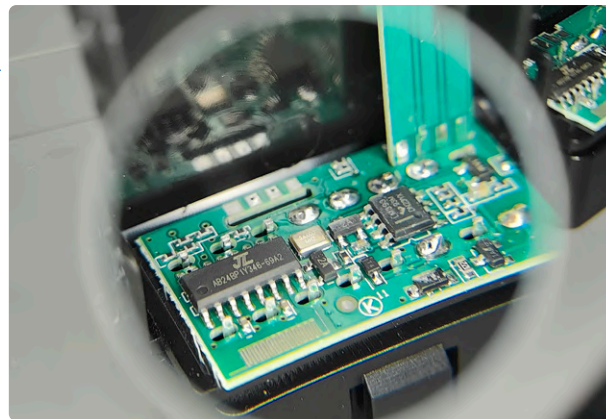


Figure 5: Detailed internal view highlighting the identical PCB of the common single-board ELM327 clone modules.

However, another challenge emerged: the JC3248W535 ESP32-S3 board [1] I selected only supports Bluetooth Low Energy (BLE), while the ELM327 modules rely solely on standard Bluetooth.

While, I successfully tested Bluetooth connectivity using a standard ESP32 module (which supports traditional Bluetooth). However, at the end, I ultimately favored the JC3248W535 ESP32-S3 3.5-inch LCD Development Board due to its aesthetic appearance, integrated capacitive touchscreen with a resolution of 480×320 pixels (**Figure 4**), built-in battery management circuitry, SD card and onboard speaker support. The integrated built-in speaker opens possibilities for voice warning system reminiscent of aircraft alerts — such as “Fuel Low,” or “Engine Fire” and many more (though I sincerely hope no car ever experiences that last one).

This compatibility mismatch prompted a decision to switch from a Bluetooth wireless approach to a direct, stable UART connection. This change ensured reliable communication between the ESP32-S3 and the ELM327 module. For those who prefer a wireless approach, a suitable alternative smaller display called the “Cheap Yellow Display,” compatible with classic Bluetooth, is available on Elektor’s store site [2].

Project Hardware Setup

Following the decision to shift to UART, I needed an ELM327 module variant with easily accessible serial pins. Most cost-effective ELM327 modules integrate Bluetooth and OBD2 processing into a single chip

without exposing UART (**Figure 5**). To overcome this limitation, the “Double PCB” variant of the ELM327 was selected [3], comprising two stacked boards (**Figure 6**). Interestingly, at the time of planning this article, we got another OBD project from one of our authors, and it makes use of the same module for a gear-shift indicator system [4].

In this module, the lower board contains a PIC microcontroller responsible for OBD2 communication, while the upper board handles Bluetooth functionality. Conveniently, these two boards communicate internally via UART, exposing RX, TX, and Ground pads. I soldered jumper wires directly to these UART pins, attaching a compact 4-pin JST 1.25-mm connector at the other end (**Figure 7**). This connector seamlessly interfaces with the UART pins (IO17 for RX, IO18 for TX) on the JC3248W535 ESP32S3 development board (**Figure 8**). The schematic diagram is shown in **Figure 9**.

For secure dashboard mounting, a custom backplate was designed in CAD equipped with a pivotable stand, attaching it with strong double-sided adhesive tape directly onto the vehicle’s dashboard (**Figure 10**). To withstand high cabin temperatures (often exceeding 55°C), PETG filament was chosen for 3D printing the stand and backplate, as it offers better heat resistance (approximately 85°C) compared to standard PLA, which softens around 55°C . While ABS filament would have been even more resistant (up to 110°C), it requires specialized enclosures and ventilation systems due to its toxic printing fumes, making PETG the practical material choice.



Figure 6: The PIC microcontroller-based “Double PCB” ELM327 module variant, featuring two stacked circuit boards.

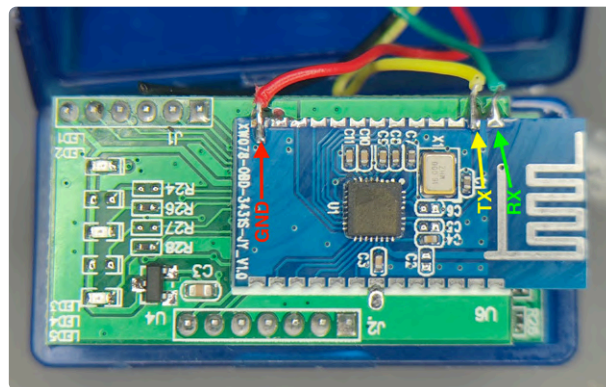


Figure 7: UART jumper wire connections soldered onto the bluetooth PCB of the PIC-based ELM327 module.

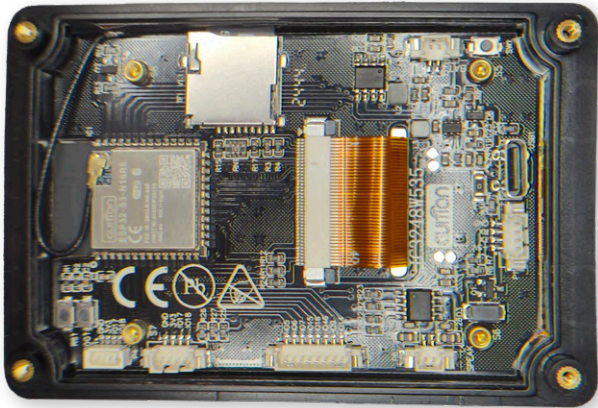


Figure 8: Backside of the JC3248W535 development board, showing GPIO connections. UART lines connected to GPIO 17 (RX) and GPIO 18 (TX) in the bottom-right corner.

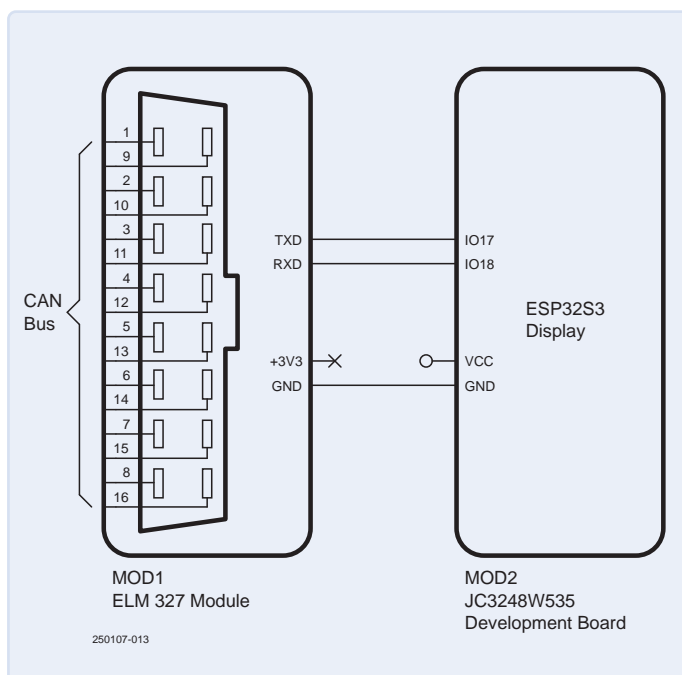


Figure 9: Schematic diagram: connections between the ELM327 module and the ESP32-S3 development board.



Figure 10: Final OBD2 sensor dashboard mounted securely inside the vehicle, as seen from outside the windshield.

Firmware and Communication

The firmware is based on the Arduino framework, leveraging *Arduino_GFX* library for graphical rendering and standard Arduino libraries for serial communication. Communication between the ESP32-S3, ELM327 module, and the vehicle's ECU follows a structured request-response pattern. The ESP32-S3 sends specific commands using OBD-II Parameter IDs (PIDs) over UART at 38400 baud. The ELM327 interprets these commands and queries the ECU, returning responses as ASCII strings containing hexadecimal-encoded sensor data. (See the textbox **Understanding the OBD2 Protocol**.) The firmware and all the files related to this project are shared via the GitHub repository [5].

Since the code follows a consistent flow — sending a command, parsing the received response, applying a specific formula based on the command, and then displaying the result — it's unnecessary to include every repeated instance in the article. Instead, only the key sections of the code are provided to clearly illustrate this core functionality.

Parsing Engine Coolant Temperature (PID: 0105)

The process of retrieving sensor data, such as the engine coolant temperature, begins with the `requestOBDData()` function, which sends a request to the ELM327 module:

```
requestOBDData(i);
// Where obdPIDs[i] is "0105"
// for coolant temperature
```

Internally, this function sends the PID command to the ELM327:

```
sendOBDDCommand("0105");
// Sending command for coolant temperature
```

The vehicle's ECU responds to the ELM327, which returns a raw response string via UART, for example `41 05 7B`. Here, `41` confirms a successful response (mode 1), `05` specifies the requested PID (coolant temperature), and `7B` represents the coolant temperature data.

This raw response is passed to the `parseOBDResponse` function, starting with cleaning unnecessary characters and spaces:

```
response = cleanResponse(response);
// Cleans response, returns "41057B"
```

Now, the cleaned response (`41057B`) is parsed based on the specific PID requested (0105). The parsing logic extracts the hexadecimal data:

```
hexA = response.substring(4, 6);
// Extracts "7B" from response
A = strtol(hexA.c_str(), NULL, 16);
// Converts "7B" from HEX to decimal (123)
value = A - 40;
// Applies formula for coolant
// temperature (123 - 40 = 83 °C)
```

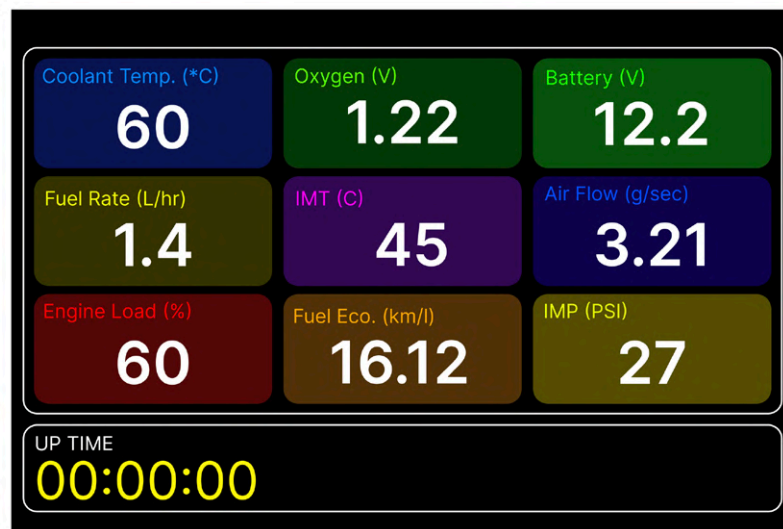


Figure 11: Initial Figma concept design used for visualizing and refining the sensor dashboard user interface.

After parsing, the value (83 °C) is available for further use:

```
Serial.printf("Coolant Temp: %.2f°C\n", value);
// Outputs: Coolant Temp: 83.00 °C
```

This value is then displayed on the screen using graphical functions defined in the firmware:

```
drawCoolantTemp(value);
// Visually displays the coolant
// temperature (83 °C) on the screen
```

This process — sending commands, receiving and cleaning responses, parsing data using PID-specific formulas, and displaying parsed values — is consistently followed for each parameter requested from the vehicle's ECU.

User Interface Design

The parsed sensor values are continuously displayed using the graphical capabilities provided by the *Arduino_GFX* library [8]. Real-time values update dynamically on the TFT screen, offering the driver immediate feedback on vehicle conditions, such as instantaneous fuel economy, battery voltage, and engine parameters.

For the user interface, I took inspiration from various after market racing dashboards available online, ultimately combining selected elements into a unique, personalized layout. Initially, Figma Design [9] was used (**Figure 11**) to precisely define and refine the visual concept before moving into actual code implementation.

Each sensor reading is presented in a dedicated visual card, enhancing readability and simplifying potential updates. The parameters chosen for display were guided by practical needs: Coolant Temperature for quick overheating detection, Oxygen Sensor Voltage due to frequent

check-engine warnings, and Battery Voltage to identify potential electrical issues. Additionally, metrics such as Fuel Rate and Fuel Economy encourage efficient driving habits, reminding me to keep my foot lighter on the gas. Parameters like Intake Manifold Temperature and Pressure, Mass Airflow, and Engine Load were selected to provide insight into engine efficiency and performance.

A personal favorite addition is the Uptime Timer, essentially a stopwatch activated upon engine start, initially intended as a warm-up timer but humorously useful for highlighting time wasted in traffic.

The following snippet illustrates how an individual sensor card, such as Coolant Temperature, is drawn on the TFT display. The `drawCoolantTemp` function takes the coolant temperature value and clearly presents it within a visually consistent card layout:

```
void drawCoolantTemp(int coolantTemp) {
    int cardWidth = 147;
    int cardHeight = 63;
    int x = 17;
    int y = 45;
    // Draw card background
    gfx->fillRoundRect(x, y, cardWidth,
        cardHeight, ROUND_EDGES, gfx->color565(0, 51, 102));

    // Draw icon
    gfx->setFont(&FreeSans9pt7b);
    gfx->setTextColor(gfx->color565(0, 128, 255));
    gfx->setCursor(x + 7, y + 18);
    gfx->print("Coolant ( C)");
    gfx->drawCircle(gfx->getCursorX() - 24,
        gfx->getCursorY() - 8, 2,
        gfx->color565(0, 128, 255));
    // Draw degree symbol
```



```
// Draw value
gfx->setCursor(x + 50, y + 55);
gfx->setTextColor(RGB565_WHITE);
gfx->setFont(&FreeSansBold18pt7b);
gfx->print(coolantTemp);
}
```

Sensor values are retrieved and parsed within the `parseOBDResponse` function, which ensures only supported, and valid sensor data is displayed. If a PID response isn't recognized or supported by the connected vehicle's ECU, the associated sensor card isn't rendered, keeping the interface streamlined:

```
else if (pid == "0105" && response.length() >= 6)
{ //Coolant Temperature
  hexA = response.substring(4, 6);
  A = strtoul(hexA.c_str(), NULL, 16);
  value = A - 40;
  drawCoolantTemp(value);
  Serial.printf("Coolant Temp: %.2f°C\n", value);
}
else {
  Serial.println("Unknown Response Format");
}
```

Although metrics like Transmission Temperature or Oil Pressure would have been beneficial, these are unsupported by my vehicle's ECU. However, integrating additional sensors into this project is straightforward if supported by one's vehicle.

The fully realized user interface, actively displaying real-time sensor data during driving, can be seen in **Figure 12**.



Figure 12: Close-up view of the finalized user interface for the custom OBD2 sensor dashboard, displaying real-time sensor data.

Understanding the OBD2 Protocol

On-Board Diagnostics II (OBD2) is a standardized vehicle diagnostic system introduced primarily to monitor and control vehicle emissions. It provides a structured method for accessing sensor data, retrieving diagnostic trouble codes (DTCs), and assessing overall engine performance directly from a vehicle's Engine Control Unit (ECU).

Communication with the ECU occurs using standardized messages known as Parameter IDs (PIDs). Each PID corresponds to a specific sensor or system status within the vehicle. To request sensor data, the OBD2 standard utilizes different "modes," with mode 01 commonly used to retrieve real-time sensor information.

A typical PID request and response format is as follows:

- Request (sent to ECU): Consists of the mode (01 for real-time data) and the PID.
Example: **01 05** requests the Engine Coolant Temperature.
- Response (received from ECU): Comprises three parts—confirmation of mode (0x40 + mode), the PID number, and the actual sensor data in hexadecimal form.

Example: A response of **41 05 7B** indicates:

- **41** (0x40 + 0x01) confirms successful mode 01 response.
- **05** confirms the requested PID.
- **7B** (hexadecimal) is the raw data value.

To obtain meaningful information, the raw hexadecimal response must be converted using a defined formula. For Engine Coolant Temperature (PID 05), the formula is:

Temperature (°C) = Decimal Value - 40

Thus, for 7B (hexadecimal), the coolant temperature is calculated as 123 (decimal) - 40 = 83°C.

Commonly used OBD2 PIDs

PID	Description	Formula/ Conversion	Units
05	Engine Coolant Temp.	A - 40	°C
0C	Engine RPM	((A × 256) + B) / 4	RPM
0D	Vehicle Speed	A	km/h
11	Throttle Position	(A × 100) / 255	%
10	Mass Air Flow (MAF)	((A × 256) + B) ÷ 100	g/s

To find out the PID requests along with their formulas, these two internet resources were used [6][7].

Future Improvements

As with any project, there's always room for further development and refinement. This custom dashboard serves as a solid foundation toward building a more sophisticated and customizable sensor interface. Currently, sensor parameters are hardcoded into the firmware, requiring manual reprogramming with the relevant PID values if users want to display different sensor data. While straightforward, removing the unit from the car or bringing a laptop to the vehicle every time can quickly become tedious.

To overcome this, the plan is to implement a webserver based configuration interface, enabling users to easily select and customize displayed sensors directly via smartphone or tablet without physically updating firmware. Additionally, integrating trip data logging would allow sensor readings to be stored on an onboard SD card in CSV format, enabling convenient download via the same web interface. This stored data can later be visualized and analyzed to track vehicle performance, identify trends, or troubleshoot issues.

Another exciting planned feature is a voice warning notification system inspired by "Bitchin' Betty," the iconic fighter jet alert system. Not only would this add a unique touch, but it would enhance safety by clearly announcing important notifications such as "Engine Overheating" or "Battery Voltage Low," reducing the need for constant visual monitoring.

Given that built-in batteries are impractical due to high cabin temperatures — particularly risky with LiPo batteries — the plan is to incorporate a Real-Time Clock (RTC) module. An RTC ensures accurate timekeeping for reliable data logging and timestamping, essential especially when combined with the intended GPS integration. This addition would offer real-time speed data, accurate route tracking, and comprehensive insights into driving performance and vehicle health.

Lastly, there is an aim to include support for wireless firmware updates (OTA), allowing easy, cable-free software upgrades and significantly enhancing user convenience and maintainability. ◀

250107-01

Questions and Comments?

If you have questions about this article, feel free to email the author at saad.imtiaz@elektor.com or the Elektor editorial team at editor@elektor.com.



About the Author

Saad Imtiaz, Senior Engineer at Elektor, is a mechatronics engineer who has extensive experience in embedded systems and product development. His journey has seen him collaborate with a diverse array of companies, from innovative startups to established global enterprises, driving forward-thinking prototyping and development projects. With a rich background that includes a stint in the aviation industry and leadership of a technology startup, Saad brings a unique blend of technical expertise and entrepreneurial spirit to his role at Elektor. Here, he contributes to project development in both software and hardware.



Related Products

- ▶ **ESP32 Development Board (Cheap Yellow Display)**
www.elektor.com/20890
- ▶ **Dogan Ibrahim, Controller Area Network Projects (E-book, Elektor)**
www.elektor.com/18219

WEB LINKS

- [1] JC3248W535 ESP32-S3 Based Development Board, AliExpress: https://s.click.aliexpress.com/e/_mM4CGkP
- [2] ESP32 Development Board (Cheap Yellow Display): <https://www.elektor.com/20890>
- [3] ELM327 Module, AliExpress: https://s.click.aliexpress.com/e/_mrXPLWP
- [4] Florian Schäffer, "OBD2: Retrofitting a Tachometer and Shift Light in your Car," Elektor 7-8/2025:
<https://www.elektormagazine.com/250005-01>
- [5] OBD2 Dashboard, GitHub: <https://github.com/ElektorLabs/obd2-dashboard/>
- [6] OBD2 PIDs for Programmers, Total Car Diagnostics:
<https://www.totalcardiagnostics.com/support/Knowledgebase/Article/View/104/0/obd2-pids-for-programmers-technical>
- [7] OBD2 PID Interactive Converter Tool, CSS Electronics:
<https://www.csselectronics.com/pages/obd2-pid-table-on-board-diagnostics-j1979>
- [8] Arduino GFX library: <https://docs.arduino.cc/libraries/gfx-library-for-arduino/>
- [9] Figma Design: <http://www.figma.com>

JOIN OUR COMMUNITY

GET FREE
DOWNLOAD



Subscribe today at elektormagazine.com/ezine-24





OBD2: Add a Rev Counter and Gear Shift Indicator to Your Car

Retro, but Super Useful

By Florian Schäffer (Germany)

If your car uses a manual gearbox, you can save fuel and keep exhaust emissions to a minimum if you don't overrev your engine. A shift light mounted on the dash can tell you exactly when it's time to upshift. The best part? You can easily add one to almost any car via the OBD2 port.

Back in the day, a rev counter was often seen as more important than the speedometer. If you want to drive more efficiently — or just a bit sportier — keeping an eye on your engine's RPMs is key for knowing when to change gear. Many economy cars don't come with a tachometer, and that's where a shift light can really come in handy. It's basically a light that flashes when it's time to shift up. These gadgets used to be mocked as gimmicks, but they might be making a comeback. Most modern cars are now so quiet (especially diesels) that it's hard to tell how hard the engine's working just by listening.

If your car doesn't have a rev counter or shift light and you think it might be useful, you can add it without even popping the hood. All you need is a low-cost OBD2 diagnostic module and some electronics to display the data.

When's the Right Time to Change Gear?

The best time to change gear can be determined by studying an engine's performance characteristic curves (see **Figure 1**). These curves are generally quite similar, irrespective of the various road vehicle engine manufacturers. The big difference in the characteristics is between engines designed to run on gasoline and those that run on diesel. Even though engine *power* is what manufacturers love to highlight in their literature, the level of *torque* developed by an engine is a more useful indicator when it comes to changing gear.

On these curves, you'll see engine speed (RPM) along the x-axis. For a gasoline engine, the output power in KW (red line) climbs steadily from about 1,500 to 5,000 RPM, while the torque in Nm (blue line) is reasonably flat beyond 1480 RPM. That means for this engine, there is no particular band you can use the gearbox to exploit while accelerating.

Diesel engines are a different story. They produce a lot of torque (green line) at low RPM, which is great for acceleration, but only if you stay in a narrow RPM range, in the power band around 1,800 RPM (lilac line). With diesels, you need to shift through the gears more quickly to keep the engine running in that sweet spot.

The perfect time to shift is when your engine hits the RPM where it makes peak power without the torque dropping off. For most gasoline engines, that's around 4,500 RPM while for diesels, it's about 1,800 RPM. That's also when the gear shift light comes on, if you have one.

You can usually find a chart that graphs these output performance curves for your own engine. This will allow you to double-check the recommended shift points.

This all mostly applies when you're accelerating. Downshifting isn't so important. You really only need to use it in two situations:

- **Quick acceleration** — For overtaking. Downshifting bumps up your RPM, which might drop torque a bit, but increases power and gives a speed boost for safer overtaking. You can upshift again as the car accelerates.
- **Slowing down** — You can save fuel by *not* downshifting too soon. Wait until the engine is almost at idle speed before going to the next lower gear. That way, your car goes into "fuel cut-off mode," where the engine burns no fuel at all. The engine and gearbox act like a brake, to slow you down. This can feel a little jerky in gasoline cars because they tend to use flywheels with less rotational mass than diesel engines, which helps to smooth out sudden engine speed changes.

Just one thing to watch when going downhill. You need to be sure the engine braking is strong enough to actually slow the car down. If the gear selected is too high, gravity can actually overcome engine braking, and you'll still pick up speed.

How Can We Access the Data?

These days, getting live data from your car such as engine RPM is easier than ever, thanks to the OBD2 port (that's the little connector usually found under the dashboard). Through it, you can access RPM information along with a whole bunch of other engine data.

The first company to bring an easy-to-use OBD solution to the public was ELM Electronics in Canada. They created a series of chips that could talk to all the different car communication systems and protocols, and they made them super simple to use via a serial interface. Their most popular chip was the ELM327, which became a sort of standard.

ELM, unfortunately, did not enable the copy protection feature of the PIC controller, allowing pirates to simply avoid development costs, lift the code and pump out their own ultra-cheap OBD reader knock-offs. These copies, costing as little as €5, flooded the market. Some versions feature USB, Bluetooth, and Wi-Fi connectivity. The downside? There's been no further development, and even bugs identified a long time ago in the original hardware and software have ended up in the cloned versions. The clones work well enough and have become so popular that ELM Electronics lost their market and sadly had to shut up shop in 2022.

Lots of Data

With an ELM327-based adapter, there are now loads of smartphone apps that will let you see what your car is actually doing in real time. The exact data you can get depends on your car's engine and its engine control unit (ECU), but there is often a lot more than you'd expect. Originally, there were around 20 data points called Parameter Identifiers (PIDs), but that number has grown to over 100, reflecting how many

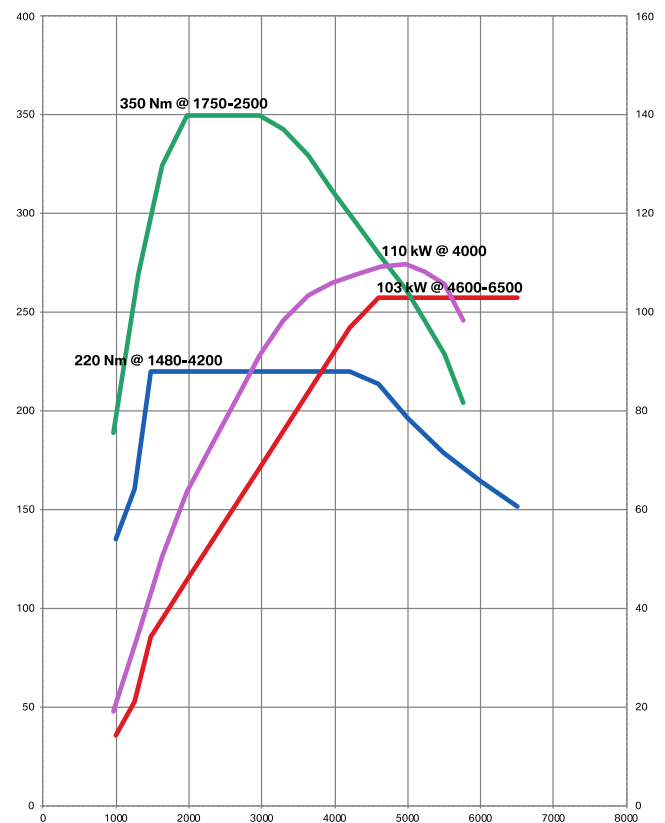


Figure 1: Typical gasoline and diesel internal combustion engine performance characteristics (torque in Nm and power in kW against engine speed in RPM). Blue/Red: torque/power of gasoline engine. Green/Lilac: torque/power of diesel engine.

more measurements advanced modern vehicles rely on. You can check out engine stats like coolant or oil temperature (which most cars don't even show on the dashboard), your actual road speed value and fuel consumption data (especially in newer cars). For petrol heads, it's a treasure trove of information, as long as your car supports the data.

The good news is that engine RPM information is always available at the OBD port. If you want to build a custom rev counter or gear shift light, you're good to go. The RPM data is made up of two bytes and is available in the form: PID $0C_{Hex}$ as a 16-bit value. These two bytes can then be plugged into a simple formula to get the actual value of RPM.

The RPM value is defined to be within the range from 0 to 16,383.75 RPM and is given by:

$$RPM = \text{Diagnostic value} / 65,535 \times 16,383.75$$

If the first byte, Byte (A), has the value $0C_{Hex}$, for example, and the second (B) has the value $B8_{Hex}$, we combine them together to form a 16-bit value $0CB8_{Hex}$. Plugging this value into the formula gives an engine speed of 814 RPM. The example code described in the article uses a slightly different method to calculate this, but gets to the same result in the end, even though it doesn't follow the official OBD spec exactly.

A Single Indicator

Using a smartphone to display engine data while driving isn't exactly ideal — it's bulky, distracting, and just not very practical. A simpler, more focused display works way better. All you really need is an indicator light to tell you when to change gear; you can skip the digital RPM display entirely and just use a single LED light, mounted somewhere on your dashboard.

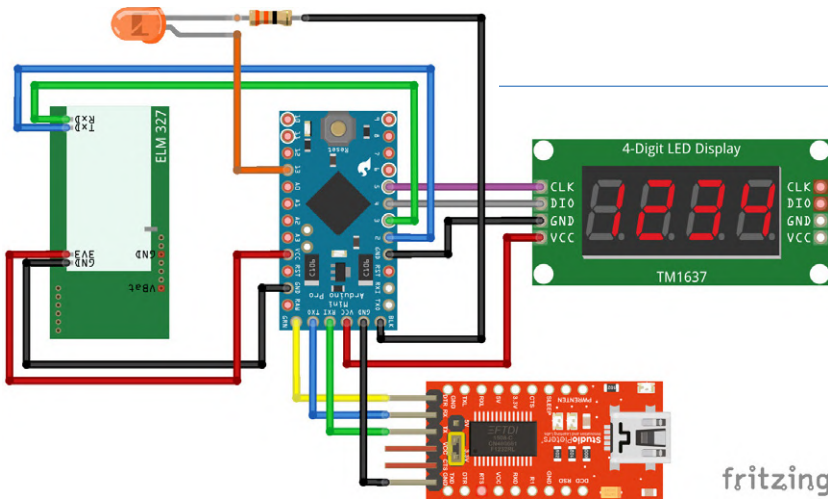
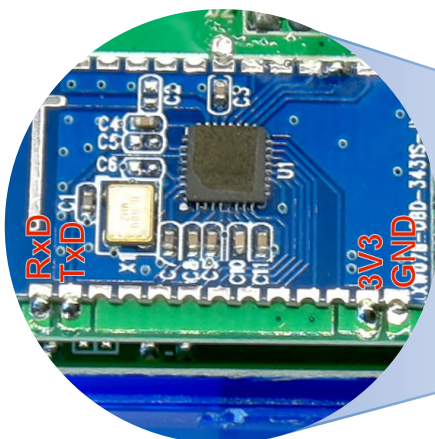


Figure 2: Block diagram of the OBD2 rev counter display with gear change indicator. The red USB adapter is only required for programming.

Figure 2 shows the basic layout of a DIY RPM display with a gear shift light. This project uses a cheap ELM327 clone from China to handle communication with your car. The great thing about these modules is that they include not only the ELM chip (which grabs and formats the data) but also all the circuitry needed to support various car communication protocols. To handle the logic and display output, the project uses an Arduino Pro Mini (the 3.3 V version works best with the generic ELM clones). The Arduino reads the RPM data via UART and displays it on a simple four-digit numeric display. It also controls the shift light LED according to the engine speed.

Choosing the Right ELM Clone

First, you'll need an ELM327 interface. Since the demise of ELM Electronics, there's now not so much guilt in using one of the many knockoffs floating around. The easiest route is to buy a cheap prebuilt OBD2 interface and do a little hacking. The ELM chip was originally designed for serial communication with a PC and had an integrated CAN controller. That's why all the clones still only support serial connections. That's not really a big deal; there are plenty of modules that can send serial data over USB, Bluetooth, or Wi-Fi. To build the display setup, you can just tap into the ELM's UART pins. USB versions are now rare and usually overpriced, so the simplest solution is to opt for a Bluetooth OBD2 adapter. These come in small thumb-sized housings that can be easily popped open. Remove the Bluetooth module (which you won't need for this project) and wire the ELM board directly to the Arduino's serial communication pins (see **Figure 3**).



fritzing

These clones are available in a variety of colours, such as transparent blue and some can be three times more expensive. Under the hood, however, it's just the same pirated ELM chip in every module. Don't pay too much to attention to marketing hype quoting version numbers or fancy names, those are mostly invented by sellers to make their own products sound more attractive. Some newer versions use external CAN controllers like the MCP2515 or even chip-on-board technology (COB) [2]. Be cautious: one Wi-Fi module I tested didn't support all OBD protocols, even though it claimed to. Avoid modules where the wireless unit is fully integrated with the ELM chip, you won't be able to access the serial pins, which would make it useless for this project.

The most reliable setup that I found can be built using a model that consists of two circuit boards with a clearly separate (usually blue) Bluetooth module. Look closely at product photos when buying the adapter and use the return option if what arrives doesn't match. Some listings claim the dual-board design is a new innovation, but it isn't. Just keep in mind, even this version doesn't support every protocol. It can't handle J1850 VPW, a rare and outdated standard mostly found in old GM and Chrysler vehicles.

Lastly, this project intentionally avoids any wireless communication between the ELM and Arduino for good reason. Wireless connections can be flaky in a car environment and introduce unnecessary hassle. There's also a potential security risk: a constantly active wireless ELM device could, in theory, be a backdoor into your vehicle's control network and may be exploited by ne'er-do-wells to unlock your car or cause other mischief.

A Look Inside the ELM Clone

Most ELM327 clone housings can be popped open pretty easily. Once inside, you'll usually see a Bluetooth PCB module soldered in place, it's typically connected at five points. You can desolder it without too much trouble, but be extra careful with the RxD and TxD pads, since they aren't connected to pin headers (see the zoomed-in view in **Figure 3**).

To program your Arduino, you'll need a USB-to-TTL adapter. Just hook it up as shown in the schematic (**Figure 2**). In the Arduino IDE, select



Figure 3: The blue PCB with the Bluetooth Module is not needed and can just be desoldered.

What Does the Law Say?

A frequently debated question: *Am I allowed to do this, and will it cause problems?* The short answer is: yes and no. Some countries are planning to restrict access to OBD port data to prevent misuse of data by third parties. This is more of a data protection issue. Some legislatures do not allow permanent, fixed installations in vehicles, at least not without approved components and a technical inspection. If any device plugged into the OBD port can be removed with a single action, then its use is acceptable. The rev counter can, of course, just be unplugged before your vehicle goes in for its next check-up. We do, of course, plug accessories into the car's electrical system all the time without giving them a second thought.

The second part of the question is a bit trickier. According to ISO standards, it's the vehicle manufacturer's responsibility to ensure that continuous access to OBD2 does not impair the car's functionality or drivability in any way.

In the early days of OBD2 port implementation, there were stories of control system crashes, causing the instrument cluster to switch to disco mode with all displays flashing at random rates. The solution (as ever, where computers are involved) was to remove the diagnostic device, turn the ignition off, and back on again. Continuous data monitoring via an external reader was problematic.

Stories of more serious malfunctions of the braking system or other horror scenarios, however, are nothing more than urban myths.

the usual settings for a Pro Mini: ATmega328, 3.3 V, and whatever programmer you're using. Make sure the Serial Monitor window is not open while uploading the code; otherwise, you may run into upload errors.

This project uses a seven-segment LED display with a TM1637 controller. The display is available in different colours and works well both at night and in bright sunlight. If you'd rather not use such a display, feel free to skip it or swap it for something else and tweak the code accordingly.

For the gear shift light LED indicator, go with something super bright, such as 10,000 mcd. That's plenty to catch your eye, even if it's not mounted directly in your line of sight, just don't stare straight into it. A reflective LED mount helps make it even more visible. As for the LED series resistor, you'll need to calculate the right value based on your LED's specs. Just make sure the current draw doesn't go over 40 mA.

Power to the Unit

Powering electronics from a car's electrical system always comes with its headaches. While we generally accept that cars run on "12 V," that's really just a nominal value. In reality, the voltage can go up to 14.4 V when the engine is idling. During startup, you might even see huge voltage spikes (hundreds of volts!) or dips below 5 V. Whilst driving, the voltage level tends to fluctuate, and has high levels of electrical noise.

The designers of cloned ELM devices generally do not pay too much attention to mitigating the electrical noise issues. The unit needs to be cheap, so every possible corner has been cut to save costs. Most of the designs use the AMS1117 type of voltage regulator to bring 12 V down to 3.3 V. That particular regulator isn't especially robust: it can only handle up to 15 V input and its "1 A" output rating is also more wishful thinking than reality. On top of that, it has zero overvoltage protection, so if it fails, your whole circuit might fry from the application of raw battery voltage.

Another issue is that most OBD ports only supply constant power (always-on 12 V), not switched power (on only when the ignition is on). That means the ELM doesn't actually know when the car turns back on. Unless you unplug the unit after use, it will keep trying to initiate a new diagnostics session over and over. It doesn't draw much current

doing this, but it can interfere with your vehicle's control units, which may stay awake instead of going into sleep mode. Over time, this process can result in a drained battery.

If you want to know how your car responds in this situation, try measuring the current draw directly at the battery. Use a DC clamp ammeter with the bonnet, doors, and windows all closed (don't start the engine or touch the ignition key while testing!). After about two minutes, if the current draw from the battery with the rev counter unit attached is the same as without it, there is nothing to worry about.

The best solution is to change the wiring to the OBD connector. Pin 16 provides a continuous 12 V supply. This can be replaced by a cable carrying a switched supply so that the voltage drops to zero when the ignition is off. This mod will not affect how the socket is used in normal operation when a garage technician connects diagnostic equipment to the socket, because it can only communicate with the ECU when the vehicle ignition is on. In some older European VAG (Volkswagen/Audi) cars, a "Terminal X" supply is also available in the wiring harness and could alternatively be used to provide power to the socket. Terminal X only supplies 12 V after the engine starts running.

If you prefer to use an extension adapter cable between the OBD socket and the rev counter unit, it is possible to swap the wire to pin 16 in the OBD extension cable with either of the alternative switched supplies mentioned above. The rev counter unit will then be turned on and off along with the ignition.

The Program Code

You can download the full program for this project from Elektor Labs [3]. Using the *ELMduino.h* library included in your Arduino sketch makes talking to the ELM327 module really simple. You just need a few function calls, and you're good to go. The same goes for the *TM1637Display.h* library used for outputting data to the 7-segment LED display. Both libraries are available via the Arduino IDE's Library Manager and can be installed in just a few clicks.

The source code is well commented, making it easy to follow and understand how everything works. At the start, there is a section for customization. This is where you can set the target RPM for the shift light to trigger (default: 1850 RPM), and how long the LED should

stay on (suggested value: 3000 ms, or 3 seconds). The RPM data can technically be polled rapidly (depending on the OBD protocol), but that would make the display flicker a lot. By default, there's a 200 ms delay between updates, which makes the displayed information less jumpy. You can even choose to display a dot as a thousands separator if you like.

Further configuration options include ELM communication settings, display characters, serial port assignments for the ELM, and data lines connection assignments for the 7-segment display.

A timer interrupt (Timer 1) fires every 200 ms to flash the decimal points on the display. That way, you can tell at a glance that the system is active and attempting to establish an OBD connection. If you enable debugging in the code, you can use the serial monitor in the IDE to check for test outputs and current RPM readings.

Once the connection with the ELM is established, the sketch enters a loop that continuously fetches the RPM data. It processes the data for display formatting, such as adding a separator or rounding the last digit to zero so that the readout is more stable. (An ultra-precise RPM readout isn't that useful for our purposes and just makes the displayed values flicker.)

The most important part of the code checks whether the current RPM has reached or exceeded the configured threshold. If it has, the gear shift LED lights up and then turns off after a configurable delay period elapses. The program locks out any further blinking until the RPM drops back below the threshold, so you don't get constant flashing while accelerating or cruising in the highest gear.

The sketch also checks for communication errors during OBD polling. If something goes wrong, it attempts a quick reconnect of the ELM327 to the OBD so the display keeps working without interruptions. If it detects a more persistent error (for example, when the car's ignition is turned off), it clears the display and waits for a longer interval before trying again. This keeps the system from needlessly polling the OBD when the ignition is turned off.

Installation in Your Car

Housing all the components and making a neat installation can be a little tricky. Your best bet is probably to design and 3D print your own enclosure or use a commercial OBD plug and enclosure (**Figure 4**). In this setup, the original blue ELM case was opened with a Dremel and replaced with a custom OBD plug that also has a matching housing [4]. ◀

Edited by Rolf Gerstendorf / Translated by Martin Cooke — 250005-01

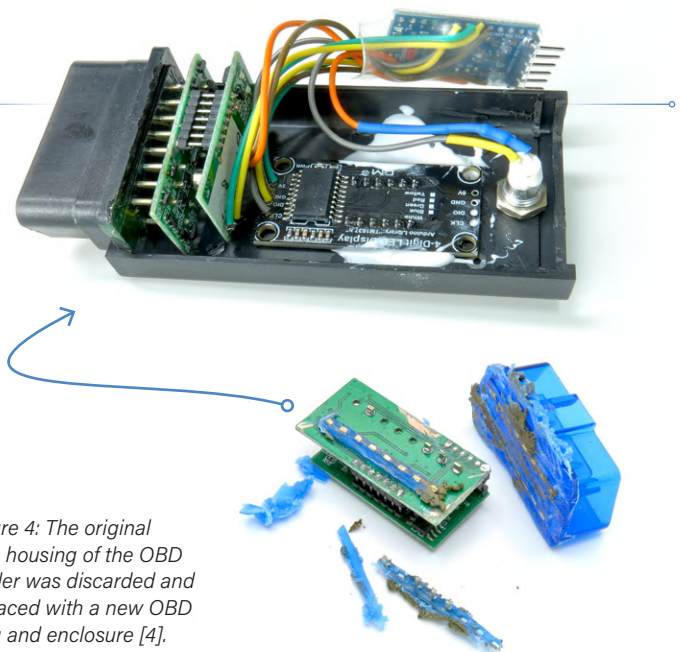
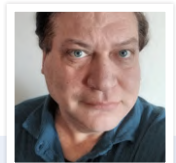


Figure 4: The original blue housing of the OBD reader was discarded and replaced with a new OBD plug and enclosure [4].

Questions or Comments?

If you have any technical questions or comments regarding this project please contact the author at frage@obd2-shop.eu or the Elektor editorial team (editor@elektor.com).



About the Author

Florian Schäffer has been working as an author of nonfiction books and a freelance editor for over 30 years. His interests include electronics and programming microcontrollers. Years ago, he got stuck into computer-based car diagnostics to fix his own car. Since then, he has been exploring the possibilities of vehicle OBD and even runs his own online shop.



Related Products

- **Seeed Studio OBD-II CAN Bus Development Kit**
www.elektor.com/20236
- **FNIRSI DMC-100 True RMS Smart Clamp Multimeter (600 A)**
www.elektor.com/21114

WEB LINKS

- [1] Saad Imtiaz, "OBD2 Sensor Dashboard," Elektor 7-8/2025: <http://www.elektormagazine.com/250107-01>
- [2] Chip on board, Wikipedia: https://en.wikipedia.org/wiki/Chip_on_board
- [3] Download: <https://www.elektormagazine.com/labs/elektor-articles-software-downloads>
- [4] Modular OBD-2 system: https://www.obd2-shop.eu/index.php/cPath/21_71/language/en

LiDAR and Vision Sensors for Robotics

By Saad Imtiaz (Elektor)

As sensing technologies continue to evolve, they are quietly shaping how robots perceive and interact with their environment. This article looks into a range of LiDAR sensors, and depth cameras that are commonly found across robotics projects — from research and education to more advanced applications. It also touches on software ecosystems, typical use cases, and a few higher-end options that hint at what's possible as these tools become more widely adopted.

In the rapidly evolving world of robotics, a machine's ability to perceive and understand its environment is critical to its success. From autonomous vehicles navigating complex terrains to factory robots collaborating alongside humans, the fusion of advanced sensors has enabled remarkable progress. Today, technologies like LiDAR, depth-sensing cameras, and AI-enabled vision systems are no longer reserved for industrial giants — they are increasingly accessible to engineers, students, and makers alike.

As part of this growing accessibility, a range of reliable sensors — such as those from YDLIDAR [1], which is available via the Elektor Store — have made it easier than ever to get started with robotics projects or scale them up with more capable hardware. These sensors are widely used in educational platforms and prototypes, offering a solid foundation for mapping, localization/navigation, and obstacle detection.

This article offers an in-depth exploration of some of the most popular and community-supported sensors available today, comparing their performance, software ecosystems, and ideal applications. Whether you're building a mobile robot, a smart surveillance system, or an autonomous drone, understanding these technologies will help you select the right perception tools to turn your vision into reality.

LiDAR vs. Cameras vs. Radar

Robotics engineers often combine multiple sensor types to cover the weaknesses of each. LiDAR, cameras, and radar all sense the environment but in very different ways. Here is a brief conceptual comparison of how they work and their best-use scenarios:

LiDAR (Light Detection and Ranging)

Light Detection and Ranging (LiDAR) works by emitting pulses of laser light (usually in the infrared spectrum) and measuring the time it takes for the reflections to return. This allows it to calculate precise distances and generate detailed 3D point clouds of the environment (**Figure 1**). As an active sensor, LiDAR performs equally well in daylight or complete darkness.

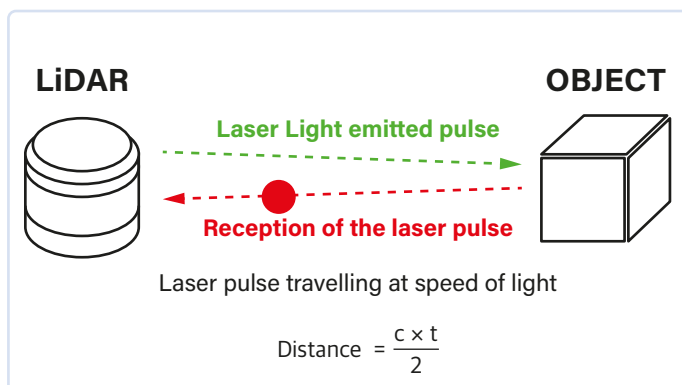


Figure 1: Working principle of LiDAR sensors.

In robotics, LiDAR is primarily used for mapping, SLAM (Simultaneous Localization and Mapping), and obstacle avoidance with high geometric precision (**Figure 2**). Mapping refers to creating a spatial representation of the environment — essentially a digital map of walls, objects, and terrain. SLAM goes a step further by allowing the robot to build this map while simultaneously estimating its own position within it, which is essential for navigating unknown or changing environments.

LiDAR is commonly mounted on top of mobile robots or vehicles to provide a 360° horizontal view (**Figure 3**), though smaller units can also be placed at the front or sides for more localized sensing. While it offers excellent resolution and accuracy, LiDAR can be affected by fog, rain, and dust, and tends to be more expensive and power-intensive than cameras. Nonetheless, it remains the sensor of choice for applications requiring detailed spatial awareness, especially in autonomous navigation.

Camera (Passive Vision) with AI

Cameras passively capture visible (or infrared) light, producing images. A standard camera on its own gives a 2D projection of the 3D world — meaning depth information is ambiguous (a far large object can appear the same size as a near small object). However, cameras excel at detecting colors, textures, and identifying object classes (with the help of AI algorithms). Modern robotics uses cameras with computer vision or deep learning to detect and interpret features (e.g., recognizing a human, a stop sign, or a spilled drink on the floor). Stereo camera setups or depth cameras (like structured-light or ToF) can infer depth, but with some limitations under certain conditions.

Cameras are cheaper and higher resolution than other sensors, and can leverage immense research from the computer vision field. They do require sufficient lighting (or IR illumination for night use), and their performance can drop in darkness or overly bright, shadowy scenes.

Best-use scenarios for cameras include object recognition, scene understanding, and when rich visual detail is needed. They are also used for approximate ranging (via stereo disparity or structure-from-motion), but generally are less direct and less accurate in measuring distance than LiDAR or radar. In sum, cameras with AI are indispensable for any task involving reading signs/text, recognizing specific objects or people, or in cost-sensitive projects (since cameras are inexpensive and ubiquitous).

Depth Cameras: Bridging Vision and Distance

Depth cameras combine the advantages of traditional imaging with the ability to perceive three-dimensional space, offering a middle ground between LiDAR's precision and standard camera's visual richness. They typically use technologies like stereo vision, structured light, or time-of-flight (ToF) to estimate distances to objects by analyzing how light patterns or disparities change across a scene (**Figure 4**). In ToF systems, the camera emits modulated



Figure 2: The real-time point cloud of automotive-grade solid-state Robosense LiDAR M1. (Source: Robosense)

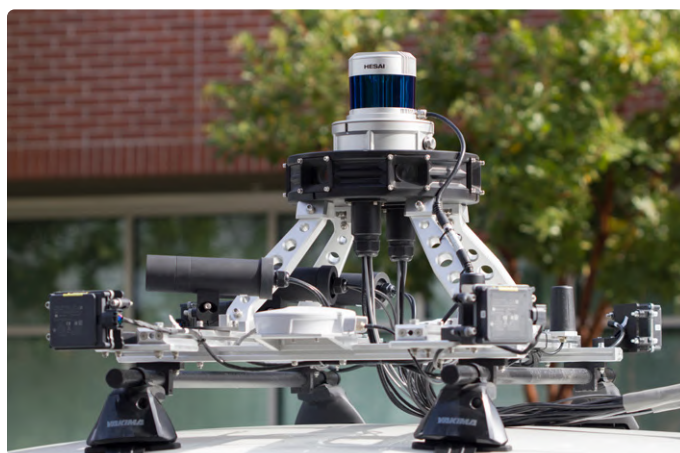


Figure 3: LiDAR sensor mounted on the top of an automobile. (Source: Adobe Stock / Tada Images)

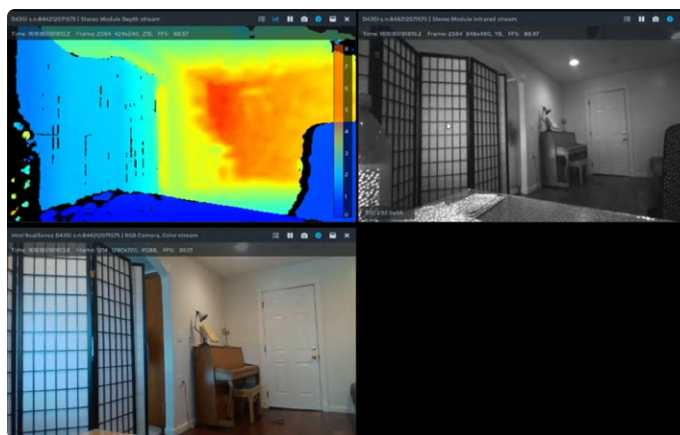
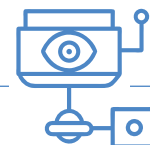


Figure 4: Intel D435 Depth Camera in Action. (Source: Intel)



Technologies and Use Cases

Technology	Strengths	Limitations	Typical Use Cases
Camera	High-res imagery, object recognition	No depth without AI/stereo, lighting-dependent	Visual tracking, classification, surveillance
LiDAR	Precise distance, 3D mapping, long range	Expensive, affected by weather/glare	SLAM, autonomous driving, high-fidelity environment modeling
Depth Camera	Dense 3D data, affordable, RGB+Depth	Range-limited, sunlight sensitivity (structured light)	Indoor navigation, gesture recognition, object grasping
Radar	All-weather performance, velocity data	Low spatial resolution, sparse data	Collision avoidance, speed tracking, backup sensing

infrared light and measures the time it takes for the light to bounce back from surfaces — this delay is used to calculate distance with high precision. Structured light, on the other hand, involves projecting a known infrared pattern — such as a grid or dot matrix — onto the environment, and then using the deformation of this pattern when viewed by the camera to calculate depth. This requires an integrated IR projector and is typically optimized for indoor use. Depth cameras produce dense depth maps, enabling robots to understand not just what objects are, but exactly where they are relative to the sensor. They are passive or semi-active systems (depending on whether they project IR patterns) and often work best in controlled lighting conditions.

Depth cameras are ideal for indoor SLAM, gesture recognition, object manipulation, and 3D scanning, providing a detailed spatial context at relatively low-cost and high resolution compared to LiDAR. However, they can struggle outdoors under direct sunlight (especially structured-light sensors) or over very long distances, and depth accuracy typically degrades beyond 5 to 10 m. Best-use scenarios for depth cameras include building mobile robots for indoor navigation, interactive robots that recognize hand gestures, warehouse automation for object picking, or AR/VR applications needing detailed room-scale mapping.

Radar

Radar uses radio waves (often in the microwave band, like 24 GHz or 77 GHz in automotive radar) and measures their reflections. Radar can penetrate fog, rain, and dust far better than optical sensors — weather has little effect on radar’s longer wavelengths. It’s also an active sensor (emits its own signals), working in darkness and bright light alike. The trade-off is that radar has much lower spatial resolution — it doesn’t create a detailed image, but rather detects the position and velocity of objects (often outputting a sparse set of detected targets or a coarse point cloud).

In robotics and vehicles, radar is fantastic for long-range detection and speed measurement (Doppler returns give direct relative velocity of targets). For example, an autonomous car’s radar can detect another vehicle at more than 200 m even in heavy rain,

where a camera or LiDAR might struggle. However, it might only tell “there’s an object at 150 m ahead in lane” but not detail its shape or type. Radars are also usually limited to planar scanning (automotive radars have a narrow vertical field).

Best use cases for radar are in adverse visibility conditions or as a safety overlay for long-range obstacle detection (e.g., drones or cars use radar to sense far-away obstacles or approaching objects at high speed). In environments with lots of metal or reflections (indoors, machinery), radar readings can be noisy. In short, radar adds robustness in weather and long-range speed sensing, complementing cameras and LiDARs.

In practice, sensor fusion is key — many robots use all three: cameras (for understanding what objects are), LiDAR (for where objects are in 3D), and radar (for reliable detection in bad conditions and velocity tracking). Each compensates for the others’ weaknesses.

For example, an autonomous car might use a camera plus AI to read a street sign, lidar to precisely map its surroundings in 3D, and radar to keep tracking a vehicle in heavy rain where the other sensors falter. Understanding these trade-offs helps in choosing the right sensor mix for a given robotic application.

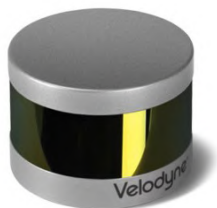
For a summary of how LiDAR, depth cameras, and radar differ in function and use, see the **Technologies and Use Cases** textbox.

LiDAR Sensors (and a ToF Camera)

LiDAR sensors produce 2D or 3D point clouds that are commonly used in robotics for SLAM, obstacle avoidance, and spatial navigation. Depending on the model, these sensors output data via USB, UART, CAN, or Ethernet. High-resolution units typically require processing on a PC or embedded system like an NVIDIA Jetson, while lighter-weight models are often compatible with Raspberry Pi boards and, in some cases, even microcontrollers — making them versatile for a range of robotics platforms. Below are some LiDAR sensors widely used in such applications:

Ouster Velodyne VLP-16 (Puck Hi-Res)

A professional-grade 16 to 32 channel 3D LiDAR with a 100 m range and 360° horizontal view, producing up to 600,000 points/s [2]. Its compact, weatherproof design (IP67) makes it ideal for autonomous vehicles and outdoor robotics. Supported by robust ROS drivers — where ROS (Robot Operating System) is a widely used open-source framework for developing and integrating robot software — and Ouster's SDK, it's widely used in research and industry for high-fidelity SLAM and mapping.



Source: [2]

Slamtec RPLIDAR A1

A budget-friendly 2D LiDAR offering a 12 m range with up to 8,000 samples/s [3]. Ideal for indoor robots and educational SLAM projects, it features USB connectivity, open SDKs, and strong ROS community support. Its low cost around \$100 and plug-and-play setup make it a go-to for hobbyists and students.



Source: [3]

YDLIDAR TG30

This is a compact, 360° 2D LiDAR sensor based on Time-of-Flight (ToF) technology, capable of measuring distances up to 30 m with high accuracy and stability. It operates at up to 20,000 samples/s with a scan frequency between 5 and 12 Hz, offering angular resolutions as fine as 0.09° at lower speeds. With IP65-rated protection, strong resistance to ambient light, and support for ROS and a dedicated SDK, it's a robust choice for indoor navigation, SLAM, and educational robotics. The TG30 outputs data via UART and can be easily connected through USB using the included adapter. It's available in the Elektor Store along with order variants from YDLIDAR [4], making it an accessible and well-supported option for anyone starting or scaling up their robotics projects.



Source: [4]

Slamtec RPLIDAR S1

A long-range (up to 40 m) 2D Time-of-Flight LiDAR designed for indoor and outdoor use. It captures over 9,000 points/s and resists sunlight interference — ideal for large-area SLAM or outdoor navigation. The S1 is compact, durable, and ROS-supported, offering premium performance at a mid-tier price [5].

VSemi Sentinel — 3D ToF Camera

The VSemi Sentinel [6] is a compact Time-of-Flight (ToF) 3D camera designed for near-range depth sensing. It offers a resolution of 160 × 60 pixels and operates effectively within a range of 0.1 m to 7.5 m. With a field of view



Source: [6]

of 51° × 20° and an accuracy ranging from 2 cm to 2% of the measurement distance, it's suitable for applications like gesture recognition and indoor robotics. The device is compatible with ROS, Linux, Windows, and Mac platforms, facilitating integration into various systems.

Livox MID-360

The Livox MID-360 is a hybrid solid-state LiDAR sensor offering a 360° horizontal field of view [7]. It's optimized for low-speed robotics, providing advanced 3D perception for navigation and obstacle avoidance. The sensor is compact and lightweight, making it easy to install on mobile platforms. It operates effectively in various lighting conditions and supports integration through the Livox SDK and Livox Viewer 2 (a software to view the data).



Source: [7]

RoboSense Airy

The RoboSense Airy is a compact hemispherical digital LiDAR featuring a 360° horizontal and 90° vertical field of view [8]. With 192 beams and a point cloud output of 1.72 million points/s, it delivers high-resolution data for detailed environment mapping. The sensor weighs under 240 g and consumes less than 8 W of power, making it ideal for mobile robotics applications requiring continuous operation.



Source: [8]

RoboSense E1R

The RoboSense E1R is a solid-state LiDAR sensor designed for robotic applications [9]. It offers a 120° horizontal and 90° vertical field of view, with a detection range of up to 75 m. Built on an automotive-grade platform, it has passed over 60 rigorous reliability tests and operates within a temperature range of -40°C to +85°C. The sensor's robustness makes it suitable for various environments, including those with high vibration and shock levels.



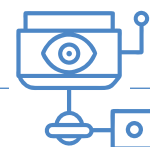
Source: [8]

An overview and side-by-side comparison of the LiDAR sensors discussed can be found in the **LiDAR Sensors Comparison** textbox (see next page).

Depth Sensing Stereo Vision Cameras

There are numerous depth-sensing cameras on the market today, each leveraging technologies like stereo vision, structured light, or time-of-flight. Among them, devices like the original Microsoft Kinect helped popularize affordable 3D vision in robotics and gaming, paving the way for the more advanced sensors in use today.

The five models highlighted here are commonly found in robotics projects, research platforms, and commercial systems. They offer strong technical performance and provide reliable digital



LiDAR Sensors Comparison Chart

Model	Type	Range	Field of View (FoV)	Points/s	Applications	Approx. Price
Velodyne VLP-16 (Puck Hi-Res)	3D LiDAR (16 channels)	Up to 100 m	360° H × 20° V	~600,000	High-fidelity mapping, autonomous navigation	~\$950–\$1,500
Slamtec RPLIDAR A1	2D LiDAR (Triangulation)	Up to 12 m	360°	Up to 8,000	Indoor SLAM, hobby robots, education	~\$99
Slamtec RPLIDAR S1	2D LiDAR (ToF)	Up to 40 m	360°	Up to 9,200	Outdoor SLAM, autonomous rovers	~\$649
VSemi Sentinel (3D ToF Camera)	3D ToF Camera	0.1 m – 7.5 m	51° H × 20° V	N/A	Gesture tracking, depth sensing, indoor robots	~\$199
Livox MID-360	3D LiDAR (Solid-State)	Up to 40 m	360° H	N/A	Mobile robotics, warehouse navigation	~\$1089
RoboSense Airy	3D LiDAR (Digital)	Up to 30 m	360° H × 90° V	1.72 million	Dense mapping, SLAM, drone navigation	~\$1,099
RoboSense E1R	3D LiDAR (Solid-State)	Up to 75 m	120° H × 90° V	260,000	Rugged robotics, outdoor automation	~\$999
Slamtec Slamkit & C1	2D LiDAR + SLAM Solution	Up to 12 m	360°	N/A	Mapping, localization, indoor AGV navigation	~\$380

output over USB or Ethernet, making them easy to integrate. Most are compatible with PCs, embedded platforms like the NVIDIA Jetson, and in some cases, even Raspberry Pi boards — though microcontroller support is typically limited due to bandwidth and processing constraints. Their robust software ecosystems, ROS compatibility, and active developer communities make them especially well-suited for robotics applications.

Intel RealSense D435

The Intel RealSense D435 is a versatile active stereo depth camera equipped with global shutter sensors, making it suitable for capturing fast-moving objects [10]. It offers a depth range of 0.3 m to 3 m and supports resolutions up to 1280×720 at 30 fps. With a wide field of view (85° × 58°) and compatibility with ROS, it's ideal for applications like SLAM, obstacle avoidance, and robotic manipulation.



Source: [10]

Intel RealSense D455

Building upon the D435, the D455 extends the depth range up to 6 m and enhances accuracy by increasing the baseline between the stereo sensors [11]. It maintains the same resolution and frame rate capabilities, offering improved performance in outdoor and long-range scenarios. The D455 is well-suited for mobile robotics and applications requiring precise depth perception over extended distances.

Stereolabs ZED 2i

The Stereolabs ZED 2i is a passive stereo camera that provides high-resolution depth sensing up to 20 m [12]. It features a wide field of view (110°) and integrates an IMU, barometer, magnetometer,

and temperature sensors. With support for ROS and SDKs in C++ and Python, the ZED 2i is ideal for 3D mapping, autonomous navigation, and AR/VR applications.

Orbbec Gemini 335Le

The Orbbec Gemini 335Le is a rugged, industrial-grade stereo vision depth camera designed for robotics in demanding environments [13]. It offers depth sensing from 0.25 to 20 m, with optimal accuracy up to 6 m, and delivers both depth and RGB streams at 1280×800 resolution using global shutter sensors. With an IP67-rated enclosure, Power over Ethernet (PoE), and support for ROS1/ROS2 and NVIDIA Jetson, it's well-suited for mobile robots, automation, and outdoor robotic applications.



Source: [13]

See the **Depth Sensing Cameras Comparison Chart** textbox for a technical and practical comparison of the most widely used depth cameras in robotics.

Why Some Sensors Cost More Than Others

When browsing sensor options, you'll notice huge price variations – e.g., a hobby RGB-D camera might be \$200, while a high-end 3D LiDAR can exceed \$10,000+. Several factors drive the cost:

- **Range and Power:** Sensors designed to see farther (or in tougher conditions) need more power and advanced components. A LiDAR that reaches 200 m uses more powerful lasers and sensitive detectors (often with cooling and precise timing circuits), increasing cost. For instance, a basic indoor LiDAR

Depth Sensing Cameras Comparison

Model	Depth Range	Field of View (H × V)	Resolution	Frame Rate	Key Features
Intel RealSense D435	0.1–10 m	85° × 58°	Up to 1280×720	Up to 90 fps	Global shutter, ROS support
Intel RealSense D455	0.4–10 m	87° × 58°	Up to 1280×720	Up to 90 fps	Extended range, improved accuracy
StereoLabs ZED 2i	0.5–20 m	110°	Up to 4416×1242	Up to 100 fps	Passive stereo, integrated sensors
Azure Kinect DK	0.25–5.5 m	120° × 120°	1024×1024 (depth)	Up to 30 fps	ToF sensor, advanced AI capabilities
Orbbec Astra	0.4–8 m	60° × 49.5°	640×480 (depth)	30 fps	Structured light, compact design

with 6 m range uses a low-power laser diode, whereas a 200 m automotive LiDAR uses amplified laser pulses and APD (avalanche photodiode) receiver arrays – clearly more expensive tech. Similarly, a depth camera like Microsoft’s Azure Kinect (around \$400) uses a sophisticated Time-of-Flight sensor to reach ~5 m with millimeter accuracy, whereas a \$150 structured-light sensor only works to ~0.8 m reliably.

- Resolution** (Channels / Pixels): Higher resolution means more sensing elements or more complex optics. In LiDARs, the number of laser channels (beams) is a big cost factor — a 64-beam Velodyne has 64 individual laser diodes and detectors, precision-mounted on a spinning assembly, which costs a lot more to manufacture than a single-beam RPLIDAR. Each additional channel also increases the data processing load (needing beefier onboard processors or FPGAs). For cameras, higher megapixel sensors or global shutter sensors cost more than low-res or rolling shutter ones. Depth cameras with higher resolution depth maps (e.g., 1024 × 1024 ToF arrays) will be pricier than those with 640 × 480.
- Processing and Intelligence:** Some sensors bundle significant onboard processing which adds cost. An AI camera like the Luxonis OAK-D has the Myriad X VPU – essentially a mini computer – inside it. This adds to the price, but the benefit is it offloads your main computer and simplifies software. High-end lidars often output processed point clouds in real-time, requiring built-in FPGA or ASIC processing. Cheaper sensors might output raw data for the user to process (shifting the cost to the user’s computing resources instead). The inclusion of IMUs, onboard calibration modules, or sensor fusion (e.g., cameras with integrated depth processing like the ZED) also raises price.
- Software Stack and Support:** When you pay for a sensor, especially industrial-grade ones, part of what you pay for is the robust software and support behind it. Industrial vendors (like Basler or Zivid for cameras, Velodyne for LiDAR) provide extensive SDKs, documentation, and customer support. They ensure drivers are maintained for new OS versions, offer calibration utilities, and have warranties. This “software assurance” cost is built into professional products. In contrast, a \$100 sensor from eBay might come with minimal documentation and no support, leaving you to reverse-engineer data or rely on the community.

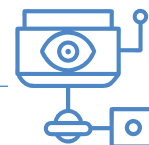
- Build Quality and Durability:** Robotics sensors that are meant for heavy use or harsh environments are built to higher standards (ruggedized enclosures, waterproofing, wide operating temperatures, shock/vibration resistance). For example, the Velodyne Puck is rated IP67 (waterproof) with an aluminum housing and has undergone extensive testing — this drives up cost. Cheaper sensors might be just plastic housings not meant for rain or long-term use. In industrial or automotive settings, companies are willing to pay more for sensors that reliably last tens of thousands of hours. Additionally, manufacturing tolerances (for optics alignment, etc.) and calibration of each unit add to cost. High-end devices are often individually calibrated and tested.

In summary, you get what you pay for: budget sensors are fantastic for hobby projects and early prototyping, but for applications needing long-range, precision, and reliability, the investment in a higher-end sensor pays off in data quality. It’s often a balance – many robotics teams use a mix of high and low-cost sensors to meet their needs without breaking the bank. For instance, a research robot might use a \$3000 LiDAR for mapping but a set of \$50 sonar or IR sensors as backup collision detectors at close range.

Industrial-Grade Sensor Vendors to Know

Finally, it’s worth knowing the key players in the sensor industry, even if their products are beyond the needs of a small project. In the LiDAR space, companies like Velodyne, Ouster, Hesai, and Livox are leading development. Velodyne (a pioneer since the DARPA Grand Challenge days) offers a range from the compact Puck to the Alpha Prime (128 beams, long range). Ouster focuses on affordable high-resolution digital lidars (with a single chip integrating many lasers). Hesai produces high-performance LiDARs used in many self-driving car tests. Livox (affiliated with DJI) takes a novel approach to scanning to reduce cost. These vendors are pushing technology forward.

For depth and 3D cameras, industrial machine vision companies like Zivid and Basler are notable. Zivid’s cameras (e.g., Zivid Two) are used in robotics picking systems for their high precision color 3D capture (accurate to 0.1 mm, but costing in the tens of thousands). Basler’s blaze camera uses Time-of-Flight to produce metric 3D data for logistics and factory automation, with the reliability



About the Author

Saad Imtiaz, Senior Engineer at Elektor, is a mechatronics engineer who has extensive experience in embedded systems and product development. His journey has seen him collaborate with a diverse array of companies, from innovative startups to established global enterprises, driving forward-thinking prototyping and development projects. With a rich background that includes a stint in the aviation industry and leadership of a technology startup, Saad brings a unique blend of technical expertise and entrepreneurial spirit to his role at Elektor. Here, he contributes to project development in both software and hardware.

one expects from German engineering. Intel RealSense remains widely used in robots and drones, and its open-source SDK lives on – RealSense exemplified how relatively low-cost depth cameras could be deployed in robotics at scale.

In the radar domain, if your application needs it, look at companies like Texas Instruments (they offer mmWave radar modules and chipsets that have been integrated into drones and robots) or automotive suppliers like Bosch and Continental that provide compact radar sensors (sometimes hackable for robotics use).

In conclusion, the landscape of LiDAR sensors and cameras for robotics is broad and evolving rapidly. The good news is that there are options for every budget and requirement – from a student building a small robot with a \$20 Pi Camera, to a researcher assembling a sensor suite rivaling a self-driving car's. By understanding the strengths of each sensor type and paying attention to specs like range, resolution, and support, you can mix-and-match the right sensors to give your robot the perception it needs. And as tools like the Tangram Vision LiDAR comparison tool [14], the community is actively helping make sense of all these options, so you're never alone in the journey of choosing sensors for your next robotics project. ◀

250341-01

Questions or Comments?

If you have questions about this article, feel free to email the author at saad.imtiaz@elektor.com or the Elektor editorial team at editor@elektor.com.



Related Products

- > **YDLIDAR X4Pro Lidar**
– 360-degree Laser Range Scanner (10 m)
www.elektor.com/20546
- > **YDLIDAR TG15 Outdoor Lidar**
– 360-degree Laser Range Scanner (15 m)
www.elektor.com/19237
- > **YDLIDAR G4 Lidar**
– 360-degree Laser Range Scanner (16 m)
www.elektor.com/18934
- > **YDLIDAR TG30 Lidar**
– 360-degree Laser Range Scanner (30 m)
www.elektor.com/20211

WEB LINKS

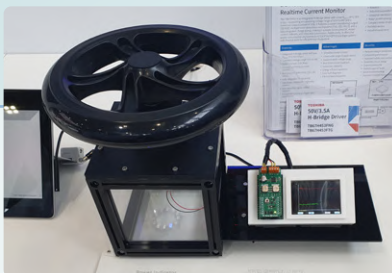
- [1] YDLIDAR: <https://www.ydlidar.com/>
- [2] Velodyne VLP-16, Ouster: <https://ouster.com/products/hardware/vlp-16>
- [3] Slamtec RPLIDAR A1: <https://www.slamtec.com/en/lidar/a1>
- [4] YDLIDAR TG15 Outdoor LiDAR: <https://www.elektor.com/products/ydlidar-tg30-lidar-360-degree-laser-range-scanner-30-m>
- [5] Slamtec RPLIDAR S1, RobotShop: <https://eu.robotshop.com/products/rplidar-s1-360-laser-scanner-40-m>
- [6] Sentinel 3D ToF Camera, VSemi: <https://vsemi.io/products/sentinel>
- [7] Livox MID-360: <https://www.livoxtech.com/mid-360>
- [8] RoboSense Airy: <https://www.robosense.ai/en/rslidar/Airy>
- [9] RoboSense E1R: <https://www.robosense.ai/en/rslidar/E1R>
- [10] Intel RealSense D435: <https://www.intelrealsense.com/depth-camera-d435/>
- [11] Intel RealSense D455: <https://www.intelrealsense.com/depth-camera-d455/>
- [12] StereoLabs ZED 2i: <https://www.stereolabs.com/en-pk/store/products/zed-2i>
- [13] Orbbec Gemini 335Le: <https://store.orbbec.com/products/gemini-335le>
- [14] Depth Sensor Visualizer, Tangram Vision: <https://www.tangramvision.com/resources/depth-sensor-visualizer>

2025 Sensor+Test and PCIM

From May 6 to 8, 2025, two fairs took place in Nuremberg, Germany: Sensor+Test [1] and PCIM Europe [2], dealing with power electronics. The PCIM has grown to six halls, with a lot of well-known companies joining, like ST, Infineon, Microchip, TI, and many others. This underlines that the market of high-power electronics is growing, with application fields such as electric vehicles, renewable energy, and server farms for artificial intelligence. Elektor's editor-in-chief, Jens Nickel, was on hand to look for interesting new products. As always, his personal selection is only a small sampling of all the innovations seen at these trade shows.

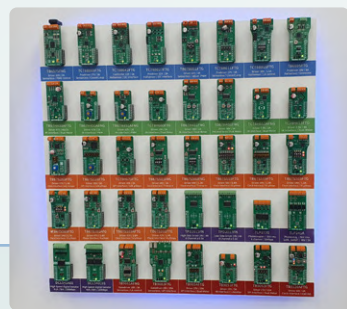
DC Motor Control

Motor control was, of course, a key topic at the PCIM. Toshiba showed a nice demo based on the brushed DC motor control chip TB67H453 with current sensing. When slowing down the motor with your hands at the wheel, the higher current was shown at the LED ring. Current feedback can be used for replacing position switches, cutting down the bill of materials and costs.



For easy evaluation of their motor control chips, Toshiba developed together with MikroElektronika a big set of Click boards. Find more about Toshiba at the PCIM at our Elektor TV Industry YouTube channel (see textbox).

<https://tinyurl.com/toshiba-motor-driver>



Non-Contact Temperature Measurements

Optris is specialized in industrial IR cameras and IR sensors, covering applications as surveilling large power facilities on one end to chip research on a microscopic scale on the other end. The German company has shown at the Sensor+Test a lot of products from their portfolio, for example the thermosensor on the left, here measuring the surface temperature of an aluminium cube. A simple "coating" — in that case an Edding marker spot — does the trick!

<https://optris.com/>



PCIM in Video

Engineer and Journalist Stuart Cording was also going around on the PCIM, as always with the professional camera man Michael "Mic" Klausner. Qoitech's Oti power analyzer was only one of the exciting products they reported on. Check out this video and a mashup of other company video reports at our Elektor TV Industry YouTube channel:

www.youtube.com/@ElektorIndustry



elektor TV





Ultrasonic Flow Meters

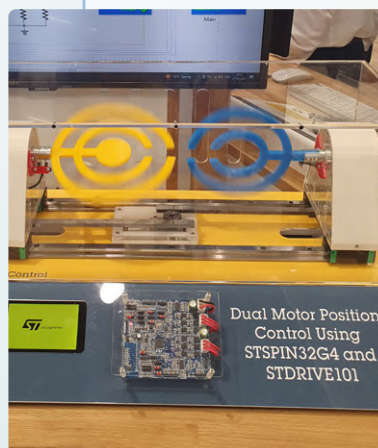
At the Sensor+Test show, flow meters formed a big segment, together with pressure and gas detection sensors. Germany-based Allengra specializes in ultrasonic flow meters, tailor-made or from the shelf. Applications range from cooling/heating and consumer products like coffee machines to fuel consumption measurements in motor sports. As the velocity of sound in liquids depends on the temperature, temperature sensors are integrated to enhance the precision. With the ultrasonic technology, bubble detection is also possible.

<https://allengra.eu/>

Dual Motor Control

A nice demo of two motors controlled in parallel was shown at the ST booth at PCIM (ST was also present at the Sensor+Test). The yellow and blue plastic objects fit together during fast rotation, if the motors on each side are precisely controlled. For this demo working together on a dual motor controller board: The STSPIN32G4 motor controller for driving 3-phase brushless motors and the STDRIVE101 as a triple half-bridge gate driver, controlling together 12 power MOSFETs.

www.st.com/content/st_com/en/events/pcim-europe.html



Weles Acoustics

As I am interested in audio projects, "sound maps" always attract my interest (see my report of Sensor+Test 2024 [3]). This system from Weles Acoustics is using a special sound probe to detect the sound pressure and particle velocity at different positions and in different frequency ranges. The scan must be processed manually; the system detects the position of the probe with two LEDs, in order to draw maps of sound intensity. This is interesting for production machine and automotive development, measurements of sound protection and many more applications.

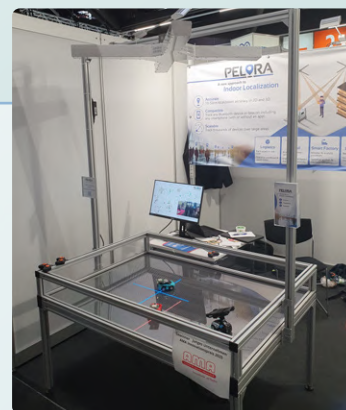
<https://weles-acoustics.com/en/>



Tracking Thousands of Devices

High-accuracy tracking with low-cost devices is a kind of holy grail of many IoT applications, like storage, traffic, and retail. The German start-up company Pelory uses an array of antennas to track Bluetooth chips (which are, for example, anyhow integrated in smartphones) with an accuracy of some centimeters. One antenna array can track thousands of devices at the same time, and with more antennas, you can cover big halls and places. According to the manufacturer, a German supermarket chain is highly interested, but it could also be interesting for fairs and events. Not only to see where the most successful booths are located, but also to navigate visitors to the exhibitor of their choice. Especially in Nuremberg with a kind of labyrinthine arrangement of halls, this could be indeed helpful!

www.pelora.io



250359-01

WEB LINKS

[1] Sensor+Test - The Measurement Fair: <http://www.sensor-test.de>

[2] PCIM Expo: <https://pcim.mesago.com/nuernberg/en.html>

[3] Jens Nickel, "Sensor+Test 2024 and PCIM 2024," Elektor 7-8/2024 Bonus Edition: <https://www.elektormagazine.com/iot-sensors>

Contact-Free E-Field Measurements (1)

A Vibrating Membrane for Assessing DC Voltages or Static Electric Fields

Source: Adobe Stock

By Michael Monkenbusch (Germany)

In this article — the first of a series — we are exploring the possibility of making a measuring device for static electrical fields, which thereby could also allow a completely contactless measurement of DC potentials of objects. As sensor for the field, a vibrating membrane is used. In the second installment, the amplitude of the membrane's vibration will be assessed, with the help of a laser-based makeshift vibrometer.

To minimize required mechanical work, a vibrating membrane (aluminum foil, or Al foil) was chosen as the sensor element. The vibration is generated by a miniature loudspeaker (sound transducer), which is coupled by a resonating air column to the membrane. The latter ensures sufficient distance between transducer electric fields and the sensitive membrane.

In practice, the column is realized by a cardboard or plastic tube of about 4 cm diameter and 20 to 30 cm length. The sensor membrane being a piece of Al-foil glued across one opening; the miniature loudspeaker attached to the other end of the tube. This results in a resonance at a frequency between 400 to 1000 Hz, which is employed during the measurement.

The membrane is connected to ground via a 100 kΩ to 1 MΩ resistor R, thus a vibration in the presence of an electrical field induces a (small) current, which is measured by taking R as a shunt (i.e., the voltage across R is measured). This yields a signal that is proportional to the electrical field at the Al-membrane.

To assess the vibration amplitudes of membranes in this type of contraption, a makeshift laser-interferometer has been set up. This enables comparison of the amplitudes with those inferred from the E-field signal. This will be the subject of the next installment of this article.

Signal

For a simplified consideration on how the measured (current) signal depends on the electric field, E, or the potential (voltage), U, of a nearby object, we assume that the membrane oscillates like a stiff plate with a uniform amplitude, a, and a frequency, ν. For that, we may discuss two scenarios:

Adjacent (parallel) plate at potential U

Here we assume a parallel plate with distance d between plate and our sensor membrane with an area A. Together they form a capacitor C with a capacitance value of:

$$C = \frac{\epsilon A}{d}$$

This capacitor then holds a charge Q:

$$Q = U C$$

The membrane vibration causes periodic changes of the plate distance d and thereby periodic changes in C and corresponding changes δQ in the stored charge Q. These changes imply a current:

$$I(t) = \frac{\partial Q}{\partial t}$$

Therefore, with a plate vibration expressed by:

$$d(t) = d_0 + a \sin(2\pi\nu t)$$

we have

$$Q(t) = U \frac{\epsilon A}{d(t)}$$

i.e.,

$$I(t) = \frac{\partial Q(t)}{\partial t} = -U \frac{\epsilon A}{d^2(t)} \frac{\partial d(t)}{\partial t} \cong$$

$$\cong -U \epsilon A \frac{1}{d^2} a 2\pi\nu \cos(2\pi\nu t)$$

which means that the current signal $I(t)$ is a sine (cos) of frequency ν with an amplitude proportional to potential (difference) U and inversely proportional to the square of the distance d . The sensitivity depends on the product of the (effective) membrane area A , the vibration frequency ν and the (yet unknown) vibration amplitude a .

Electrical field

An alternate view of the situation may start with a given electrical field at the surface of the membrane (e.g., resulting from some charged object in the vicinity). If the membrane moves a distance δx along the electrical field lines, its potential (voltage) should change by $\delta U = E \delta x$. The voltage change in the capacitor C , however, is compensated by a current flow in the shunt resistor R to ground. Again, the associated charge is $\delta Q = C \delta U = C E \delta x$ and since $\delta x = a \sin(2\pi\nu t)$ leading to a current:

$$I(t) = CE a 2\pi\nu \cos(2\pi\nu t)$$

This may be compared to the previous result if we observe that $E = -U/d$ and $C = \epsilon A/d$.

Note, however, that C is the "mutual" capacitance between membrane and charged object plus free space; any extra capacitance added by connecting cables and other circuit components has little or no influence on the signal, as long as a $C_{\text{total}} R \ll 1/\nu$.

In the real world

For the present realization the membrane displacement is not that of a rigid vibration plate but rather that of a membrane with fixed boundary, such that the displacement must be averaged over the area, yielding an average that has a value of about 10% of the maximum central displacement amplitude.

The amplitude (and the phase) of the displacement is frequency-dependent, and for closer plate distances d depends on this distance (as well as the exact resonance frequency).

The presented construction incorporates measures and strategies to cope with these difficulties to enable also quantitative results. Later, the inferred membrane amplitudes were compared to results from a laser interferometer setup.

Challenges

While the basic measuring task is clear and consists of simply determining the amplitude of the induced current $I(t)$, the practical realization poses a number of challenges:

- > Isolate any electrical signals emerging from the vibration drive from the sensitive membrane.
- > Deal with amplitude relations (changes) and phase shifts between the drive and the membrane signal.
- > Suppress the influence of external field noise, in particular environmental 50 Hz electrical grid "fog".

The presented design addresses these by:

- > Establishing coupling between drive actuator and sensitive membrane over 10 to 30 cm distance by a resonant air column. The best (resonant) frequency is determined at start by a scan (Arduino program).
- > Using a two-channel phase synchronized detection employing two 90° phase-shifted reference signals. From the two detector voltages, the amplitude and the phase angle of the membrane current can be computed (Arduino program). The phase yields the sign of the applied field.
- > To extract the sign (positive and negative polarity are associated with a 180° phase difference) and magnitude of the voltage of the object under test in a calibration sequence, we observe the signal change upon charging the membrane with a known potential. This decouples the determination of the test object voltage from exact knowledge of the membrane distance and stray capacitance.
- > Besides the phase sensitive detection using a frequency that is different from multiples of 50 Hz, the diagram contains a 50 Hz notch T-filter.

Circuit Diagram

The circuit diagram of the electric field sensor is shown in **Figure 1**. Here, its function is explained. From left to right, the functional blocks include:

- > A coupling and protection network, including the shunt resistor $R1|R22$ or $R22$ alone.
- > A first amplification $\times 22$ stage ($U1$) of the shunt resistor voltage.
- > A 50-Hz blocking T-filter network ($R6, R7, R8, C3, C4, C5$).
- > A second amplification $\times 22$ stage ($U2$).
- > Two synchronous switching stages (unity amplification ± 1) with 90° phase difference ($U3, U4$ with $Q1, Q2$).
- > Averaging RC combinations ($R17, C6$ and $R18, C7$) connected to buffer amplifiers ($U7A, B$).
- > A four-channel (2 differences) 16 bit ADC-shield (based on ADS1115 by Texas Instruments) that reads the outputs of the buffer amplifiers and communicates these via I²C to the Arduino-programmed ATmega386 microcontroller.
- > A chain of two D-flip-flops $U5A, U5B$ that receive a frequency signal from the microcontroller (GPIO4) that is generated via the tone-command.
- > XOR logic and buffers ($U8A, U8B$) that create symmetric 90° shifted reference switching signal for the synchronous detector's outputs $U8A$ (Gf) and $U8B$ ($G1$).
- > Filtered and current limited ($C1, R25$) transistor switched frequency output (at $1/4$ of the tone frequency) to drive a miniature loudspeaker (8 to 80 Ω) that is resonance coupled to the sensor

foil and thus drives its vibration. The sensor foil is connected via the *Membrane* BNC connector (J1, bottom-left side of schematics).

- Results in terms of voltages and inferred phase angles are shown on and I²C coupled 16 × 2 LCD.

The analog ground reference is obtained from the 5 V supply by the rail splitter U10. As opamps, the zero-drift low-noise types OPAx182 are used throughout.

In **Figure 2**, a part of the prototype is shown, which includes the frontend amplifier, the 50-Hz filter stage, the sync detector and the signal amplification stages. It's realized on a full-copper boards (an overall shield), layered with strip solderable PC Boards and opamp breakout boards.

Figure 3 illustrates the whole prototype of this design, including the ADC module with U6, the amplifier module (U7), the ATmega328P controller, an Arduino board — used as a plain USB interface, without the controller — and the LCD module.

Synchronous Detection

To detect the tiny currents from the sensor membrane movements in the applied static electric field, we need a synchronous detection scheme that only “tunes in” on signal components that are synchronous with the driver excitation. Since, however, there is an unknown (and

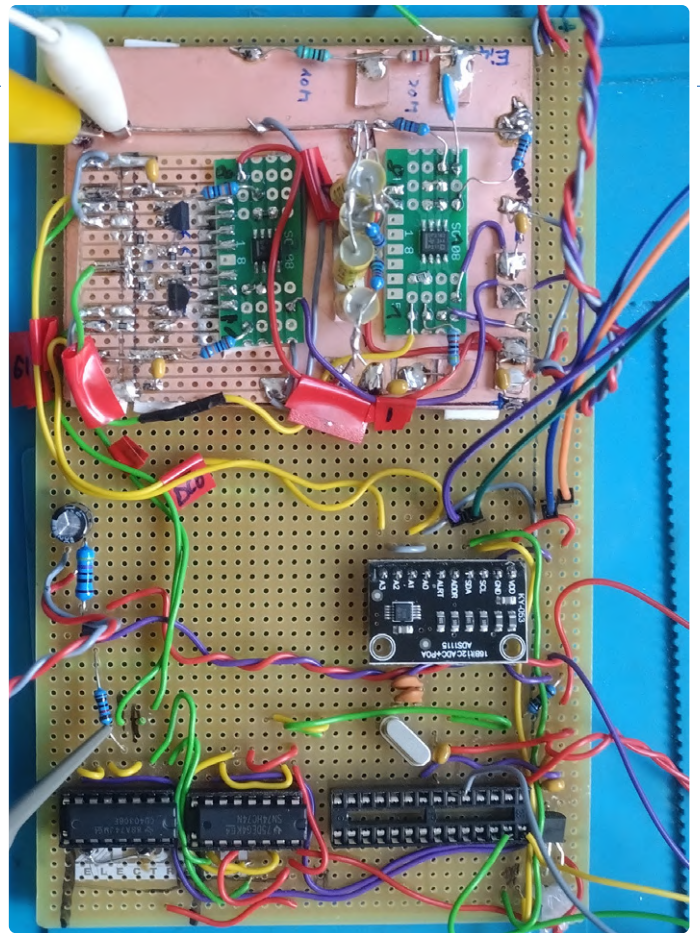


Figure 2: Amplifier frontend and sync-detection.

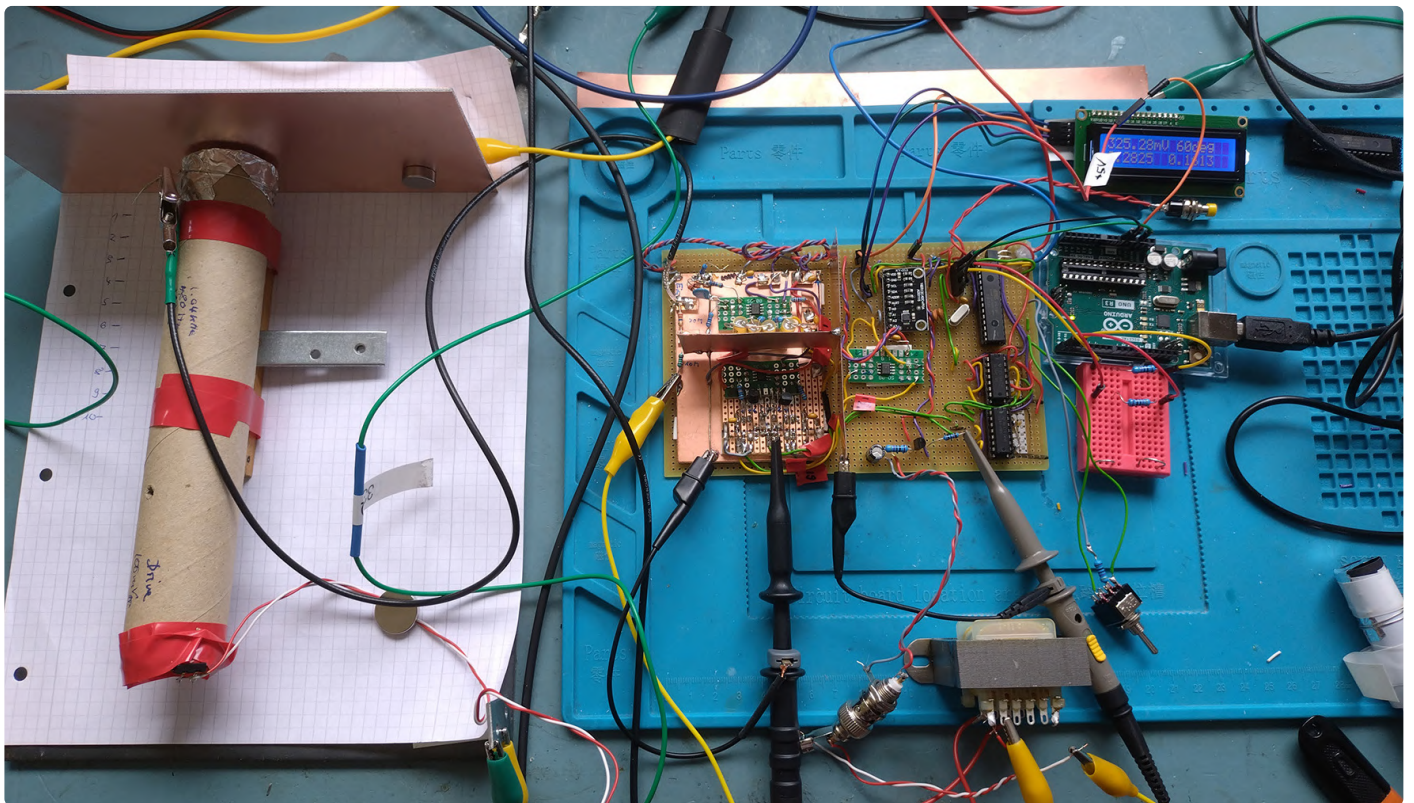


Figure 3: The complete design, with its main circuitry in the center; it includes: Amplifier (upper left), sync-detection (lower left), ADS1115 ADC, buffer amplifiers, drive transistor (middle from above), ATmega328 (Arduino), dual flip-flop, XOR-gate(s) (right from above). The Arduino UNO board — with ATmega chip removed — has been used as a programming interface and disconnected later on.

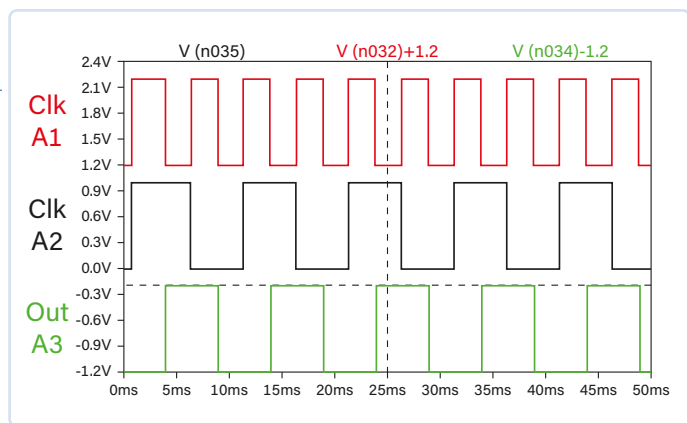


Figure 4: Signal sequence: "tone" frequency input (red), with U8A (black) and U8B (green) outputs showing the sync detection signals.

starkly frequency dependent) phase-shift between the loudspeaker excitation and the membrane vibration, the actual phase must be adjusted or determined during the actual measurement.

This is achieved by starting with an Arduino generated square wave four times the operation frequency f , which is divided down to $2f$ and then further to f by two cascaded flip-flops. The first division ensures a $2f$ signal that has a duty cycle of exactly 50%. Combining the $2f$ and f square waves (clk-A1 and clk-A2 in **Figure 4**) using an XOR gate (**Figure 5**) made with A1 and A2 (U5A and U5B in the schematic, respectively) creates a further $1f$ square wave (out A3) with a 90° phase-shift.

Feeding the clk A2 and clk A3 square waves to the gates G1 and G2 of the switching JFETs in the two synchronous detector stages around U3 and U4, yields averaged voltages U_{outG1} and U_{outG2} that are proportional to the in-phase and the 90° phase-shifted detector signal components. One of the square waves with frequency f (here clk-A2) is used to drive the speaker (i.e., the membrane); since the acoustic coupling yields sufficient filtering, only a simplistic one transistor

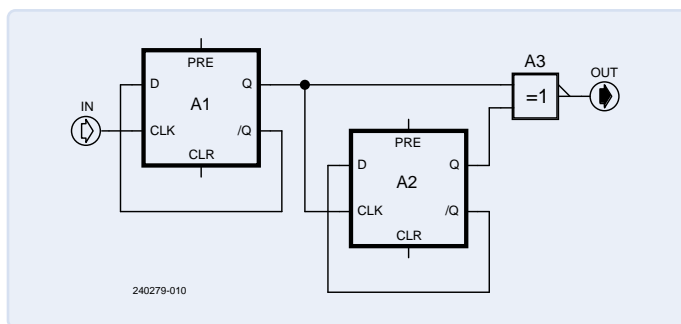


Figure 5: Flip-flop pair receiving the "tone" signal as input to A1. With the XOR gate A3 two 90 degree phase shifted sync-detector signals (U8A and U8B outputs) are generated.

(Q3) drive for the speaker with only C1 to partially smooth the square signal is foreseen.

The JFETs switch the amplifier stages U3 and U4 from $+1$ to -1 amplification, depending on their gate voltage. Finally, the averaged (via R17/C6 and R18/C7) voltages U_{outG1} and U_{outG2} — buffered by U7A and U7B — are digitized by ADC U6. From the digitized voltage values, the full signal amplitude and the phase are computed in the Arduino program. The extracted phase further contains the E-field polarity information.

With that, the signals at the outputs of the phase detector are illustrated in **Figure 6**. The traces shown are shifted for better visibility. To suppress the considerable noise in the input signal (CH1 = Monitor out) the averaging function was used for the oscilloscope acquisition, thereby cleaning the signal in a way similar to that of the sync-detection stages, to extract "noise-free" DC-voltages representing the input vibration signal.

The outputs of the sync-detector opamps are connected to $1\mu\text{F}$ averaging capacitors by $100\text{ k}\Omega$ resistors (low-pass filter). To achieve sub-mV accuracy, the capacitor voltages are buffered by another pair of OPAx182 opamps, used as voltage followers.



Figure 6: Phase detector outputs CH2 and CH3 obtained by multiplication of the yellow CH1 signal, CH4 shows the reference frequency. The oscilloscope's traces are obtained using averaging. The curves are vertically shifted, for a better visibility.

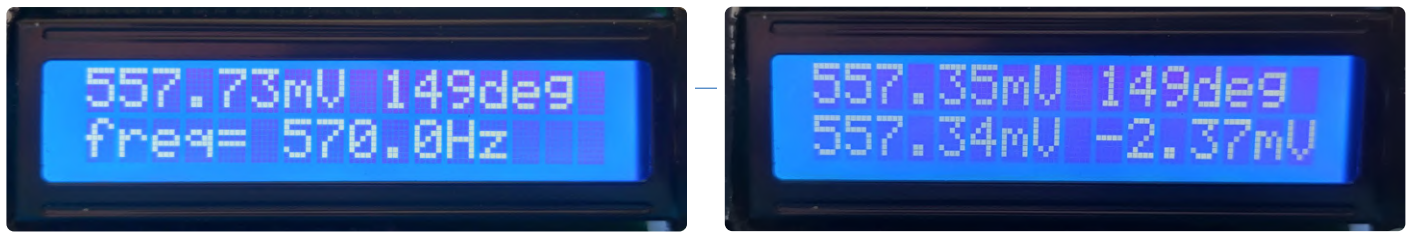


Figure 7: Display as shown during frequency scan (left) and in "normal" measuring mode (right).

The final averaged voltages relate to the amplitude U_R at the shunt resistor via:

$$U_a^1 = 2/\pi \alpha U_R \cos \phi$$

and

$$U_a^2 = 2/\pi \alpha U_R \sin \phi$$

with $\alpha \approx 400$ the amplification factor of the front end and ϕ the phase angle.

Operation and Measuring Sequence

At start/reset, first a frequency scan is initiated that determines the resonance frequency. This requires the presence of a (reasonably) constant applied field to the sensor membrane. As shown below, the resonance somehow depends on the distance d between membrane and object.

Thereby, the best frequency and the phase shift between drive and signal are determined and stored. The phase knowledge enables the determination of the sign of the applied field with respect to the reference at start.

Then measurement can be performed. The results are shown in terms of displayed voltage and phase angle — settling after an averaging period of a few seconds.

To interpret the results as potential of the investigated object, a compensating (enhancing) voltage can be applied to the membrane — as we'll see later on — via a 20-M Ω resistor (still only a negligible signal bypass!). The ratio of the measured values with and without compensating voltage and the original value multiplied with the compensating voltage yields an estimate of the object potential.

Program

The microcontroller is programmed using an Arduino environment and is downloadable at the Elektor Labs page for this project [1]. Upon start or reset, the program starts with a frequency scan (400 to 750 Hz) to find the resonance point of the sensor tube used. To do this properly, a constant field must be applied. During that phase, the display (top of **Figure 7**) shows the actual frequency together with signal amplitude and phase.

The switch SW3 (see circuit diagram) allows extending the scanning (up-ready-down) as long as one of the settings is kept. After the scan, the resonance frequency will be applied and the present phase determined and stored. Then the display (Figure 7, bottom) will show the actual total voltage

$$U_a = \sqrt{(U_a^1)^2 + (U_a^2)^2}$$

and the phase angle ϕ . The second line shows the voltages after "rotation" with the determined reference phase. I.e., if the phase has the original value, the first voltage is equal to U_a and the second voltage should be close to zero. This enables to check the polarity of the field regarding the reference field, as well as potential resonance shifts.

Mechanics

As an easy-to-make makeshift sensor arrangement, a cardboard or plastic (sewer tube) pipe can be used. The main setup is shown in **Figure 8**. For my experiments I chose lengths between 20 to 30 cm, the tube thickness was about 40 mm. At one side, a membrane consisting of standard household Al-foil was glued across the opening (no tension applied).

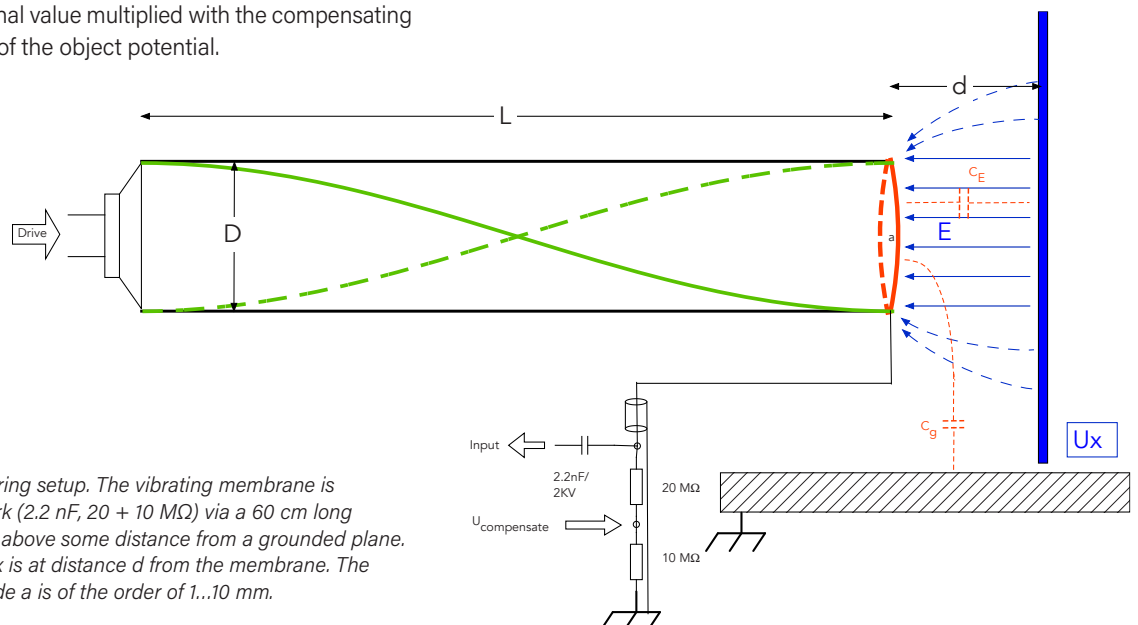


Figure 8: Sketch of the measuring setup. The vibrating membrane is connected to the input network (2.2 nF, 20 + 10 M Ω) via a 60 cm long RG174 coax cable. The tube is above some distance from a grounded plane. A plane at elevated voltage U_x is at distance d from the membrane. The maximum membrane amplitude a is of the order of 1...10 mm.



Figure 9: Setup for some test measurements (distance dependence, offset voltage effect). The "sensor" with the vibrating membrane at the "right" end faces the Cu plate, charged at a potential up to 1000 V, connected via a 33 M Ω resistor. The "sensor" sits on a cardboard box to yield some distance from the grounded steel plate. The Cu plate is isolated by a glass plate from the steel ground. On the bottom right, the display shows the total average voltage and the phase angle (1st line); the second line shows the voltages obtained by "rotating" the sync detector voltages with the calibration standard phase angle; thus the first value contains the proper E-field sign and the second value ideally should be zero.

At the other end, a miniature loudspeaker (8 to 80 Ω) was attached/glued. The Al-foil was connected to the amplifier input by a shielded RG174 cable with 40 cm length. A naive estimate of the basic $\lambda/2$ resonance of the tube pipe then would yield $330 \text{ m/s} / (0.4 \dots 0.6 \text{ m}) = 550 \dots 850 \text{ Hz}$. Acoustic impedance effects of loudspeaker and membrane and coupling to their resonance (may) cause further deviations.

For flexible experiments and tests, I used a steel ground plate which allowed to fix a potential plate (Cu printed circuit board material) as well as the tube using a number of assembly brackets and small NdB magnets. **Figure 9** shows a photo of this assembly. The U_x potential parts were isolated from the steel by a glass plate. The steel plate was connected to ground. The coupling capacitances C_E and C_g determine the ratio between the displayed output voltage and U_x .

Acoustic Resonance

The microcontroller program searches for the (optimal) resonance frequency of the actual setup. The general behavior of the response structure of the "tube" sensors (with applied voltage at the adjacent plane) is displayed in **Figure 10**. At shorter distances between membrane and adjacent potential plane (d), the exact position of the main resonance depends on this distance. Here it lies between 500 to 600 Hz.

Results

To test the distance (d) dependence of the resulting voltage U_a experiments were carried out. Here I report the result obtained with the "sewer pipe sensor." Initial resonance determination was initiated with

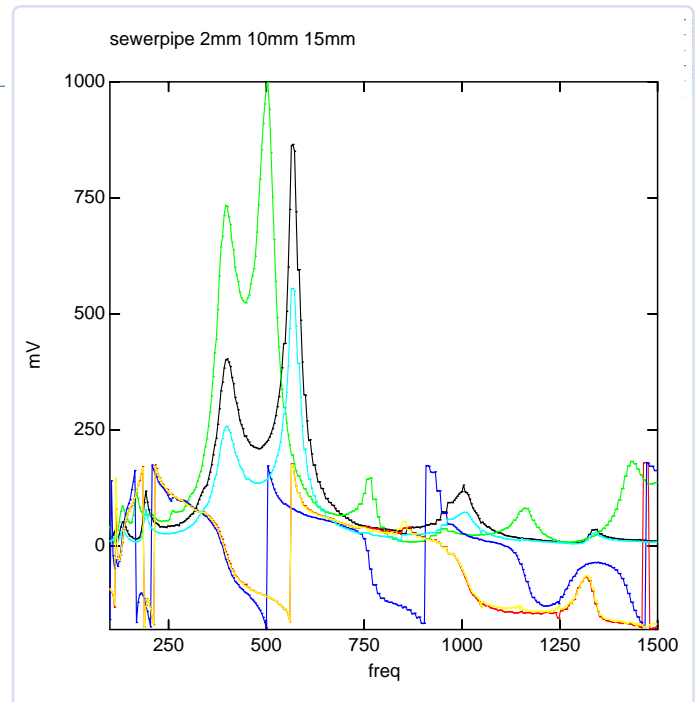


Figure 10: Resonances of the "sewer pipe" assembly: the black curve represents the signal amplitude with the membrane at a distance of 2 mm, the green one at a distance of 10 mm. The red and blue curves represent the corresponding phases ($1^\circ = 1 \text{ mV}$). Note the artificial back-folding jumps when ± 180 degrees are reached.

16 mm charged plate distance and 1000 V plate potential. Then the distance was varied and the resulting voltage U_a without and with applied (here 285 V) offset voltage U_b were registered.

The results are shown in **Figure 11**. In **Figure 12** the ratio of resulting voltages without and with compensating offset voltage (applied to the membrane) are shown. The limiting value for $d \rightarrow 0$ allows to infer the U_x value of the object under investigation.

Thus, the setup may be used to estimate the potential of an electrostatically charged part. I tried a Styrofoam plate charged by rubbing to some cloth. The ratio of $U_a \approx 200 \text{ mV}$ and change by 285 V offset $\Delta U_a \approx 6 \text{ mV}$ yields an estimate of $285 \text{ V} \times 200/6 = 9.5 \text{ kV}$. One can also monitor the slow decay of electrostatic charges. This may be a useful application of the setup to assess ESD risks.

Order of Magnitude of Effect and Membrane Amplitudes

With the values: shunt resistor $R_s = 1 \text{ M}\Omega$, first stage amplifier x420 ($647.03 \text{ mV} / 701.82 \text{ mV} = 0.922$ damping of the 50 Hz T-filter at the used frequency of 520 Hz); synchronous detection x1 and $2/\pi = 0.6366$ we obtain a sensitivity such that 1 nA at input yields 0.2465 V at the DC output of the synchronous detector.

The capacitor membrane velocity $v = dd/dt$ (assuming rigid plate movement) depends on the displacement amplitude a :

$$v = 2\pi f a \cos(2\pi f t)$$

i.e., the velocity amplitude $v_0 = 2\pi f a$; with an assumed estimated value of 5 pF for C , a field $E = 30 \text{ V/mm}$ and measured $U = 0.3 \text{ V}$ we would get $I_0 = E C v_0 = E C 2\pi f a \approx 1 \text{ nA}$ yielding $a = I_0 / (E C 2\pi f) \approx 2 \text{ }\mu\text{m}$.

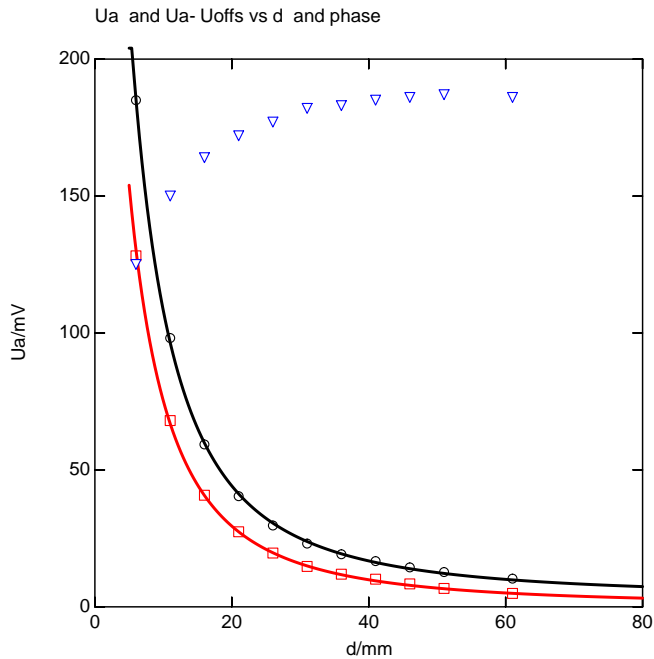


Figure 11: Measured voltage values as function of distance d at a plate potential of 1000 V and a detector shunt resistor value of 100 k Ω . The red curve is the value obtained with applied compensation voltage of 285 V. The change of phase (resonance) with Cu-plate distance is illustrated by the blue symbols. The lines are fits with expressions that somewhat deviate from the theoretical $1/d^2$ behavior due to influences of the parasitic capacitance C_g .

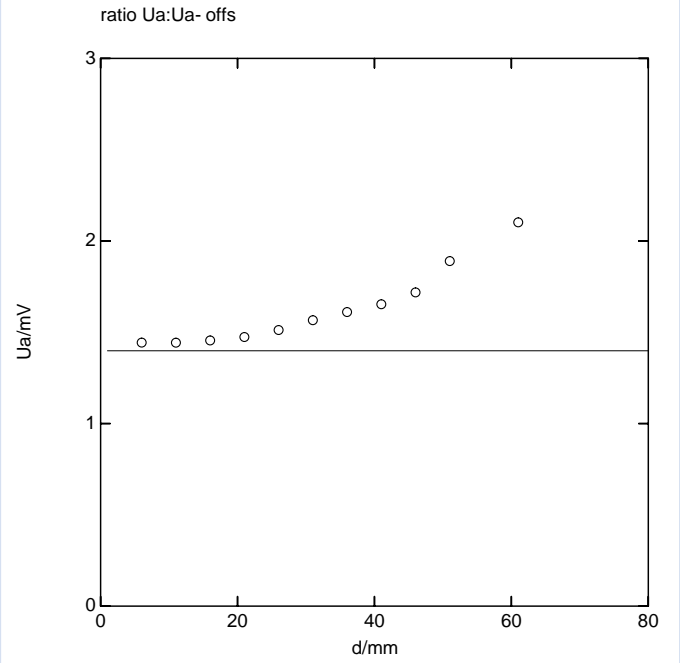



Figure 12: Ratio of the measured U_a values without and with compensating (externally) applied offset voltage. Ideally — without parasitic capacitive couplings — the value should enable the assessment of the U_x value of the adjacent plate. The expected ratio would be $U_x/U_x - U_{comp}$ here: 1000 V / (1000 V - 285 V) as indicated by the horizontal line. The influence of stray capacitance in a larger distance is clearly visible in deviation of the measured points from this value. Nevertheless, the offset method allows the approximate assessment of U_x .

Since the real displacement of the membrane is not that of a rigid plate but rather that of a membrane that is fixed at the boundaries, the effective displacement amounts only to 10 to 15% of the central amplitude. Thus, we infer a maximum central membrane amplitude up to 13 to 20 μm at resonance.

What's Next

In the next installment of this article, I will describe the interferometric measurement of this amplitude — using a simple makeshift interferometric setup. Stay Tuned! 

240279-01



About the Author

Michael Monkenbusch is a retired physicist who worked in the fields of neutron scattering, instrumentation, and soft-matter physics. Reviving an old electronics hobby led to the project presented here.

Questions or Comments?

Do you have technical questions or comments about this article? Please contact the author at michael.monkenbusch@googlemail.com or the Elektor editorial team at editor@elektor.com.



Related Products

- Douglas Self, *Small Signal Audio Design* (4th Edition), Focal Press
www.elektor.com/18046
- FNIRSI 1013D 2-ch Tablet Oscilloscope (100 MHz)
www.elektor.com/20644



WEB LINK

[1] Elektor Labs page for this project: <https://tinyurl.com/2t4wmw3w>



Wireless Mailbox Notifier

From Optical Sensors to Radar, Exploring a Few Options

By Jean-François Simon (Elektor)

Are you eagerly waiting for a letter or a small parcel? Let's see what sensors and RF modules we can use to build a simple mailbox notifier, and why infrared worked best in the end.

Unnecessary trips to check an empty mailbox can be frustrating, especially in bad weather or for people in rural areas where the mailbox might be far from the house. This is particularly true for individuals with mobility challenges. Additionally, in areas where mail theft is a concern, a notifier can enhance security by alerting the user as soon as the mail arrives, reducing the risk of tampering.



Figure 1: A typical French mailbox.

Of course, this problem can easily be solved by purchasing a ready-made system. Many options are available, from mechanical solutions to electronic ones. However, building it yourself is an excellent opportunity to explore different types of sensors and wireless systems while enjoying the satisfaction of creating something unique!

Design Objectives

The most obvious design goal for the project is of course that it should be compatible with the particular type of mailbox I have, which is shown in **Figure 1**. The system must operate entirely wirelessly, both for power and communication, for ease of installation. Additionally, it should require minimal modification to the mailbox itself. Those modifications will have to be discreet, as I don't want the letter carrier to be confronted with an eccentric device which could worry them or give them the impression of being spied on.

Simplicity is another key objective; I will be favoring pre-existing modules for sensors and communication. Moreover, the system must have sufficient range to transmit information reliably over approximately 20 meters, traversing multiple concrete walls to reach the indoor receiver, as well as a battery life of at least six months. Lastly, in the absence of an existing home automation system, the notifier must function as a standalone solution without reliance on servers or internet connectivity.

A Look At Several Types of Sensors

Since there's really no such thing as a dedicated mail sensor, we have to capture its arrival in a roundabout way. Sensing movement is a relevant approach. Magnetic sensors containing reed switches, like those used in alarm systems for doors and windows, are simple, cheap and consume minimal power (the sensor itself doesn't use



any power, only the microcontroller monitoring it does). However, if your mailbox has both a door and a flap for smaller items, two sensors would be needed, as well as their two corresponding magnets, which slightly complicates wiring and mechanical modifications to attach the magnets and sensors.

Weight sensors at the bottom of the mailbox could work, but they're a bit on the complex side to implement. Assuming the mailbox itself can't be moved, it would be needed to install a double floor at its bottom, with a load cell under it. A kitchen scale could be adapted for this, but there are probably simpler options. Light sensors are promising, as they can sense changes in illumination when the mailbox is opened. However, they may be hard to calibrate so that they work in any ambient light, even at night. Camera modules have also been used in similar projects. For example, several people have shared online their implementations using ESP32-CAM modules. To fully leverage such a camera module, it is ideal to have the ability to view the images remotely, which requires a substantial bandwidth between the transmitter and receiver. Typically, this is achieved using Wi-Fi, but I'd rather not take that route for this project.

Passive Infrared (PIR) Sensors

PIR sensors are widely used in applications such as alarm systems and home automation, making them readily available and extremely affordable. A quick search for "PIR Sensor" on platforms like AliExpress reveals an extensive selection in various shapes and sizes, often at remarkably low prices. They are easily recognizable by their distinctive plastic Fresnel lens (**Figure 2**). PIR sensors detect infrared radiation emitted by warm bodies, such as humans and animals. At first glance, it seems unusual to use this sensor for detecting a letter or parcel, as these are typically at a temperature close to the ambient environment. Yet, several commercial mailbox notifiers use this type of sensor, so they are obviously worth considering. I'll investigate myself later.

Optical Proximity Sensors

Proximity sensors often consist of an infrared emitter and receiver housed in the same unit. By emitting pulses of infrared light and analyzing the reflected signals, they can determine if an object has entered their detection range. Periodic measurements allow the system to detect movement by comparing changes in reflected light. For instance, the VCNL3040 [1] from Vishay offers a detection range of 0 to 300 mm and is promising for this application. However, it's housed in a small SMD package, with eight solder pads constrained in a tiny 4 mm × 2 mm rectangle, which makes prototyping a challenge. Other optical proximity sensors, such as the VL53L series [2] from STMicroelectronics, use a different technique based



Figure 2:
PIR motion detector.

on Time-of-Flight (ToF) measurements. Depending on the model, these sensors can measure distances from a few millimeters to several meters. This is of course not needed for a mailbox, but very impressive nevertheless.

Radar Modules

Radar modules are another fascinating option. These sensors emit extremely high-frequency signals, at 10 GHz and above, and analyze the returned signal to determine the position and velocity of objects within their detection range. Their primary advantage lies in their immunity to environmental factors such as temperature, humidity, light, airflow, dust, etc. which make them relevant in harsh industrial environments. Low-cost modules like the CDM324 and HB100 [3] are widely available from Chinese suppliers. There are also more expensive and better documented options available at Western distributors, such as the IPM165 from InnoSenT or the A111 and XM125 [4] from Acconeer. At the other end of the cost spectrum, a minimalist option is the RCWL-0516 [5], which offers basic motion detection on a very simple PCB. All these modules are fascinating and would deserve to be studied in detail, but fully leveraging their capabilities is beyond the scope of this article. I did a few tests with the RCWL-0516 but had difficulties in reliably detecting motion inside the mailbox, likely due to signal reflections off the internal walls.

RF Systems

Now that we have an idea of the possible sensors, let's look at the possibilities in terms of wireless transmission. 315 MHz and 433 MHz RF modules are cheap and commonly available. The simplest and cheapest modules use On-Off Keying (OOK) and there is no built-in mechanism to avoid transmission errors, thus it's required to take care of it in software, which can be quite a bit of work to get a reliable transmission. More advanced modules take care of this for you and can be used as a wireless serial port, such as the well-known HC-11 and HC-12 [6] modules, or the slightly more expensive E32 modules from Ebyte. However, depending on the chosen sensor, a full-blown serial port may not be needed.

Figure 3: The insides of the PR100 sensor.



Wi-Fi, while offering higher data throughput which can be useful for using together with a camera module, is power-hungry which makes it tricky for long-term battery use. Bluetooth Low Energy (BLE) is power-efficient but limited in range, making it less practical for mailboxes located far from the house. Protocols like Zigbee and LoRa are good for low-power, long-range communication. They provide better range at the cost of slower data rates, but could be well-suited for basic “you got mail” notifications.

Some Decisions

All this research provided fascinating possibilities. I wanted to try everything! Optical time-of-flight sensors, with their fancy surface-emitting infrared lasers, got me curious. I also found radar sensors extremely interesting and found a great video about the CDM324 on The Signal Path channel [7]. However, time is running out, and what was meant to be a “quick and dirty project” shouldn’t turn into a six-month research endeavor. As a result, I had to make some practical compromises. My strategy has been to focus on off-the-shelf devices containing both a sensor and a communication device in one package. This approach allows for prototyping with functional, validated blocks already tested by their manufacturers, avoiding the need to reinvent the wheel.

While researching wireless motion detectors online, I found that the vast majority rely on PIR sensors. Initially, I had doubts about their ability to detect the motion of a letter sliding into a mailbox, as these sensors are

primarily designed to detect humans or animals based on the temperature difference (and infrared radiation) between them and the surrounding environment. For parcels, however, this issue is less critical; the mail carrier must open the mailbox to deposit them, and this motion (or the mail carrier) is easily detected by a PIR sensor. Furthermore, I noticed that several commercial mailbox notifiers, such as those from the French brand Novlee and the American brand Ring, also use PIR sensors. This gave me confidence to test them myself.

A quick search for “Wireless PIR” on platforms like AliExpress revealed a wide range of options. These fully integrated modules, typically battery-powered, include a PIR sensor and an RF transmitter that alerts the receiver when motion is detected within the sensor’s field of view.

Among the many available modules with different radio options available — Zigbee, Wi-Fi, 315 MHz, 433 MHz, 868 MHz, and others — I chose the PR100 module by Staniot (**Figure 3**). It uses the EV1527 protocol, a widely adopted standard often found in garage door remotes and similar devices. The sensor sends EV1527 pulse trains whenever motion is detected. This choice allows me to select from a wide range of compatible receivers, offering good range, easy setup, and no need for a Wi-Fi network or a LoRa gateway. This module is powered by two AAA batteries and its quiescent current is below 10 μ A, with an active current of around 15 mA during detection and transmission. This should result in excellent battery life.

EV1527?

There are many devices labeled with “EV1527” or “1527” but the documentation leaves much to be desired. In fact, EV1527 is not really a protocol but rather an IC [8] made by Silvan Chip, specifically for garage door remote controls. It has four inputs for up to four buttons, and one output used to toggle the RF transmitter on and off. The implementation of the RF part, typically at 315 MHz or 433 MHz, is left to the remote manufacturer’s discretion. It uses a fixed-code transmission mechanism, where a 20-bit address and 4-bit data are sent as a serialized pulse train. To use the remote, you first have to make the receiver learn the particular 20-bit address of the transmitter. This system was so successful that it was widely copied and is now massively used in wireless doorbells, remote-controlled switches, and many more devices.

Receiver

The choice of a 433-MHz module with a built-in sensor and transmitter naturally determines the options for the receiver. A quick search for “EV1527 receiver” on popular electronics marketplaces reveals a wide range of compatible modules. These vary in features such as the number of output channels, relay or MOSFET-based outputs, etc.

Since only one output channel is needed, I selected the QA-R-011 receiver [9] from Qiachip (**Figure 4**). I bought it in a pack that also contained a small EV1527-compatible remote control in key-ring format, which will be handy for testing.

This module features an open-drain output driven by an NTD20N06 MOSFET (20 A max.), which is a bit excessive for handling logic level signals but otherwise perfectly functional. I connected a simple LED indicator to the receiver's output. The receiver can be configured in several modes, one of which is "latching." In this mode, the receiver must learn a code to turn on the output, and a different code to turn it off. The PR100, on the other hand, always sends the same code when motion is detected, which will be the "on" code from the receiver's point of view. This way, a single movement detection will turn on the indicator light and keep it on. One of the buttons on the key ring remote control can be used to transmit the "off" code to switch off the indicator light once you've picked up your mail, or you may power cycle the receiver for the same purpose.

Practical Build

Some YouTube tutorials suggest that installing a module like the PR100 inside of a mailbox, placing the receiver in the house, and connecting it to a light is all it takes. This approach can work – but only if your mailbox is made of wood or plastic, at least on one side. For a fully metal mailbox, however, the signal is almost completely blocked by the sheet metal, preventing it from reaching the receiver indoors.

To address this, I needed to place the PIR sensor inside the mailbox while positioning the radio transmitter outside. This required drilling a few holes in the mailbox.

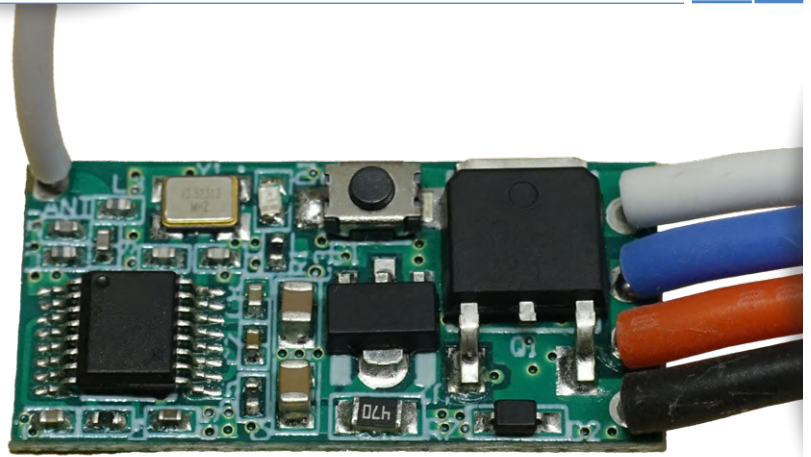


Figure 4: Close-up of the QA-R-011 receiver.

To maintain good relations with the landlord, I replaced the house's mailbox with another old and rusty unit that I had no hesitation modifying. The PR100 module was disassembled, with its NS312 PIR sensor relocated inside the mailbox, by soldering it to a piece of perfboard which is then attached with double sided sticky foam. The NS312 sensor is then covered with a small Fresnel dome salvaged from another PIR sensor kit I had in stock. The rest of the PR100's PCB, including its antenna, was placed in a plastic junction box mounted outside the mailbox (**Figure 5**).

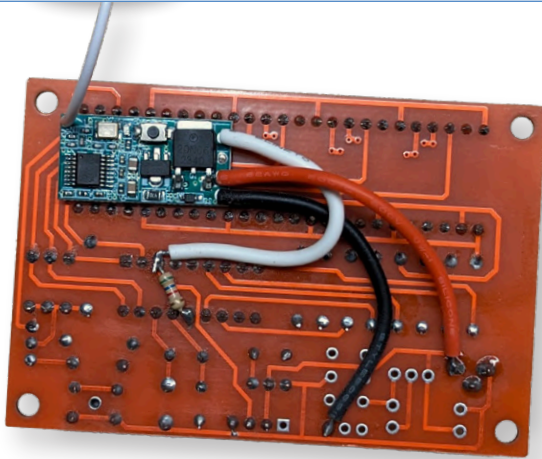
Sensor Placement

It quickly became apparent that the PR100 frequently triggered false positives, lighting the LED even when no mail was delivered. Indeed, placing the sensor at the back of the mailbox worked well for detecting mail but also picked up pedestrians on the street through the mailbox opening. The same issue occurred when the sensor was mounted on upper internal side of the mailbox. To diagnose the issues and help me find the best position for the sensor, I switched the receiver mode from latching to momentary action, so each detection



Figure 5: The PCB of the PR100 has to be mounted outside the box to allow signal transmission.

Figure 6: EV1527 receiver and pulse counter connected together.



by the PR100 produced a pulse output from the receiver. These pulses can then be counted by an Arduino or any other similar device.

Staying true to the “less is more” philosophy, I utilized a module I already had in stock, which you can easily find on eBay or AliExpress. It’s a small red PCB equipped with a 5-digit, 7-segment display and a Microchip PIC16F628A microcontroller, typically sold as a “Frequency Counter - Crystal Tester.” Heinz Winter, from the YouTube channel TheHWcave, has made several excellent videos [10] about modifying this module, including a mode for counting single pulses, even down to DC. You can find the explanation in the video and the source code on his GitHub repository [11]. I simply connected the open-drain output of the EV1527 receiver to the counter’s pulse input and added a pull-up resistor (non-critical value, I used 2.6 kΩ). The receiver is just small enough to fit between pins of the 7-segment display and was secured with a dab of hot glue (Figure 6).

Every day, the counter reading provides a clear indication of what happened. 0: No mail. 1: A letter or package



Figure 7: The best position for the sensor, at least for me.

is present. 2: Both a letter and a package (or possibly junk mail). 3 or more: The sensor likely detected external movements near the mailbox! To reset the counter every day, its power supply (an old phone charger) is plugged in a timer socket.

Ultimately, the best position for the PIR sensor was on the inner side of the front door, pointing toward the back of the mailbox (Figure 7). This configuration reliably detected envelopes slid through the slot without opening the door and had no trouble detecting motion when the door was opened to deposit a package, all without detecting nearby pedestrians. Initially, I wanted to build a better, prettier receiver unit, but the cheap 7-segment LED counter blends right in with my living room now (Figure 8). In the end, I kept the rusty old mailbox. Nothing is more permanent than a temporary solution! However, these fancy radar sensors are intriguing though, I’ll have to keep experimenting and find a use for them in another project! ◀

240715-01



About the Author

Jean-Francois Simon (Engineer, Elektor) has a longstanding passion for electronics and enjoys topics as varied as circuit design, test and measurement, prototyping, playing with SDRs, and more. He likes to create, modify and improve his tools and other systems. He has an engineering background and also enjoys mechanics, machining, and all things technical. Follow him on X at x.com/JFS_Elektor.

Questions or Comments?

Do you have questions or comments about this article? Email the author at jean-francois.simon@elektor.com, or contact Elektor at editor@elektor.com.



Figure 8: Yay, something has arrived!



Related Products

- SparkFun Sensor Kit
www.elektor.com/19620
- HC-SR501 PIR Motion Sensor Module
www.elektor.com/18420



FEATURED TOPIC

Visit our **IoT & Sensors** page
for articles, projects, news, and
videos.



[www.elektormagazine.com/
iot-sensors](http://www.elektormagazine.com/iot-sensors)

WEB LINKS

- [1] VCNL3040 sensor: <https://www.vishay.com/docs/84917/vcnl3040.pdf>
- [2] VL53L sensor series: <https://www.st.com/en/imaging-and-photonics-solutions/vl53l0x.html>
- [3] Stefano Lovati, "The HB100 Doppler Motion Sensor," Elektor 7-8/2023: <https://www.elektormagazine.com/230205-01>
- [4] XM125 Sensor at Sparkfun: <https://www.sparkfun.com/products/24540>
- [5] Information on the RCWL-0516: <https://github.com/jdesbonnet/RCWL-0516>
- [6] HC-12 RF module datasheet: <https://www.elecrow.com/download/HC-12.pdf>
- [7] Video about the CDM324: <https://www.youtube.com/watch?v=5vqSX40seqA>
- [8] EV1527 datasheet: <https://www.sunrom.com/download/EV1527.pdf>
- [9] QA-R-011 receiver: <https://qiachip.com/blogs/usermenu/light-control-module-specification-qa-r-010>
- [10] Video from TheHWCave about the pulse counter: https://www.youtube.com/watch?v=3imEpm_zztQ
- [11] Source code for the counter: <https://github.com/TheHWCave/PIC-freq.counter-modification/>

FRONT PANELS. ENCLOSURES. MILLED PARTS.

Design it yourself & order directly

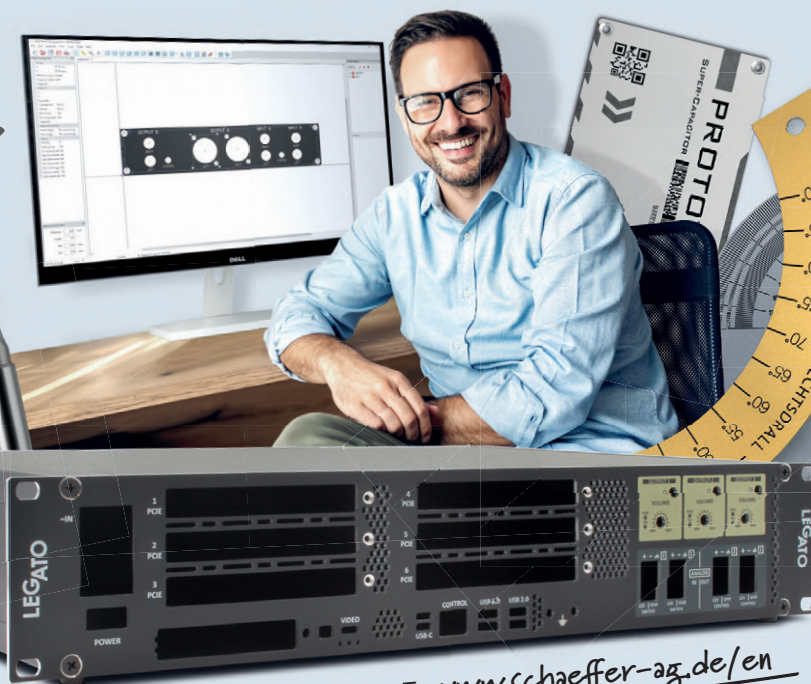
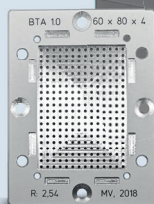
Free design
software

For Windows,
Linux and Apple



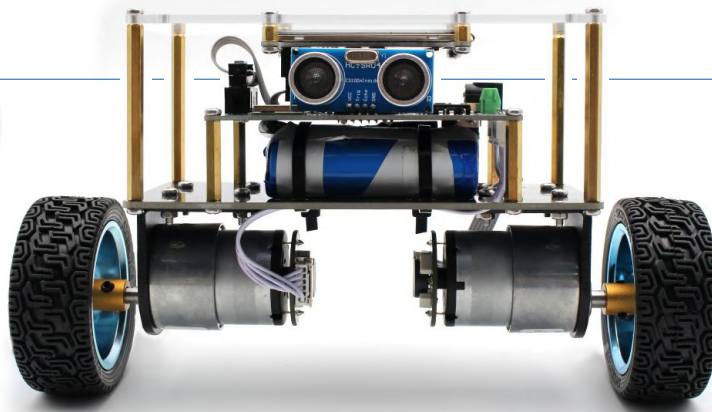
CNC
Excellence

Starting at
1 piece



MORE AT www.schaeffer-ag.de/en





Elektor Mini-Wheelie

A Self-Balancing Robot

By Clemens Valens (Elektor Labs)

The Elektor Mini-Wheelie is an experimental self-balancing robot powered by an ESP32-S3 microcontroller and programmable with Arduino. Designed for hands-on learning, it features wireless connectivity, sensor integration, and advanced motion control, making it a great platform for exploring robotics and embedded systems.

Specifications

- › ESP32-S3 microcontroller with Wi-Fi and Bluetooth
- › MPU6050 inertial measurement unit (IMU)
- › Two independently controlled 12 V electric motors with two hall-effect sensors
- › Ultrasonic transducer HC-SR04 (4-pin)
- › 320 × 240 pixels TFT colour display
- › MicroSD card slot
- › Battery power monitor
- › 3S rechargeable Li-Po battery (11.1 V, 2200 mAh)
- › Native USB-C port
- › Serial-to-USB-C converter
- › Arduino-based open-source software

The Elektor Mini-Wheelie is an experimental self-balancing robot platform designed for autonomous operation. Powered by an ESP32-S3 microcontroller, this robot is fully programmable using the Arduino environment and open-source libraries. Its built-in wireless capabilities enable remote control via Wi-Fi, Bluetooth, or ESP-NOW, as well as communication with users or even other robots. An ultrasonic sensor is included for obstacle detection, while its colour display can showcase expressive faces — or, for the more practically minded, detailed debug messages.

The Elektor Mini-Wheelie comes as a kit of parts that you must assemble yourself. Everything is included, even a screwdriver. After assembling the robot, it must be programmed. It comes preloaded with a simple program that shows if the motors and display are working, but that's all it does. To make the robot balance, it must be programmed with the Arduino sketch *SBRobotDabble* available at [1]. There, you will also find detailed instructions on how to do this.

The *SBRobotDabble* program needs a smartphone app called Dabble to work. With this app, the robot can be controlled over Bluetooth using

a virtual gamepad. The Dabble app is free and available for Android and Apple smartphones and tablets. Again, detailed instructions can be found at [1].

With the Dabble app, the Mini-Wheelie can move forward or backward, and it can turn left and right. However, it is important to understand that this is not as easy as it sounds, and it requires some practice to make it move the way you want it to move.

How Does Self-Balancing Work?

Basically, a self-balancing robot works like an inverted pendulum (e.g., a rocket). To remain upright in an unstable equilibrium, it must constantly measure its orientation in the vertical plane and correct its attitude accordingly. The Mini-Wheelie is simpler than an inverted pendulum as it can only fall over forward or backward, not sideways.

When nothing is done, the robot will fall over. However, it has two motorised wheels that allow to adjust the robot's centre of gravity. If the robot starts tilting forward, it moves the wheels forward to bring its centre of gravity back over the wheels. If the robot tilts backward,

it moves the wheels backward. By continuously making these small corrections, the robot remains upright.

To detect tilt, the robot is equipped with an Inertial Measurement Unit (IMU) that includes an accelerometer and a gyroscope. The accelerometer measures linear acceleration and helps detect the tilt angle relative to gravity. The gyroscope measures angular velocity (how fast the robot is tilting) and helps detect rapid changes in orientation.

The IMU on the Mini-Wheelie is an MPU6050 from InvenSense (TDK), a popular IMU in the maker community for which extensive open-source software libraries are available and which is easily found mounted on a small module. The MPU6050 integrates a so-called Digital Motion Processor (DMP) that combines the gyro and accelerometer data into easy-to-use precise tilt data. Called sensor fusion, this greatly simplifies the application program as it doesn't have to fuse the data itself.

The tilt angle is sent to a PID (Proportional Integral Derivative, see inset) controller algorithm that continuously determines how much the speed of the wheels must be adjusted for the robot to stay upright by calculating the difference (the error) between the desired and actual tilt angle.

Moving Forward

A robot keeping balance and remaining upright is nice, but not very useful nor interesting. Allowing it to move around is what we want, which implies some way to control it remotely. For this reason, the Mini-Wheelie comes equipped with wireless capabilities in the shape of Wi-Fi and Bluetooth. This permits easy communication with a portable device like a smartphone or tablet. An additional ultrasonic sensor can be mounted to help with obstacle detection and navigation.

How do you make a self-balancing robot move forward or backward? Easy, just manipulate its tilt angle, much like how you control movement on a hoverboard or a Segway-style vehicle with your body (**Figure 1**). By intentionally introducing an offset in the calculated tilt angle, the robot will attempt to correct this imbalance, causing it to move in the direction of the tilt.

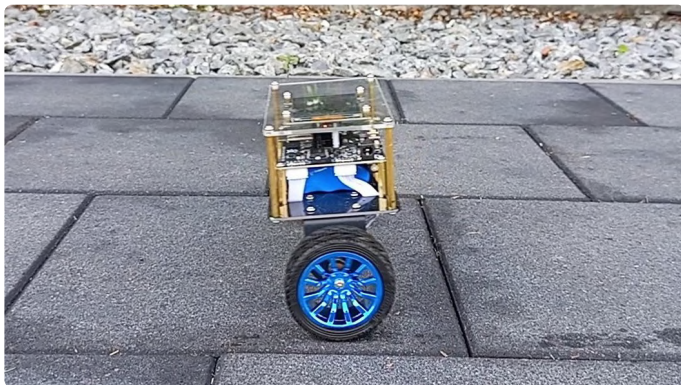


Figure 1: The robot will move into its tilting direction.

About the Software

The Mini-Wheelie is compatible with an open-source self-balancing robot project that has been online for several years now and in multiple incarnations. It can be found in various places, but where exactly it originated is unclear. For this project, we tracked down the original libraries as much as possible and then adapted them to the Mini-Wheelie platform where needed. Therefore, it is highly recommended to use the libraries available from [1] as some have been modified by us.

We used the *Dabble* library for remote control over Bluetooth, mainly because we had used it before. It works fine but has one drawback: it is not compatible with the ESP32 Boards Package version 3 or higher. If, for some reason, you are obliged to use version 3 or higher, go look for another Bluetooth control library. Make sure it is non-blocking, as otherwise the Mini-Wheelie will not balance properly. Again, see [1] for more details.

In this state, the robot behaves similarly to a satellite in a geostationary orbit — constantly “falling.” However, unlike a satellite that won't crash into Earth because the orbit is wider than the planet's diameter, the robot operates on a large flat surface. To prevent tipping over, it relies on its motorized wheels and a sophisticated control algorithm to maintain balance while moving. This is a delicate balance. If the control algorithm can't keep up, the robot will fall over; if it anticipates too much, the robot will stop moving or go the other way.

This is why the Mini-Wheelie is easier to control on a rough surface than on a smooth surface. A rough surface presents small obstacles that keep the control algorithm active, but on a smooth surface it tends to run out of control. It needs noise and disturbances to work well (**Figure 2**).

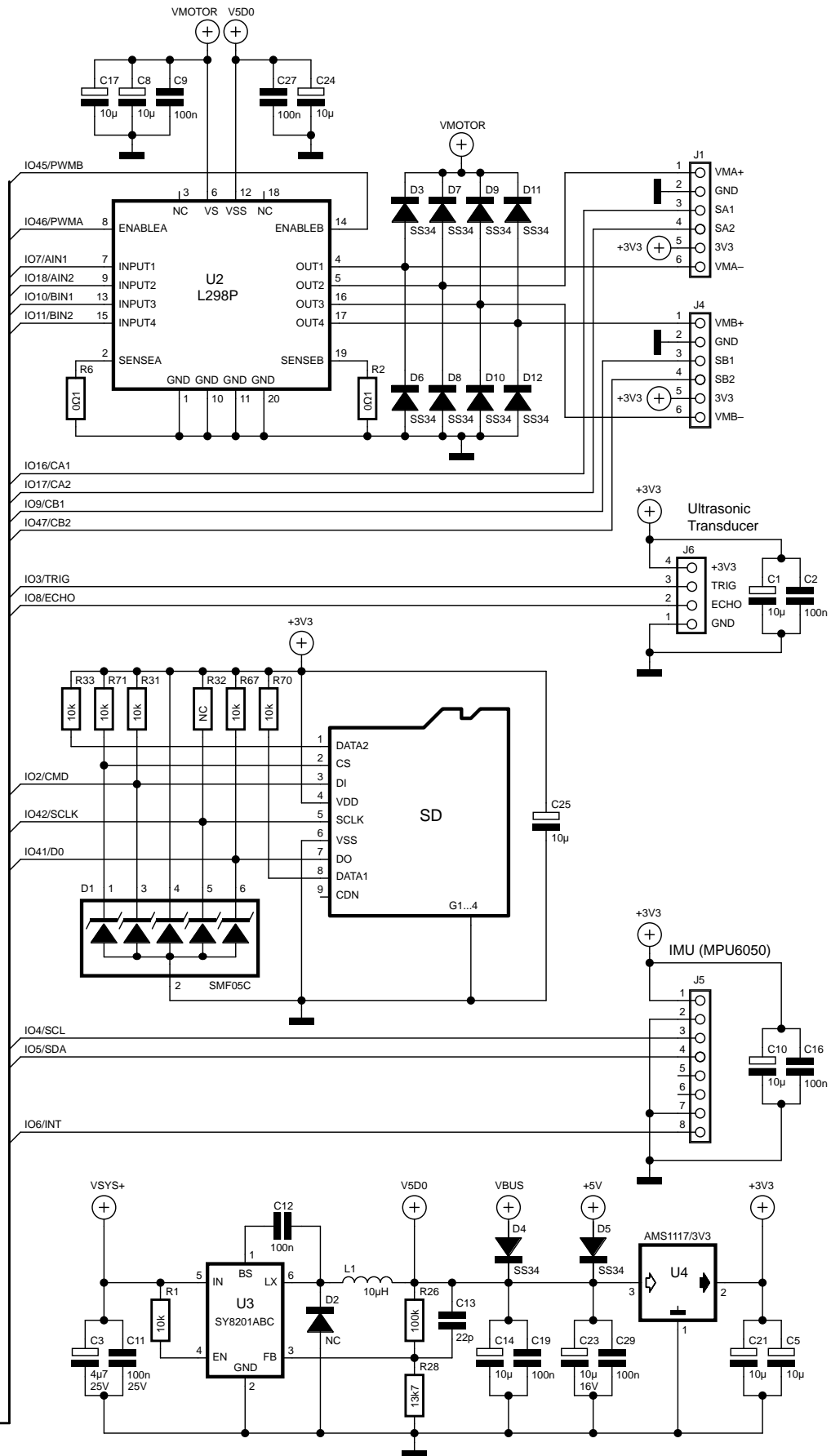
Turning is like moving forward or backward, but with one wheel spinning faster than the other. This implies introducing an asymmetric tilt offset, which doesn't make things easier. Therefore, the Mini-Wheelie with its default software does not allow turning and moving at the same time. However, turning while moving is possible simply by moving first and then turning while it is still in motion.



Figure 2: The Mini-Wheelie with its big tyres likes rough surfaces.

The Mini-Wheelie is designed as an educational development platform intended for learning, experimentation, and robotics development. It is not classified as a toy for children, and its features, documentation, and intended audience reflect this purpose. It is aimed at students, educators, and developers who wish to explore robotics, programming, and hardware integration in an educational setting.

44 July & August 2025 www.elektormagazine.com



Note that the Mini-Wheelie software assumes that the upright (vertical) position is at 180° ([originalSetpoint](#), line 160). Chances are that this is slightly optimistic in reality due to mechanical tolerances. Therefore, it is worth experimenting with this value to adapt it to your robot.

Furthermore, there may also be some play in the motor shafts, which can introduce a slight jolt when the direction of rotation is reversed.

Also, always bear in mind that humans tend to think they know better and are cleverer than the system. Trying to be too clever or interfering too much with control algorithms often leads to sub-optimal results.

Features

Besides a powerful IMU, the Mini-Wheelie has a few other features that can be used to improve balancing and remote control. The complete schematic of the main board is shown in **Figure 3**. The main board itself is shown in **Figure 4**.

Hall-Effect Sensors

The two wheels are both equipped with two Hall-effect sensors that allow detecting the spinning direction and speed of each wheel individually. In the software, these inputs are labelled **CA1 / CA2**, and **CB1 / CB2**. The sketch *SBRobot_Motor_Compare* uses them to calibrate the two motors. Refer to this sketch to see how to integrate these sensors into your own program. They can provide information about the direction the robot is moving in and if it is going straight or not. You can also use them to control or limit the speed.

Power Monitor

A power monitor lets you keep an eye on the state of the battery. A fully charged battery behaves somewhat differently than a (partly) discharged battery. This has an incidence on how efficient the motors respond to changes in their control signals. The self-balancing algorithm and the PID controller can be made more performant by taking the battery level and current consumption into account. The *SBRobot-Dabble* sketch shows how to read the power monitor (compile the sketch with `__VERBOSE__` enabled to activate its output on the serial port, see around line 54).

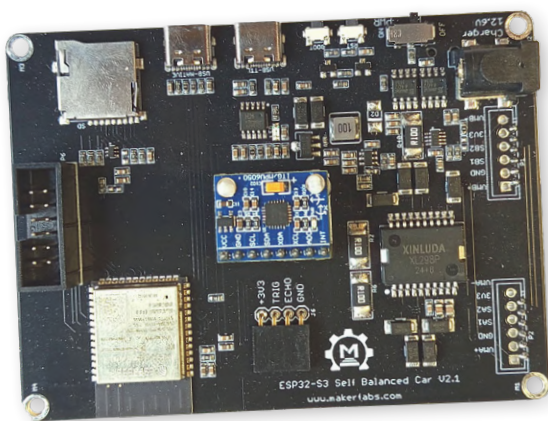


Figure 4: The ESP32-S3-based main board of the Mini-Wheelie is also an excellent platform for other projects unrelated to robotics.

What is a PID Controller?

A Proportional-Integral-Derivative controller, better known as PID controller, is a mechanism used to maintain a desired setpoint of a system by continuously adjusting the output based on three terms:

1. Proportional (P): Responds to the current error, i.e. the difference between setpoint and process variable.
2. Integral (I): Accounts for the accumulation of past errors to eliminate steady-state offset.
3. Derivative (D): Predicts future error based on its rate of change, improving stability and response speed.

To make this abstract definition a bit more tangible, think of controlling the speed of a car. In this scenario the setpoint is the desired speed, e.g. 100 km/h, while the process variable is the measured speed as indicated by the speedometer of the car. Suppose our objective is to keep the speed constant. There are several ways to achieve this:

1. Adjust the pressure of your foot on the accelerator pedal to keep the value shown on the speedometer constant. In this case, the speed v is directly related to the force F on the accelerator. This is proportional control (P). The car responds quickly, but it is not so easy to maintain a super constant speed:

$$v = k \cdot F_{\text{accelerator}} = P$$

2. Measure the distance travelled during a defined period, say five minutes. When the travelled distance per time period is constant, the average speed per time period is constant too. Note the word 'average.' Distance d is the integral (I) of speed:

$$d = v \cdot dt = I$$

3. To keep the speed of the car constant, it should not accelerate or decelerate. The acceleration must be zero. So, if you measure the acceleration and try to keep it zero, the speed of the car will be constant. Acceleration a is the derivative (D) of speed:

$$a = dv/dt = D$$

By combining these three methods, the speed of the car can be controlled very precisely and remain perfectly constant:

$$PID = k_p \cdot P + k_i \cdot I + k_d \cdot D$$

Here, PID is the signal that controls the accelerator of the car (e.g., the force you apply to the pedal). The hard part is finding the values of the constants k_p , k_i and k_d . Algorithms exist to do this in a more or less theoretically-controlled way, but they can also be obtained by experimenting when the system is non-critical and inexpensive, like a small self-balancing robot. For the cooling system of a nuclear power plant, you would probably first try with an algorithm. An interesting tool to optimize PID parameters in (Arduino) code can be found in this article [2].

Note that you don't always need all three values to control a system. P and PI controllers are common too, PD a bit less. ID controllers, i.e., without P, do exist but are only used in rare cases.

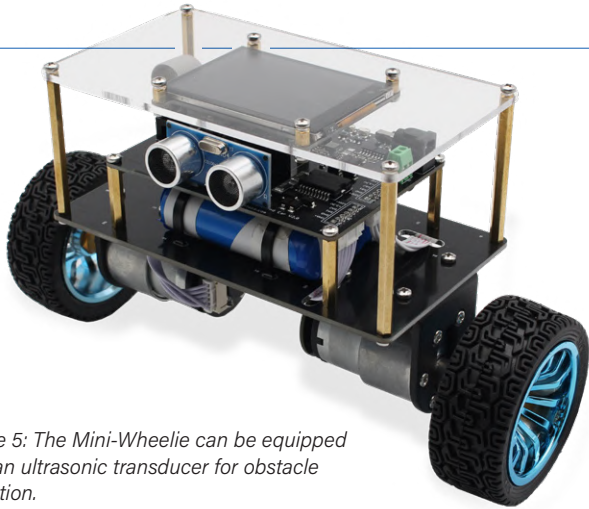


Figure 5: The Mini-Wheelie can be equipped with an ultrasonic transducer for obstacle detection.

Obstacle Detection

To enable obstacle detection, an ultrasonic transducer can be connected to J6 (Figure 5). This is a common HC-SR04-type transducer used in numerous projects found online. This sensor is supported by many Arduino libraries. A simple suitable library is e.g. *Bifrost library for HC-SR04*. In the *SBRobotDabble* sketch, the transducer is connected to pins `ULTRASONIC_TRIG` and `ULTRASONIC_ECHO`. After installing this library, using the transducer is as simple as adding:

```
#include <hcsr04.h>
HCSR04 hcsr04(ULTRASONIC_TRIG, ULTRASONIC_ECHO, range_min, range_max);
```

Here, `range_min` and `range_max` define the detection range in millimetres. In the main program, call the function below to obtain a reading:

```
int distance = hcsr04.distanceInMillimeters();
```

Advanced Features

Strictly speaking, the Mini-Wheelie doesn't need a full-colour graphical display, but it has one anyway. It can be used to implement a face-like interface or so. It can be practical however, for displaying debug information while driving around and trying to improve self-balancing or obstacle detection.

A microSD card slot is available for logging data and for storing images and robot parameters. The *SBRobotDabble* sketch contains functions to read JPEG images from the SD card and to display them. Also functions for writing simple text and floating-point values are included.

Multitasking

It is important to keep in mind that sending large amounts of data to the display slows down the control loop, and even more so if it consists of JPEG encoded images. When the control loop is interrupted for too long,

the robot will fall over. This is also true for remote control and SD-card operations. This is why, in the *SBRobotDabble* sketch, the Dabble library was modified to allow calling it repeatedly (`Dabble.processInput_tick`) instead of using its blocking default message handler. A good solution to this issue would be to use a multitasking operating system where one task handles the self-balancing algorithm, and the other(s) take care of the display and similar time-consuming tasks (like a webserver).

USB-C

You probably noticed that the Mini-Wheelie is equipped with two USB-C ports. The one labelled *USB-NATIVE* is connected to the ESP32-S3's built-in USB controller. This is a so-called "future expansion" and hasn't been used yet. For programming the microcontroller and debugging over the serial port, use the port labelled *USB-TTL*.

Experimentation and Learning

The Elektor Mini-Wheelie is a hands-on self-balancing robot designed for experimentation and learning. With features like Hall-effect sensors, power monitoring, and ultrasonic obstacle detection, it offers opportunities to explore advanced control techniques and sensor integration. Beyond just balancing, the Mini-Wheelie challenges you to refine its movement, optimize its PID controller, and experiment with multitasking, making it an excellent introduction to embedded systems and real-world robotics applications. Enjoy! ◀

240134-01

Questions or Comments?

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

About the Author

Clemens Valens started working for Elektor in 2008, and he has held various positions since. Currently, he is part of the product development team. His main interests in electronics are (digital) signal processing and its applications in music production and sound synthesis.



Related Product

► **Elektor Mini-Wheelie Self-Balancing Robot**
www.elektor.com/21087

WEB LINKS

- [1] This project on Elektor Labs: <https://www.elektormagazine.com/labs/self-balancing-robot-with-maker-fabs>
- [2] Johannes Sturz, "PID Control Tool," Elektor 5-6/2025: <http://www.elektormagazine.com/240274-01>

Solar Cells

By David Ashton (Australia)

Solar cells have, over time, become an entirely commonplace component in a myriad of different sizes and applications, becoming part of our daily lives. Let's look at them in a brief overview.

I have always found something magical about the idea of solar cells being able to generate sometimes large quantities of electricity just using sunlight. For this, they depend on the photovoltaic effect. This was first noted by Edmond Becquerel, who, in 1839, demonstrated the production of an electric current when two plates of gold or platinum in a solution were unevenly exposed to sunlight.

In 1884, Charles Frits produced an experimental solar cell using a thin layer of selenium covered with a thin film of gold, but the efficiency was poor.

Modern solar cells are based on semiconductor P-N junctions, much like LEDs. In fact, an LED will generate a voltage when exposed to light — I just got 1.46 V out of a clear red LED with a powerful torch next to it. The current would be very low, though. The polarity is the opposite of what you'd normally use to illuminate an LED.

General Features

Solar cells come in a wide variety of sizes, from a thumb-nail-sized cell in an ornamental solar garden light, up to the solar panels on my roof, which generate 440 W each in full sunlight and are nearly 2 square meters in size. Ever larger wattages are available as the technology gets better, and efficiencies are up to 25% for average

panels, as much as 40 to 50% for more expensive types. When you consider that solar radiation is up to 1000 W per square meter, that's a lot of power available.

Solar cells may be monocrystalline or polycrystalline. Monocrystalline cells are made from a large bar of crystalline silicon cut into thin wafers, and are usually black, whereas polycrystalline cells are made from fragments of silicon melted together, and are usually a blue color. Monocrystalline cells are thought to be better and more efficient than polycrystalline cells, but the differences are small and the quality of manufacturing probably has more effect than the type used.

Perovskite solar cells are another type of solar cell. Perovskites are a class of crystalline materials having the same structure as calcium titanate, but include a host of organic and inorganic compounds. Perovskite solar cells are cheaper to produce, easier to make flexible, and exhibit better performance under low light conditions such as cloudy days. Some perovskites use lead compounds, which has led to resistance to their use. They have good efficiency, but some perovskites exhibit chemical instability, especially in the presence of moisture. This has resulted in relatively limited acceptance, but research is overcoming these problems.



Figure 1: The small PV cell that powers the rooftop sensors of a weather station. (Courtesy of: Bresser GmbH, Germany — bresser.de)



Figure 2: A rooftop photo-voltaic high-power installation. (Source: Wikimedia Commons — <https://commons.wikimedia.org/wiki/File:Photovoltaikanlage.jpg>)



Recent Applications

With the advent of very-low-power microcontrollers with wireless features, small solar cells are seeing wide use in remote environmental sensors, for example the outdoor sensor units for weather stations (**Figure 1**). Teamed with lithium batteries, they can power many remote sensors for IoT and other applications. Solar garden lights and ornaments are now ubiquitous, and discarded ones are a good source of small solar cells for the inquisitive hobbyist.

Larger solar installations, such as on house roofs or in solar farms, are helping to speed the demise of fossil-fuel-based power generation. Australia has one of the highest rates of domestic solar power in the world, and large solar plants are being constructed everywhere. Battery storage to store some of the power and supply it at night is advancing rapidly and contributing to the uptake of solar power. **Figure 2** shows a large rooftop solar power installation. ◀

About the Author

David Ashton was born in London, grew up in Rhodesia (now Zimbabwe), lived and worked in Zimbabwe, and now lives in Australia. He has been interested in electronics since he was “knee-high to a grasshopper.” Rhodesia was not the center of the electronics universe, so adapting, substituting, and scrounging components were skills he acquired early (and still prides himself on). He has run an electronics lab, but has worked mainly in telecommunications.

Questions or Comments?

If you have technical questions or comments about this article, feel free to contact the Elektor editorial team at editor@elektor.com.

250146-01

Getting Started With a Modern Radar Sensor

Is an Accurate Measurement on Your Radar?

By Jean-François Simon (Elektor)

Let's get started with the A121 radar sensor from Acconeer, a very powerful but not so easy-to-master sensor. Besides some background information, we provide a lot of tips for putting it to use!

Sensors are fascinating. Some time ago, while I was doing research to build a mailbox mail indicator [1], I came across an interesting video by Andreas Spiess [2] that, among other things, mentioned a very powerful radar sensor, the A121 from the Swedish company Acconeer, costing about €12. This strongly intrigued me and made me want to give it a try! The manufacturer's official development kit, the XE125, is quite expensive (around €120), but fortunately, SparkFun also offers a development board based on the same sensor for \$50. This is better, and the price helps support a company that does a great job, with their modules and documentation, of making electronics accessible to engineers of all levels. The fact that SparkFun and Elektor are business partners did not influence my choice, and I am glad to have paid for this module with my own money. That said, let's dive in!

Radar 101

While the general principle of a radar is to emit a radio signal toward a target and analyze the reflected signal, there are different types. In continuous wave (CW) radar, like the HB100 previously featured in Elektor [3] or the similar CDM324, a constant frequency signal is transmitted and received simultaneously, allowing measurement of the target's velocity through Doppler shift, but not its distance. Frequency-modulated continuous wave (FMCW) radar improves on this by continuously varying the transmitted signal's frequency over time. As the wave takes time to travel to and from the target, at any given instant the received signal is a different frequency than the transmitted signal. By measuring the difference between those frequencies, the distance to the object can be computed.

In contrast, Pulsed Radar transmit short bursts of signal at a single frequency and listens for echoes between pulses. Measuring the time

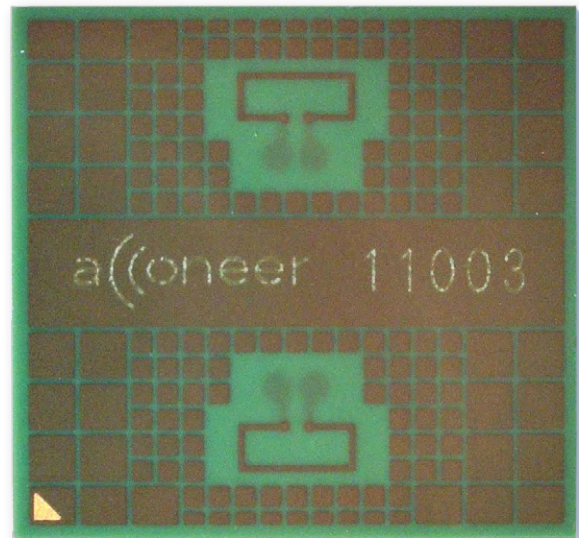


Figure 1: The A121 is tiny, less than 6 mm square. The two 60 GHz antennas are visible.

t elapsed between transmission and reception of the reflected signal gives the measured distance d by the equation $d = 1/2 * t * v$, where v is the propagation speed of the wave, which depends on the material. The A121 (Figure 1) is a Pulsed Coherent Radar, coherent meaning that it is able to measure the phase of the received signal for even more precise measurement.

Hardware Overview

The A121 [4] uses the 60 GHz band and offers a lot of features within a very small form factor (5.2 x 5.5 x 0.8 mm). It is fully integrated, with the baseband processor, RF front-end, and antenna in the same package, and a digital output via SPI. Internally, the A121 measures time differences with a resolution on the order of a picosecond, which provides excellent distance resolution, on the order of 0.1 mm (!). The maximum range is about 20 m. This is a very complex component; explaining it in detail would both be very hard, and go far beyond the scope of this article, but the manufacturer's documentation (see [5] and [6]) is very good and abundant.

Acconeer suggests applications like distance measurement, proximity or presence detection, measuring velocity, tracking objects, robot

navigation, etc. Thanks to its high resolution in distance mode, the A121 can also monitor vital life signs such as breathing and pulse rate. As the sensor outputs phase and amplitude of the received signal, it's possible (by processing the data on a microcontroller connected to it) to detect and classify different materials, and recognize basic gestures. As other radar sensors, it's immune to dust, variations of ambient light, etc.

All this is excellent, but on its own, this component is not very easy to discover and experiment with, for two reasons: first, this tiny package is a 50-ball BGA, which is not so easy to solder. Second, using the advanced features requires a lot of data processing through software, which represents a significant amount of work for anyone just considering trying out the component. To address this, Acconeer has a great solution.

The XM125 Module

The Acconeer XM125 module [7] (**Figure 2**) makes it much easier to explore the product. It is a small rectangular PCB measuring 18.6 mm x 15 mm, containing: the A121 sensor, a 24-MHz crystal required for its operation, a few resistors and capacitors, and finally an STM32L431CBY6 microcontroller from STMicroelectronics. All components are mounted on the top side, allowing the module to be soldered flat onto a Land Grid Array (LGA) footprint on your prototype PCB, using hot air or a soldering iron. Of the 28 pins with a 1.27-mm pitch (more accessible than BGA for us mortals), only a few are absolutely necessary; it would even be possible to solder wires directly for prototyping without a PCB.

The module is compact enough to be directly integrated into commercial products. It may function as a stand-alone unit, allowing customers to build their applications on top of the Acconeer software. Alternatively, it can be paired with an external microcontroller, communicating with the module using a register command protocol via a serial port or I²C.

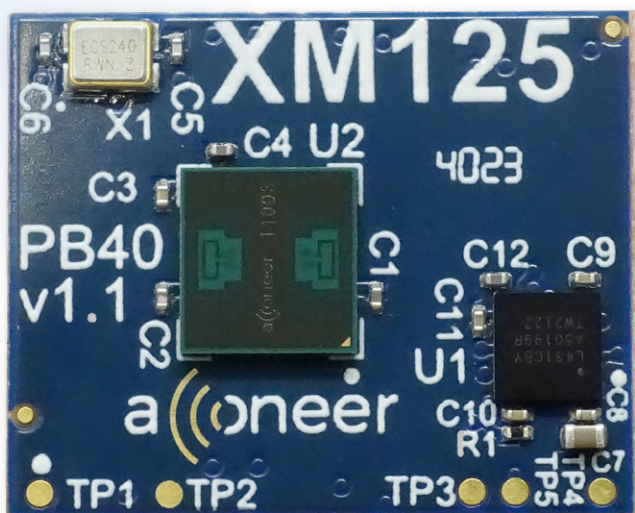


Figure 2: The XM125 modules makes development easier.

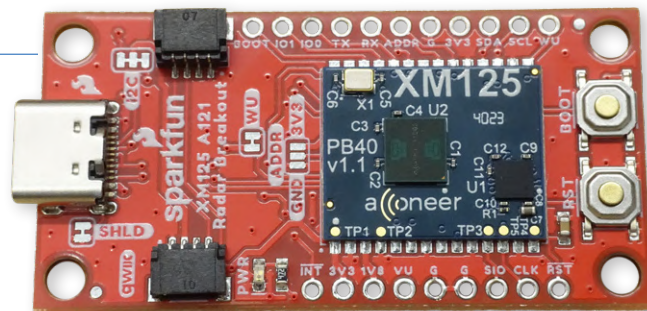


Figure 3: A PCB by SparkFun provides USB and everything needed.

A SparkFun Breakout Board

Integrating a microcontroller into the module was a smart choice. This way, Acconeer can offer downloadable firmware files that are already compiled and ready to be flashed onto the STM32L431, making it much easier to try out the sensor. SparkFun [8] makes things even more convenient, with a PCB that includes the XM125, a CH340C USB-to-UART converter, a USB-C port, and the necessary voltage regulators, as can be seen on **Figure 3**. Two Qwiic connectors allow the module to be linked via I²C to other SparkFun boards.

This breakout board is simply intended to make connections easier; it does not include an additional microcontroller. SparkFun suggests to use the provided binaries as-is for the STM32L431, and to develop your own application on an external microcontroller, for example an ESP32 on an extra board, communicating with the XM125 via I²C. For more advanced users, or those familiar with STM32 products, it is also possible to do without an external microcontroller and to fully develop your project on the provided STM32L431.

Software

Acconeer offers many options in terms of software. For the XM125, a Software Development Kit (SDK) is provided [6] and contains source code and compiled files, directly transferable to the onboard STM32L431, for many example applications (distance measurement, speed, presence detection, etc.). For those planning to use the A121 sensor in a fully custom project without the XM125 module, other SDKs are also available in different variants (Keil, ARM-GCC, IAR, etc.) for ARM Cortex M7, M4, M33, M0 platforms. A dedicated PDF guide is available for each case.

Special mention goes to the *Exploration Tool* published by Acconeer, this time on GitHub [9]. Fully programmed in Python, it provides a useful graphical interface on PC (Windows or Linux) to evaluate the A121, modify its parameters, and stream live measurements to the PC. Let's take a closer look.

The Acconeer Exploration Tool

For those who have never installed the Python interpreter on their PC, now is the time to do so. The Exploration Tool is installed using the pip package manager. However, some Linux distributions (like Mint, which I recently tried) strongly advise against using pip on the system's Python interpreter, as there's apparently a risk of messing with packages used by the system itself. In these cases, it is recommended to run pip inside a virtual environment. It may also be worth doing the same on Windows to keep control over the packages you install or uninstall for each project. I used PyCharm, which works well on both Windows and Linux.

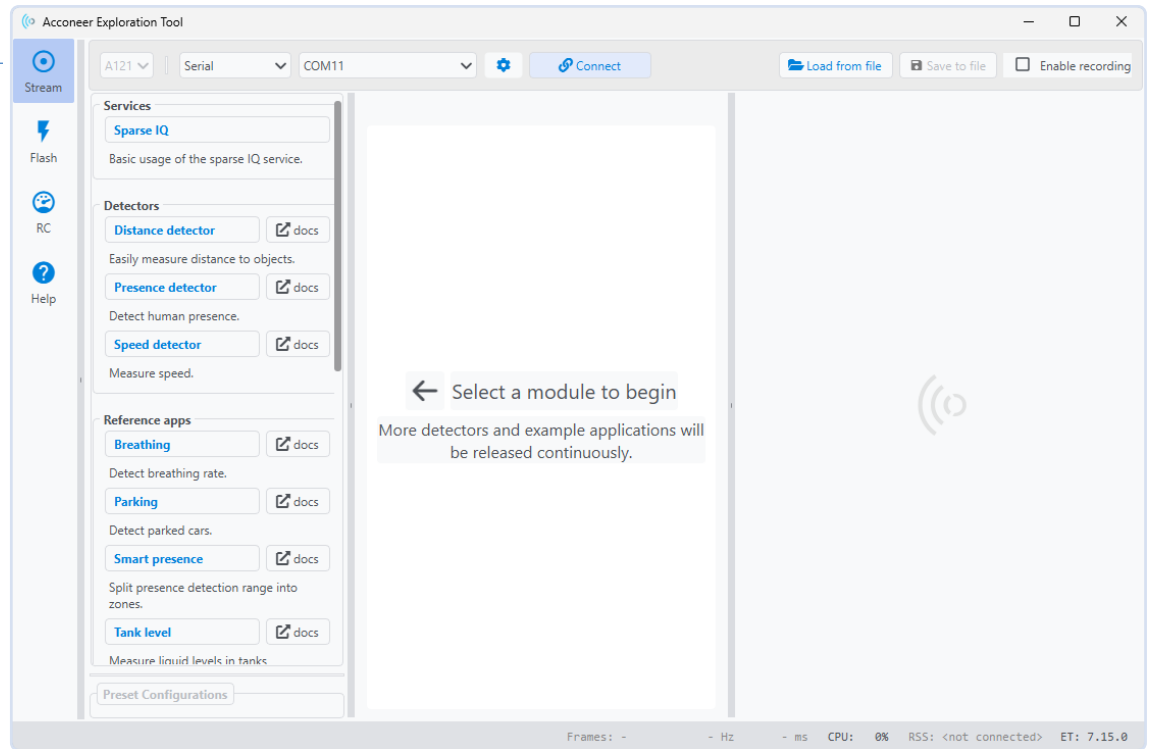


Figure 4:
Main window of the
Exploration Tool at
startup.

In the terminal (of PyCharm or your system), install the Exploration Tool with the command `python -m pip install --upgrade acconeer-exptool[app]` and, once all packages are installed, launch the GUI interface with the command `python -m acconeer.exptool.app`. The Exploration Tool also supports the predecessor of the A121, the A111. Click the left panel to open the A121 version. The main window that opens is shown in **Figure 4**.

Firmware Download

Just as each specific situation (distance, speed, presence measurement, etc.) has its own optimized firmware available in binary format, the Exploration Tool also needs a specific firmware, called the *Exploration Server*. To start, create an account at [10]. Then download the Exploration Server firmware at [6] and unzip the ZIP file.

The Exploration Tool contains four distinct sections: *Stream*, *Flash*, *Resource Calculator* (RC), and *Help*. These are accessible through buttons on the left. First, open the *Flash* page (**Figure 5**). In principle,

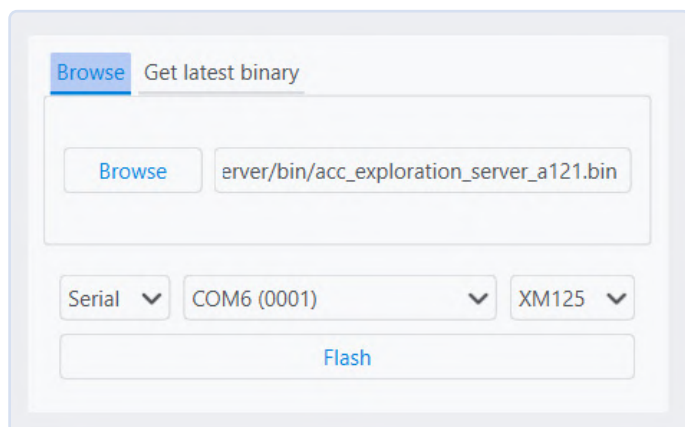


Figure 5: Preparing to flash firmware.

it is possible to directly download the latest firmware version under the *Get Latest Binary* tab. As of the time of writing, a bug seems to exist: the firmware `acconeer_xc120_exploration_server_a121-v1_10_0.bin` downloaded via the Exploration Tool does not work properly. Although flashing seems normal, trying to connect to the module afterward results in a connection failure (**Figure 6**). Among other minor issues, the link provided by the Exploration Tool to create a Developer account returns a 404 error. Use [10] instead.

To work around the firmware bug, select the *Browse* tab and manually select the firmware file from the ZIP you downloaded earlier. Choose *Serial*, then the correct serial port (on Windows, you can check the port number in the Device Manager when plugging in the module), then *XM125*. Finally, click *Flash* and follow the on-screen instructions: press the *Boot* button, then the *Reset* button while keeping *Boot* pressed, release *Reset*, and then release *Boot*. I recommend not doing this sequence too quickly; leave at least a third to half a second between each press. Flashing should complete normally.

First Tests

Now, click on the *Advanced Settings* button (to the left of *Connect* in the top bar) and type `115200` in the *Baudrate* field. Close *Advanced Settings* and click *Connect*. Congratulations, your module is connected!

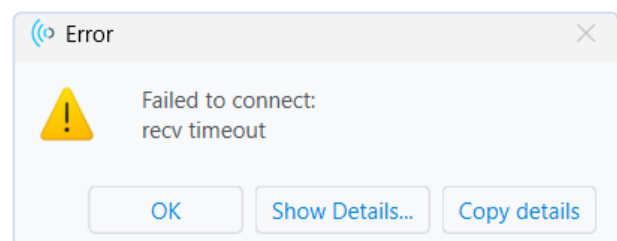


Figure 6: I pulled my hair out over this error for quite a while.

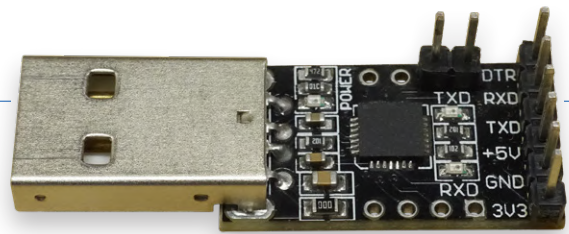


Figure 8: A CP2102-based module with the additional RTS and CTS pins (required).

You can now try out the functionalities of the Exploration Tool. The different functions are listed on the left panel: *Sparse IQ*, *Distance Detector*, *Presence Detector*, *Speed Detector*, etc. For each application, three display areas are used (Figure 7): *Preset Configurations* at the bottom of the left panel (label 1 in the figure), detailed settings on the right panel (2), and measurement results in the center (3). Each application must be started with the *Start measurement* button. Feel free to explore the various demo applications. The *Sparse IQ* page displays the raw data output from the sensor, in the form of the amplitude and phase of the received signals, represented as complex numbers. It may be a bit difficult to interpret these values by eye. Although it is listed first, it is not the most fun. *Presence Detector* and *Touchless Button* are easier to understand.

Make It Faster!

After a few tests, I realized that the graph display was not very smooth: data arrives in bursts. While some applications like *Touchless Button* work fairly well, there are five-second pauses between data bursts and refreshing of the curves in *Sparse IQ*, *Presence Detector*, and others. It's a bit frustrating considering the sensor's speed and all the effort Acconeer put into creating a great software tool.

It turns out neither Acconeer nor the sensor is to blame. It's an unfortunate coincidence that SparkFun hasn't corrected yet. The CH340C from WCH, a very popular USB-to-UART converter used in several SparkFun boards, was used here again, which is logical for SparkFun. Unfortunately, it does not fully support Hardware Flow Control,

even though it has RTS and CTS pins (required by the XM125). The CH340C datasheet version 1D is vague about this: "The USB to serial interface is only compatible with the application layer, not totally". Later, the version 3B of the same document [11] says: "For one-directional 1 Mbps and above, or bi-directional 500 kbps and above, we recommended to use CH343, and enable automatic hardware flow control."

What a pity! Here, the XM125 and Exploration Tool need Hardware Flow Control to optimize data throughput. On the XE125 (official Acconeer dev kit), a Silicon Labs CP2105 is used instead. Maybe SparkFun will release a Revision 2? For now, SparkFun's Hookup Guide [12] notes: "Note: The Acconeer Exploration Tool may run slower than expected when using the SparkFun XM125 Pulsed Coherent Radar Sensor." Acconeer's Exploration Tool FAQ also confirms this.

I didn't have a CP2105-based module on hand to try; the CP2105 Friend from Adafruit [13] would be ideal but was out of stock when writing. However, the CP2102 and CP2104 are quite similar and worth trying. They are used in many products – I had several gadgets on hand using them – but RTS and CTS pins are not always accessible. Digging through my inventory, I found a USB-serial adapter with a CP2102 where the necessary pins are exposed (Figure 8). When buying, check the available pins carefully in the photos or documentation.

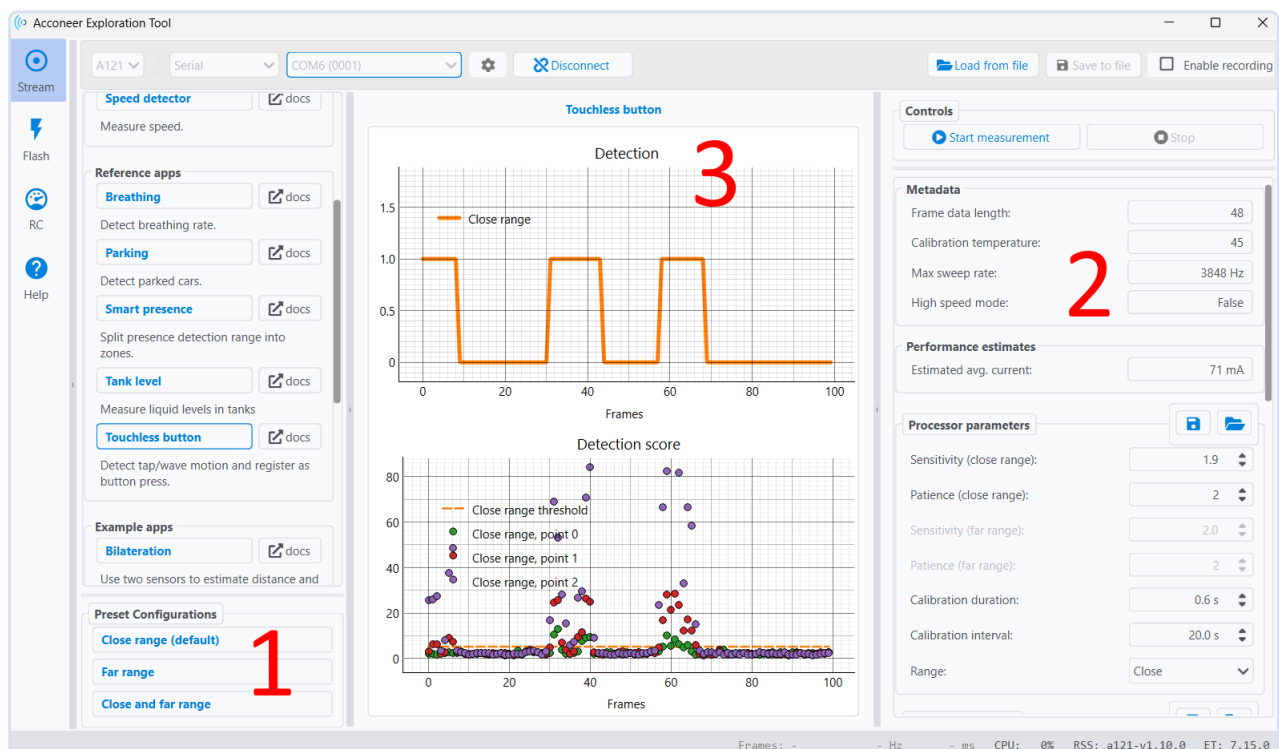


Figure 7: Three main sections in the Exploration Tool.

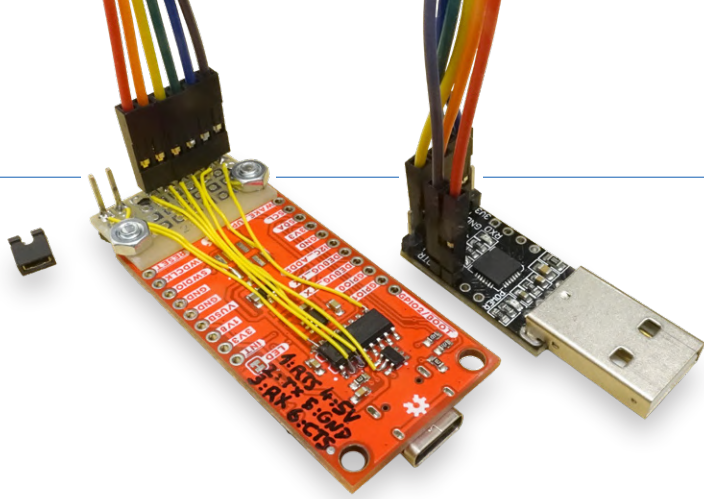


Figure 9: Modifying the wiring to support other USB to UART converters.

As you're not supposed to use a different USB to UART converter, there is no easy access to the primary TX, RX, CTS, and RTS signals; those are directly routed from the XM125 to the CH340C. The DEBUG_TX and DEBUG_RX pins on the side of the module are connected to the second serial port of the XM125, which is used for debugging only. This required some adaptation (**Figure 9**).

A small piece of perfboard was added to the sensor's back side, and a pin header was wired to the corresponding pins of the CH340 using thin wire. If you attempt this, pins TX, RX, CTS, and RTS are pins 2, 3, 9, and 14 on the CH340C, respectively. You can also solder directly onto the XM125 pads on the front side. To disable the CH340C, its VCC pin (pin 16) must be disconnected, either by cutting the trace

near the pin or carefully lifting the pin and removing solder. I added a jumper to make this reversible. Two extra wires (+5 V and GND) were also routed to the pin header to eliminate the original USB-C cable.

To use the CP210x instead of the CH340, simply change the COM port number in the Exploration Tool. In my case, the CP2102 worked properly, and the graphs now refresh continuously, between 2 and 20 times per second depending on the application. Much better! Another module based on the FT232RL by FTDI worked as well, but was slightly slower.

A Solution Looking for a Problem

The Presence Detector illustrates how the radar's sensitivity to micro-movements gives an advantage over other types of sensors. I tried sitting perfectly still a few meters away from the sensor: no matter what, my tiny involuntary movements were detected by the A121, and the Exploration Tool clearly indicated I was always present, which is reassuring in case of an existential crisis. On the other hand, the Passive Infrared (PIR) sensor placed at the same distance ignored me after a few seconds if I stayed still. More interesting applications include remote vibration measurement of machines, or non-invasively monitoring people's breathing. I tried that (see **Figure 10**), but there seems to be something wrong: During the test, I was calm and

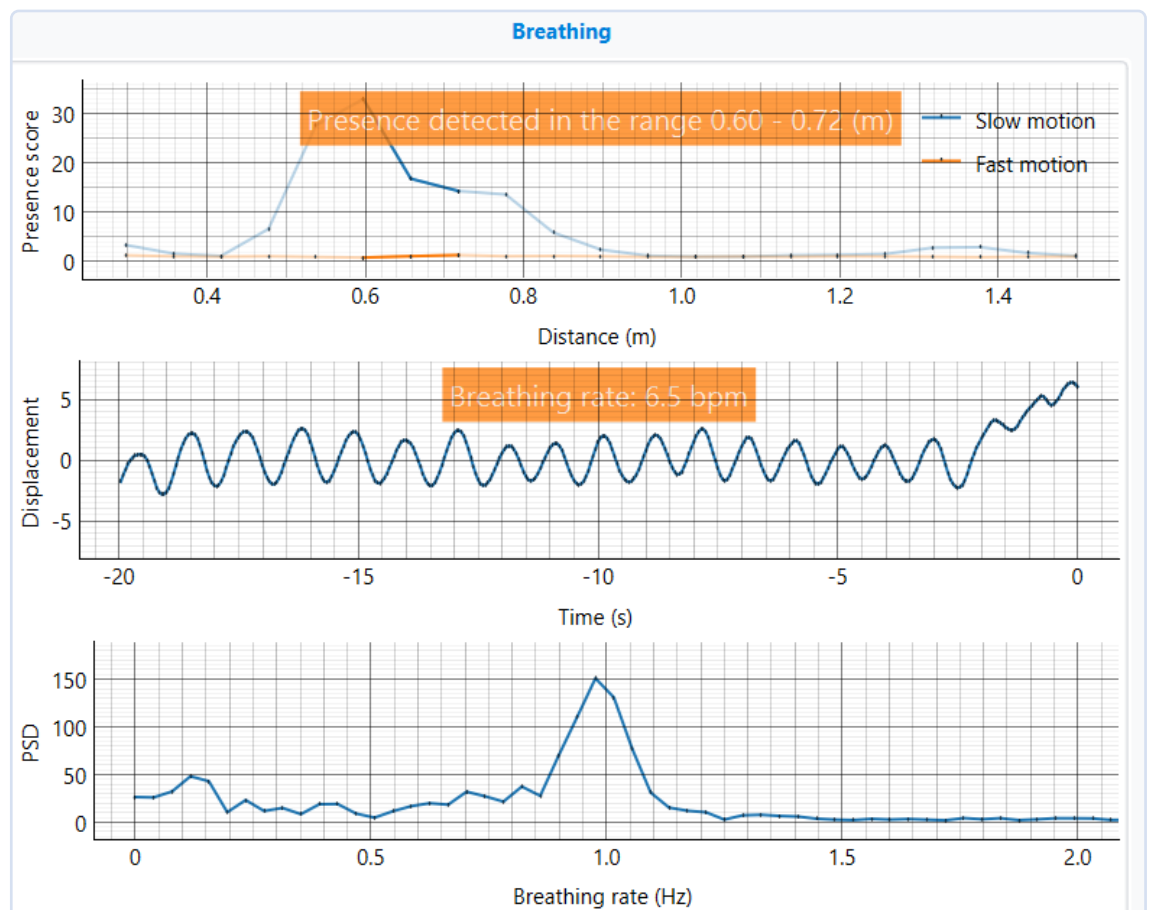


Figure 10: Example of breathing detection. However, the breathing rate value in bpm does not fit to the curve.

breathing much more slowly than what is shown. Moreover, the indication in bpm (breaths per minute) does not match the curve either. I will investigate and keep you informed in our Err-electronics section if I find the bug. Personally, I haven't yet found a direct use for the sensor at home, but studying it in detail is still a very satisfactory process. Feel free to email me your ideas or suggestions!

Going Further

It is recommended to tinker with the various settings of the Exploration Tool for yourself! Also have a look at the *Resource Calculator* panel, which provides estimates (not measurements) of the module's power consumption based on the settings. Also check out SparkFun's Hookup Guide, which offers an Arduino library [14] that you can use on an external microcontroller of your choice, to help you communicate with the XM125 in Presence and Distance modes, through I²C. I encourage reading Acconeer's documentation for a detailed understanding, and for the brave, taking a look at the source files. To develop your own application, you can also install STM32CubeIDE and build upon the Acconeer SDK with the STM32L431.

Do I recommend the SparkFun XM125 module? Hard to say. On paper, it's great to have the same kind of functionality as the official Acconeer kit, at 40% of the price. If you mainly use the XM125 in Presence or Distance mode and, like SparkFun, mainly use the XM125 through its I²C interface, then the CH340C's slowness doesn't really matter. For those who want a smoother experience with the Exploration Tool and save some money, buying the XM125 alone, a separate USB-UART converter, and making a simple adapter PCB would be a better option. Have fun! ◀

250167-01

Questions or Comments?

Do you have questions or comments about this article? Email the author at jean-francois.simon@elektor.com, or contact Elektor at editor@elektor.com.



About the Author

Jean-Francois Simon (Engineer, Elektor) has a longstanding passion for electronics and enjoys topics as varied as circuit design, test and measurement, prototyping, playing with SDRs, and more. He likes to create, modify and improve his tools and other systems. He has an engineering background and also enjoys mechanics, machining, and all things technical. Follow him on X at x.com/JFS_Elektor.



Related Products

- **YDLIDAR X4Pro Lidar – 360-degree Laser Range Scanner**
www.elektor.com/20546
- **Universal Maker Sensor Kit**
www.elektor.com/21094



WEB LINKS

- [1] Jean-Francois Simon, "Wireless Mailbox Notifier," Elektor 7-8/2025: <http://www.elektormagazine.com/240715-01>
- [2] Andreas Spiess, "Which Radar Sensor Is Best?," YouTube: <https://www.youtube.com/watch?v=s-GzUTyIH9c>
- [3] Stefano Lovati, "The HB100 Doppler Motion Sensor", Elektor 7-8/2023 : <http://www.elektormagazine.com/230205-01>
- [4] The A121 Radar Sensor: <https://www.acconeer.com/products/>
- [5] Acconeer Documentation (1/2): <https://docs.acconeer.com/en/latest/index.html>
- [6] Acconeer Documentation (2/2) and SDK: <https://developer.acconeer.com/home/a121-docs-software/xm125-xe125/>
- [7] XM125 Datasheet: <https://developer.acconeer.com/download/xm125-datasheet/>
- [8] SparkFun XM125 Board: <https://www.sparkfun.com/products/24540>
- [9] Acconeer Exploration Tool: <https://github.com/acconeer/acconeer-python-exploration>
- [10] Create Developer Account: <https://developer.acconeer.com/register/>
- [11] CH340C Datasheet, Version 3B: <https://tinyurl.com/2a6nawzt>
- [12] SparkFun XM125 Hookup Guide: https://docs.sparkfun.com/SparkFun_Qwiic_Pulsed_Radar_Sensor_XM125
- [13] Adafruit CP2105 Friend: <https://www.adafruit.com/product/6065>
- [14] SparkFun XM125 Arduino Library, GitHub: https://github.com/sparkfun/SparkFun_Qwiic_XM125_Arduino_Library



Source: Adobe Stock / mimadeo - generated with AI

From Life's Experience

Paper Factory

By Ilse Joostens (Belgium)

You are probably familiar with the saying “don’t count your chickens before they hatch.” Despite the important life lesson behind it, it seems that our more commercially minded fellow human beings do not have much affinity with this saying. The same goes for politicians, of course, but even academic institutions and scientists are guilty of this all too often.

A few years ago, I came into contact with the founders of a start-up that wanted to market a revolutionary GPS-tracker to combat bicycle theft. It was a good idea in itself, and not only were the newspapers full of it, but Flemish television also devoted considerable attention to it. However, apart from a clumsily constructed and non-functional prototype, there was no actual product available. It will come as no surprise that the story ultimately came to nothing, and not only that: a Czech entrepreneur even tried to hijack the idea. So, it’s better to have a solid, working product before you start touting it to the world.

Expensive Cells

Have you ever noticed that groundbreaking discoveries and technologies are announced with clockwork regularity in the mainstream media (but also in trade magazines) and that you usually don’t hear much more about them afterwards? It seems as if every time, the solution to one of the world’s major problems is a step closer. In electronics, too, new discoveries follow each other in rapid succession. The reasons why they ultimately come to nothing are diverse, ranging from outright fraud — such as cold fusion, STAP cells (Stimulus-Triggered Acquisition of

Pluripotency), organic semiconductors (the Schön scandal [1]), superconductivity at room temperature, and Theranos’ blood tests — to public rejection — such as the protests by Greenpeace and others against genetically modified “golden rice” [2], which produces beta-carotene, a precursor to vitamin A, and could save the lives of more than a million children a year in the Third World (**Figure 1**).

Quite a few scientific experiments look promising in the lab or on a small scale but turn out to be disappointing in the real world. For example, bleach works very well in vitro against most pathogens, but that does not mean that you can simply treat sick people with an injection of the stuff. Take graphene, discovered in 2004 [3]. On paper, the possibilities of this material seemed enormous, and there was much speculation about the most spectacular applications,



Figure 1: Who is afraid of golden rice? (Source: Adobe Stock / Prakrong — generated with AI)



Figure 2: Will the space elevator ever be built? (Source: Adobe Stock / IbotDesign)



Figure 3: Hydrogen from sunlight — it seems too good to be true. (Source: Adobe Stock / NKCoolper — generated with AI)

from replacing silicon in computer processors to extremely thin rollable television screens and even a real space elevator (**Figure 2**). Now that the hype has died down, more realistic commercial applications have appeared on the market, including (bio)sensors and solutions for cooling chips in smartphones. And how about roads covered with solar panels, printed solar panels on thin film, tubular solar panels for flat roofs [4], and solar panels that electrochemically generate hydrogen gas from water (**Figure 3**)? Unfortunately, none of these ideas came to fruition for technical and/or economic reasons. Currently, Solhyd [5], a spin-off from KU Leuven, is working on solar panels that extract water from the air at night and generate hydrogen gas during the day using electrical energy from sunlight through electrolysis. They have clearly learned from the mistakes of their predecessors, and the technology looks promising, so let's keep our fingers crossed that they succeed in commercializing these panels.

On a larger scale, too, mistakes are sometimes made. Take Northvolt [6] in Skellefteå, Sweden, for example, which

in April 2024 was still being talked about in superlatives as the company with the ambition to become the world's largest green battery manufacturer and to build a dam against Chinese, Korean, and American battery dominance. Barely six months later, the company's trajectory resembled more of an Icarus flight due to production problems combined with the choice of "expensive cells" with NMC technology (nickel-manganese-cobalt), a technology for lithium-ion cells that is considerably more expensive than LFP technology (LiFePO₄ or lithium iron phosphate). LFP was long considered inferior, but China has nevertheless succeeded in using this technology to produce high-performance and cheaper cells for electric vehicles.

Of course, things could always be worse, and I'm not going to claim that nuclear energy is the best alternative to fossil fuels, but closing all nuclear reactors and fully committing to a hydrogen economy that doesn't yet exist, as Germany has done, was a huge mistake in my opinion [7][8]. And I can see that hydrogen bubble bursting in the not too distant future.

Putting the Cart Before the Horse

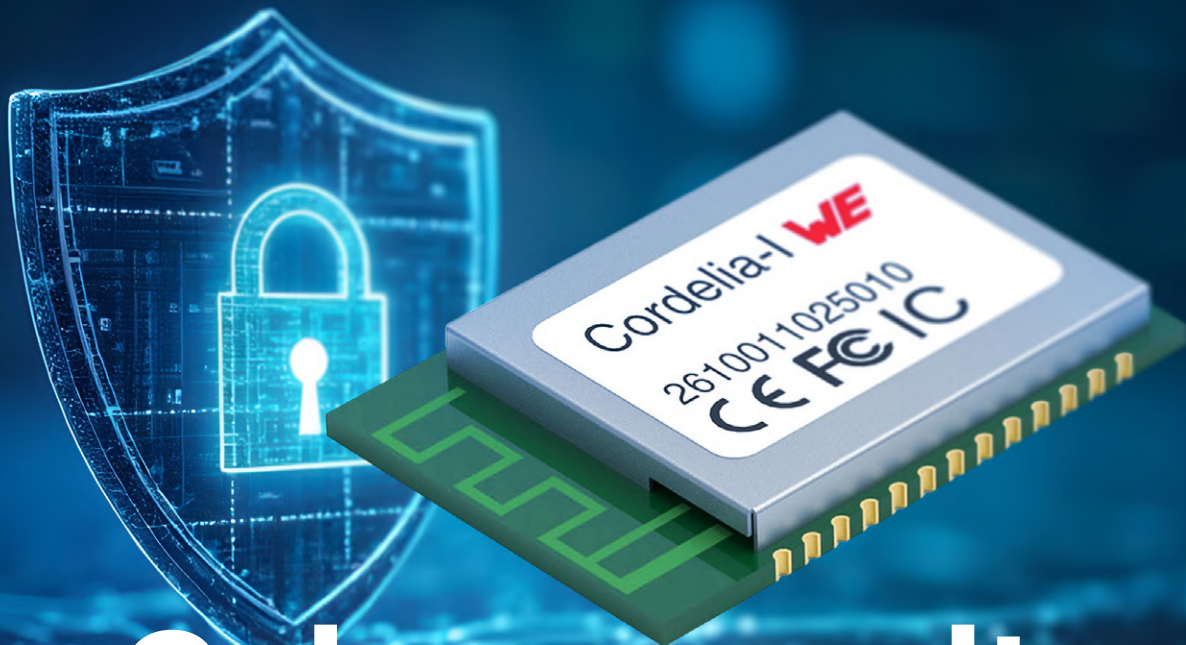
Nowadays, it seems as if people in academia publish first and think later. The reasons for publishing as much as possible are diverse, such as publication pressure (publish or perish), employment, attracting students, subsidies, ego and prestige, creating hype, obtaining immigration visas, etc. The list is long, and studies show that a significant proportion of published papers and doctoral theses are hardly ever cited or read, let alone applied, making them little more than paper factories [9][10].

I am not advocating a Trump-style crack-down on the academic world, and not all research needs to be economically useful, but reforms to maintain real, lasting impact do seem sensible to me. No one benefits from research inflation and the creation of a scientific bubble. Moreover, it does no harm to think critically for oneself and realize that scientists, researchers, and politicians are, after all, just ordinary people. ◀

Translated by Hans Adams — 250311-01

WEB LINKS

- [1] Wikipedia: Schön scandal: https://en.wikipedia.org/wiki/Sch%C3%B6n_scandal
- [2] Wikipedia: Golden rice: https://en.wikipedia.org/wiki/Golden_rice
- [3] Mark Peplow, "Coming of age — Twenty years after the ballyhooed discovery of graphene, the atom-thin carbon sheets are finding their footing," Science, Oct 2024: <https://www.science.org/content/article/twenty-years-after-its-discovery-graphene-finally-living-hype>
- [4] Wikipedia: Solyndra: <https://en.wikipedia.org/wiki/Solyndra>
- [5] The Solhyd project — Discover all about hydrogen panels: <https://solhyd.eu/en/>
- [6] Anthony King, "Northvolt bankruptcy dents European battery industry ambitions," ChemistryWorld, Nov 2024: <https://www.chemistryworld.com/news/northvolt-bankruptcy-dents-european-battery-industry-ambitions/4020603.article>
- [7] YouTube: Sabine Hossenfelder — Why I'm embarrassed to be German: <https://www.youtube.com/watch?v=W1ZZ-Yni8Fg>
- [8] YouTube: Wise Guys — Deutsche Bahn: <https://www.youtube.com/watch?v=wXjhszy2f9w>
- [9] Wikipedia: Why Most Published Research Findings Are False — John Ioannidis (2005): https://en.wikipedia.org/wiki/Why_Most_Published_Research_Findings_Are_False
- [10] YouTube: Sabine Hossenfelder — I was asked to keep this confidential: <https://www.youtube.com/watch?v=shFUDPqVmTg>



Cybersecurity

Tough Times for Hackers

By Gerhard Stelzer (Würth Elektronik eiSos)

Cybersecurity is shifting into the focus of European regulation. According to the EU Radio Equipment Directive, by August 2025, products with radio technology and internet access must protect networks and personal data as well as prevent fraud. The EN18031 standard is designed to assist manufacturers in implementation.

The EU Cyber Resilience Act (CRA) introduces entirely new cybersecurity requirements for electronic devices. It was published in the Official Journal of the EU in November 2024 and is therefore applicable law. However, no standards exist for it yet. A transitional period is therefore expected to apply until December 2027, after which new products will have to comply with the CRA.

Until then, the Radio Equipment Directive (RED) is intended to enhance cybersecurity, at least for all wireless devices. To this end, the EU amended the RED in 2022, supplementing Article 3(3), to define the following three sub-points:

- 3(3)(d) to ensure network protection
- 3(3)(e) to safeguard personal data and privacy
- 3(3)(f) to ensure protection against fraud

The deadline for implementation now runs until August 1, 2025. After that, no wireless devices may be placed on the EU market unless they meet the cybersecurity requirements of the RED.

EN 18031 Standard Facilitates Implementation

The EN 18031 standard translates the cybersecurity aspects of the RED into specific testing procedures and was developed by the CENELEC standards organization on behalf of the EU. In October 2024, CENELEC submitted EN 18031 to the European Commission for review. The standard was harmonized in January 2025, meaning that new products no longer have to be certified by an accredited testing service provider (**Figure 1**). A self-declaration of conformity is sufficient.

Conformity Test, Using a Wi-Fi Module as an Example

The conformity test is illustrated here, taking the example of Würth Elektronik's IoT Wi-Fi module Cordelia-I [1]. The Wi-Fi module is fully compliant with the RED, including the cybersecurity extension 2022/30/EU from January 2022. The Cordelia-I module complies with IEEE 802.11 b/g/n, operates in the 2.4 GHz band, and has the following features:

- Zero-touch provisioning with QuarkLink [2] (more on this later)
- Secure UART-to-cloud bridge (transparent mode)
- Small form factor: 19×27.5×4 mm³
- Sleep mode <10 µA
- Intelligent antenna configuration (2-in-1 module)
- Output power +18 dBm peak (1DSSS)
- Receiver sensitivity: -92 dBm (1 DSSS, 8% PER)
- Industrial temperature range: -40°C up to +85°C

According to the RED, cybersecurity for wireless devices such as radio modules must be ensured in accordance with Article 3(3). For the Cordelia-I radio module, only Article 3(3)(d) — protection of the network — is relevant, as neither personal data under 3(3)(e) nor financial data under 3(3)(f) is processed. It follows that Cordelia-I must be tested in accordance with EN 18031-1:2024.

The core requirements of EN 18031-1:2024 for a wireless device in terms of cybersecurity are:

- > authentication and authorization
- > data protection
- > software and firmware security
- > network security
- > physical security
- > incident response and management
- > user awareness and education
- > compliance with regulations and documentation

The cybersecurity space is volatile and constantly evolving, for example, with the CRA coming into effect at the end of 2027. It is important to conduct a cybersecurity risk analysis at an early stage of the design process and to continue this analysis throughout the product lifecycle. The implementation of cybersecurity involves more than just hardware and software. “Policies” also play an important role — for example, a vulnerability reporting system, software updates, ease of installation, maintenance, etc. The goal must be “security by design.”

Design of a Security Concept for an Embedded Device

When designing a security concept for an embedded device, several key questions have to be asked and answered: Which asset needs protection? What does the threat look like? Which security objective do we have? What do we need to achieve it? **Figure 2** illustrates these security requirements for different assets.

The current state of cybersecurity measures in a typical IoT application encompasses a range of security elements:

- > A secure “**root of trust**” is a unique identity along with a cryptographic key associated with it. As the entire security chain relies on this foundation of trust, it is essential that a RoT is unique, unchangeable, and unclonable.
- > **Secure boot** is a security feature that ensures that only trusted software is loaded during a device’s boot process. It is a firmware-based process that verifies the digital signature of the bootloader and all subsequent operating system components before they are loaded into memory. The goal of secure booting is to prevent the execution of malware and other malicious software and to compromise the system during the boot process.
- > **Firmware over-the-air (FOTA)** is a method by which the firmware of an electronic device is updated wirelessly, usually over a secure connection. FOTA updates can be used to fix bugs, add new features, or patch security gaps in the device’s firmware.

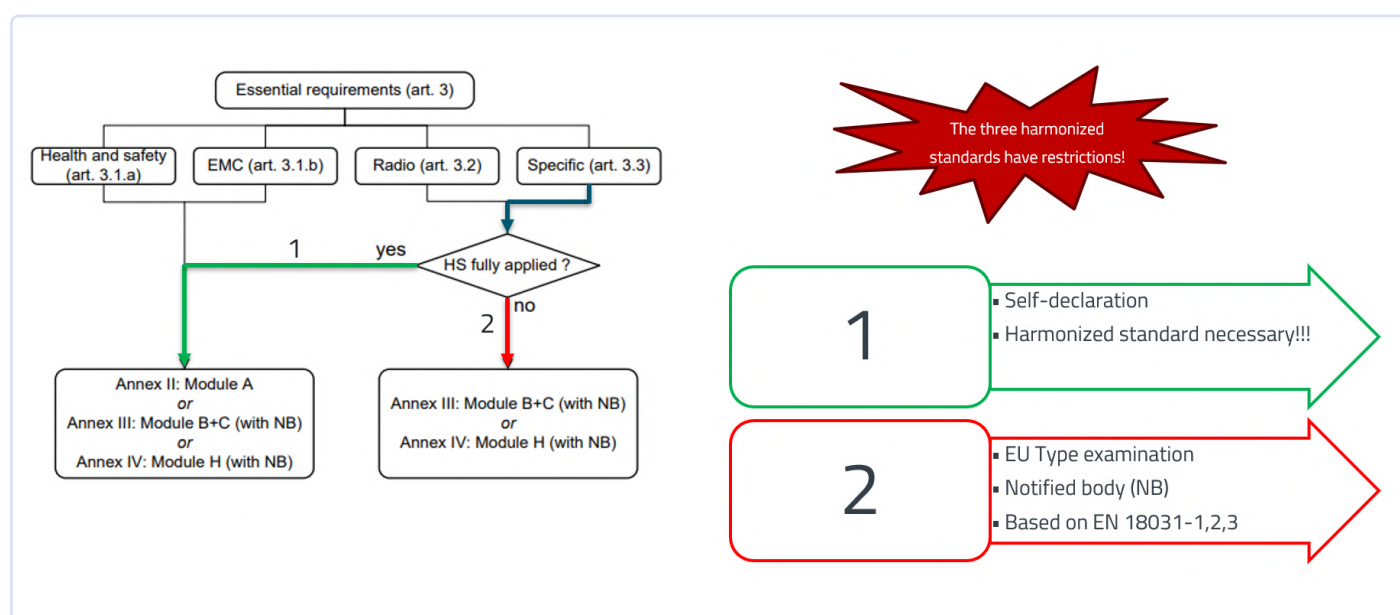


Figure 1: Whether a product’s conformity can be assessed in-house depends on the existence of a harmonized standard. Otherwise, certified service providers (Notified Bodies) come into play. (Source: European Commission — RED Guide [4])

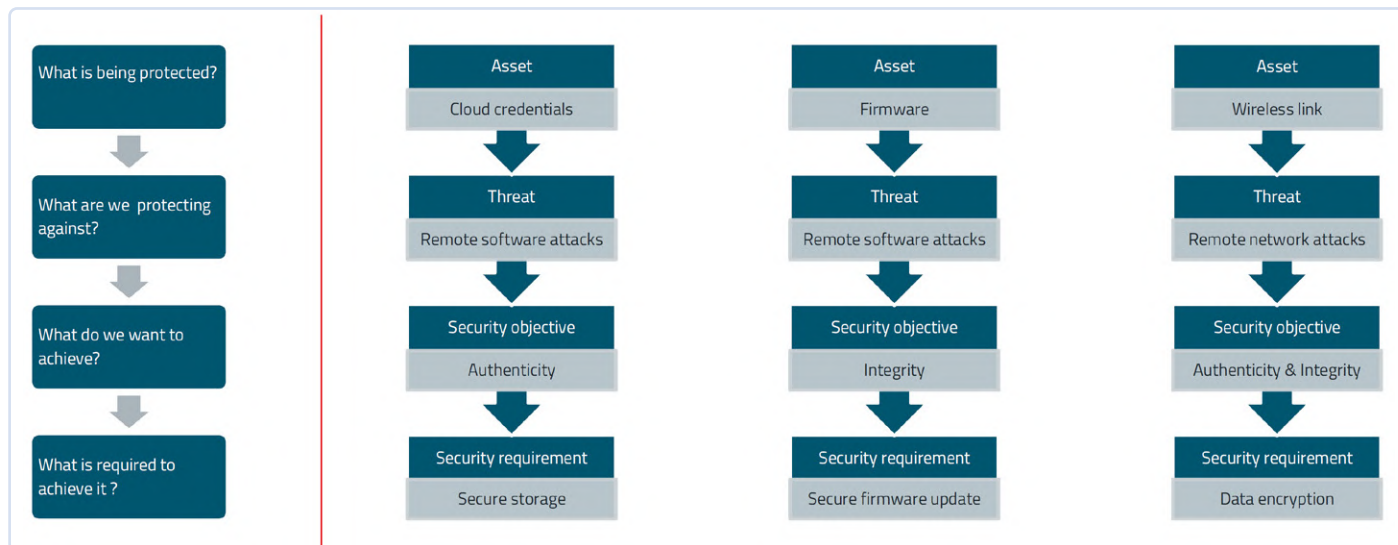


Figure 2: Design of a security concept for an embedded device with various assets to be protected. (Source: Würth Elektronik)

- Every interface to the outside world must be considered vulnerable and secured accordingly. A range of standard protocols can be used for **authentication and encryption of communication interfaces**.
- **Secure storage** areas are isolated execution environments that provide hardware-based security for sensitive data and code. They are designed to protect against attacks that attempt to access or modify data in memory or steal cryptographic keys.

These general measures to ensure cybersecurity have been implemented in the Cordelia-I Wi-Fi module (**Figure 3**). The Wi-Fi radio module includes the following security features:

- 10-byte non-manipulable unique device ID
- Secure boot
- Secure storage with an encrypted file system for storing certificates and other credentials
- Secure firmware over-the-air update
- Secure socket — Transport Layer Security protocol TLSv1.2
- Secure Wi-Fi connection according to WPA3
- Hardware-accelerated crypto engine

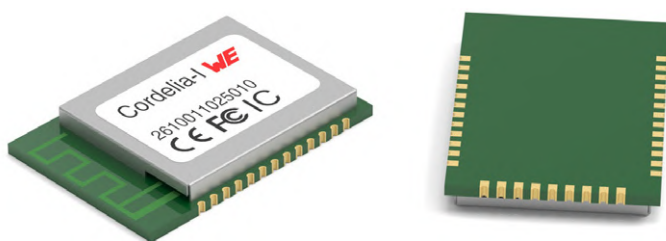


Figure 3: The Cordelia-I Wi-Fi module from Würth Elektronik offers a wide range of features to ensure cybersecurity in IoT applications.

Secure Cloud Connection

To ensure a secure connection to the cloud, the device and the cloud should perform mutual authentication, followed by exchanging a session key, which is then used to encrypt the communication channel. This is usually done using the TLS protocol.

To establish a mutual TLS connection, the following cryptographic elements must be present on the device and in the cloud. The assets must also be properly aligned with each other.

On the device side, these are:

- **Device ID:** Unique device identity that is unchangeable and tamper-proof.
- **Device key:** This is a private key that is unique to each device and must be kept secret.
- **Device certificate:** This certificate contains the public key associated with the device private key.
- **Root CA:** This is the root CA (Certificate Authority) of the cloud, which is used to authenticate the cloud endpoint.

The process of storing these parameters on the end device is known as “device provisioning”.

On the cloud side, this entails:

- **List of the device IDs:** A whitelist of device IDs that are authorized to establish connections.
- **Device certificates:** Public keys corresponding to the device IDs to enable device authentication.

The process of storing these parameters on the cloud endpoint is known as “cloud onboarding.” These cryptographic values must

be stored securely on both the device and the cloud endpoint. Disclosure of these values at any stage of the device manufacturing lifecycle can compromise security. Human interaction with these cryptographic assets often poses the greatest threat. Therefore, the steps must be followed to ensure maximum security.

The Cordelia-I module with the QuarkLink platform enables a secure cloud connection through zero-touch device provisioning and secure cloud onboarding (Figure 4). In addition, full device management — including remote cloud migration — can be carried out throughout the lifecycle of a device [3].

250415-01

About the Author

Gerhard Stelzer studied electrical engineering and information technology at the Technical University of Munich, where he graduated with a degree in engineering. Afterwards, he worked in the development of high-rate optical communication technology at Siemens AG. In 1995, he moved into technical journalism at "Elektronik" magazine. Since 2021, he has served as Senior Technical Editor at Würth Elektronik eiSos.

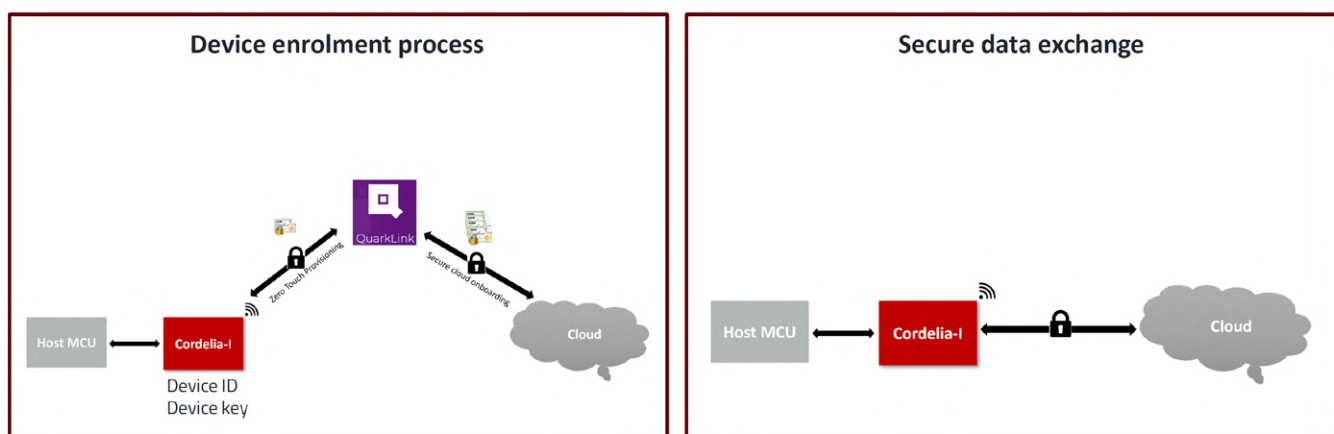


Figure 4: Enrolment to a cloud via zero-touch provisioning is handled by the QuarkLink IoT security platform from Crypto Quantique. Secure data exchange then takes place directly between the IoT device and the cloud. (Source: Würth Elektronik)

WEB LINKS

- [1] Cordelia-I Wi-Fi IoT radio module from Würth Elektronik: <https://www.we-online.com/de/components/products/CORDELIA-I>
- [2] QuarkLink, Crypto Quantique: <https://www.cryptoquantique.com/products/quarklink/>
- [3] Product Guide: Wireless Connectivity & Sensors from Würth Elektronik: <https://www.we-online.com/de/components/products/wco>
- [4] European Commission: Guide to the Radio Equipment Directive 2012/53/EU: <https://ec.europa.eu/docsroom/documents/33162>

Siglent Presents Next-Gen Multi-Channel Oscilloscopes

High-Performance Solutions for Modern Power and Embedded Systems

By Siglent Technologies Germany GmbH

The accelerating transformation across industries like automotive, renewable energy, and industrial automation is driven by the rise of advanced power electronics.

High-efficiency inverters, the switch to wide-bandgap semiconductors, and tightly integrated control loops are pushing the limits of traditional test tools. As systems become more compact and capable, engineers must observe and correlate more signals across multiple domains with greater precision. Legacy oscilloscopes, designed for simpler tasks, often fall short. Today's development workflows demand multi-channel instruments that offer deep insight, high signal fidelity, and application-focused analysis to meet the growing complexity of next-generation power and control systems.

Siglent [1] has introduced the latest addition to its high-performance 12-bit oscilloscope portfolio: the SDS5000X HD and SDS5000L series. These new oscilloscopes offer up to eight channels, a bandwidth up to 1 GHz, and mark a significant advancement in Siglent's multi-channel platform strategy. With high-speed sampling and deep memory architecture, the instruments are built to deliver reliable insight into complex signal environments. Combined with a comprehensive suite of analysis tools, the SDS5000X HD



Figure 1: The SDS5000X HD models offer a large, intuitive touchscreen, perfect for bench work and interactive debugging.

and SDS5000L are designed to meet the evolving demands of today's power electronics, embedded systems, and automated test applications.

Scalable and Flexible Solutions

The SDS5000X HD models (**Figure 1**) feature a large, intuitive, usable touchscreen interface ideal for bench work and interactive debugging, while the SDS5000L rack-mount variants (**Figure 2**) are optimized for integration into automated test systems, offering full remote control and space-saving design without a built-in display.

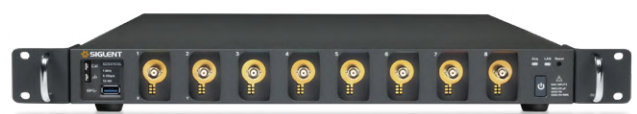


Figure 2: The SDS5000L rack-mount variants are ideal for automated test systems, with full remote control and a space-saving, display-free design.



Figure 3: The SDS5000X HD and SDS5000L series deliver the performance needed to tackle demanding measurement challenges.

These offerings give engineers the flexibility to choose the right tool for their workflow, whether it's in a hands-on R&D lab or a fully automated validation setup. The goal is to provide powerful, scalable solutions that grow with the complexity of modern test environments.

Advanced Performance for High Requirements

The SDS5000X HD and SDS5000L series deliver the performance needed to tackle demanding measurement challenges (**Figure 3**). With sampling rates of up to 5 GSamples/s, engineers can capture fast signal transitions with high precision. Each channel supports up to 500 Megapoints of memory (with all channels active), enabling long-duration acquisitions at high resolution, which is ideal for analyzing protocol sequences. In quarter-channel mode, memory depth extends up to 2.5 Gigapoints per channel, supporting deep signal analysis over extended time spans. An effective number of bits (ENOB) of up to 8.2 at 1 GHz ensures excellent dynamic range and signal accuracy, even at full bandwidth. Combined with an impressively low noise floor of just $140 \mu\text{V}_{\text{rms}}$ at 1 GHz, the new models offer clear, detailed signal visibility, which is crucial for accurate debugging in sensitive analog, power, and mixed-signal environments.

Multi-Channel Application-Focused Analysis

An 8-channel oscilloscope enables comprehensive three-phase power analysis by allowing simultaneous connection to all voltage and current signals. This synchronized acquisition ensures accurate, real-time measurement of waveforms and key parameters across all phases. Engineers can directly observe and compare phase relationships to detect imbalances and verify system performance. With built-in FFT capabilities, the oscilloscope also supports detailed harmonic analysis of three-phase systems. Optional application software further extends functionality by providing vector diagram visualization for motor diagnostics, power quality evaluation, ripple measurement, and overall system efficiency analysis.

Beyond power analysis, modern electronic systems increasingly rely on multiple integrated chips and modules, whose initialization and operation must be carefully coordinated to avoid undefined states. Power-up sequence testing verifies that signals from different circuit modules follow the correct timing, ensuring system stability and

preventing issues such as data transmission errors in communication systems. As circuit complexity grows, so do the challenges of power-up testing. The SDS5000X HD addresses this by capturing the entire power-up process of all relevant signals in a single acquisition. This significantly reduces measurement time, improving efficiency, and minimizing errors associated with repeated tests. For complex designs featuring up to eight power rails, this one-shot measurement capability is a powerful tool for reliable validation.

Accurate Measurements for SiC and GaN Power Electronics

With the introduction of Siglent's latest 8-channel oscilloscopes and the new optically isolated differential probe series ODP6000B, a crucial gap in wide-bandgap (WBG) semiconductor testing is now filled. The SDS5000X HD delivers picosecond-level rise time performance, enabling precise capture of the fast-switching behavior characteristic of silicon carbide (SiC) and gallium nitride (GaN) devices. It provides detailed analysis of voltage and current transients during switching events, including overshoot, ringing, and other dynamic effects. These capabilities help engineers optimize circuit performance and ensure signal integrity in high-efficiency power designs. Complementing these capabilities, the ODP6000B probes are available with bandwidths of 500 MHz and 1 GHz and feature an exceptional common-mode rejection ratio (CMRR) of 160 dB at low frequencies, further enhancing the accuracy and noise immunity essential for advanced WBG device measurements.

Elevating Efficiency, Accuracy, and Reliability

With the launch of the SDS5000X HD and SDS5000L series, along with the optically isolated ODP6000B probes, Siglent delivers a powerful and scalable measurement solution tailored to the demands of modern power electronics and complex embedded systems. The combination of high channel count, excellent signal fidelity, deep memory, and versatile analysis capabilities enables precise insights into demanding applications — from three-phase power analysis and power-up sequence testing to high-speed characterization of wide-bandgap semiconductors. Whether in the lab or integrated into automated test setups, Siglent provides the tools to elevate efficiency, accuracy, and reliability in development workflows. ◀

250428-01

WEB LINK

[1] Siglent Technologies Germany GmbH: <https://www.siglenteu.com/>



Bluetooth 6.0 Brings Enhanced Distance- Ranging Applications

Source: Adobe Stock / Anna generated with AI

New Version Offers Improved Device Positioning and Location Services

By Steven Keeping (Mouser Electronics)

New enhancements to the Bluetooth protocol, introduced in version 6.0, are expected to spur engineers to develop improved device-positioning and location service products. The most significant new feature is Bluetooth Channel Sounding, which significantly enhances the precision of all previous Bluetooth distance-measuring techniques.

The Bluetooth Special Interest Group (SIG) continues to diversify the target applications of the once consumer-focused short-range wireless technology. Originally designed to wirelessly connect peripherals to host computers, **Bluetooth®** technology has become almost ubiquitous as the go-to solution for power-frugal and reliable wireless connectivity, with almost universal smartphone support for the tech hardly harmed its adoption either.

Bluetooth technology use has expanded into healthcare, consumer, audio, industrial, and numerous other applications. Despite impressive uptake — Bluetooth SIG forecasts approximately 5.9 billion annual device shipments during 2025 — success in some target applications has been underwhelming [1]. One area of note is device positioning and location services.

New enhancements to the Bluetooth protocol, introduced in version 6.0, are expected to spur engineers to use the tech to develop improved device-position-

ing and location service products: The Bluetooth SIG forecasts a 22 percent CAGR to reach 563 million location services device shipments by 2028 [2].

Initial Support for Bluetooth Distance-Ranging

Theoretically, wireless connectivity can be used to locate lost objects. Think of those keys hiding in the depths of the sofa. If the keys are attached to a wireless tag, your smartphone can quickly indicate where to start looking. In an industrial environment, wireless mesh networks can help workers quickly locate goods on warehouse shelves. But in practice, implementing such applications is incredibly tricky. Factors such as multipath fading and interference from other nearby 2.4 GHz transceivers spoil the signal integrity and undermine location accuracy.

The Bluetooth SIG has already made several attempts to add distance ranging and device positioning support to its RF protocol. First was an attempt to use the transmit (TX)

parameter, which provided a reference power level one meter from the transmitting device, to provide a Received Signal Strength Indicator (RSSI). Because the strength of the RF signal tails off inversely proportional to the distance between the two radio devices, it can be used to provide a crude estimate of distance. This technique is useful for locating objects in reasonable proximity, but indicating the direction of one object relative to another is quite a different matter.

In 2019, Bluetooth Direction Finding [3] (adopted as part of the ratification of Bluetooth v5.1) enhanced Bluetooth device positioning and distance ranging. The technique enables applications to calculate the direction of a received signal using phase measurements made by the Bluetooth Low Energy controller. Two methods were defined, angle of arrival (AoA) and angle of departure (AoD).

While Bluetooth Direction Finding works well, it requires extensive design experience to implement. Designs are complex and can be expensive. As a result, the technology has been limited to high-end applications such as asset tracking high-value items, rather than being widely adopted for more day-to-day applications.

New to Bluetooth 6.0

The recent introduction of Bluetooth 6.0 brought several key improvements to the protocol, including:

- **decision-based advertising filtering**, a new functionality that enables a scanning device to use the information received from a primary advertising channel to determine whether there's a reason to look for related packets on the secondary channels;
- **monitoring advertisers**, which allows the Bluetooth Low Energy controller to filter duplicate advertising packets as instructed by the host component of an observer device;
- **enhancement to the Isochronous Adaptation Layer (ISOAL)**, which enables larger data frames to be transmitted using more compact link layer packets;
- **an extended link layer feature set**, increasing the number of link layer functions about which devices can exchange information; and
- **an update to frame spacing**, enabling flexible adjacent connection spacing used with connection event packets or connected isochronous stream events.

The most significant enhancement to Bluetooth 6.0 is the introduction of Bluetooth Channel Sounding, which significantly enhances the precision of all previous Bluetooth distance-measuring techniques. The updated Bluetooth specification defines new features of the radio (PHY), features of the controller, security measures, and procedures needed to collect raw measurement data.

New Techniques Simplify Device Positioning

Channel Sounding brings two simple and reliable solutions for distance-ranging applications: Phase-Based Ranging (PBR) and Round-Trip Timing (RTT). Both are standardized and interoperable and can be supported by very simple devices or as

an addition to a more advanced product without extra hardware costs and with minimal software additions.

PBR uses the phase shift of a signal sent by the initiator device and returned by the reflector device across multiple frequencies. The data-to-distance conversion uses dedicated algorithms and is performed at the application level. The RTT technique is based on the time it takes for radio packets to travel back and forth between the initiator and the reflector. RTT acts as a secure distance bounding technique to cross-check PBR, and the algorithms used to calculate the distance are simpler than those used for PBR.

For both PBR and RTT, power consumption is generally as low as regular data transfer over Bluetooth Low Energy. Finally, Channel Sounding supports various hardware and software configuration options for accuracy, latency, security, and power consumption.

Inspiring New Applications

Channel Sounding will make it easier for a new cohort of developers to come up with device positioning and distance-ranging applications, including the following possible examples:

- **Asset tracking:** Channel Sounding promises to bring greater precision, flexibility, reliability, and convenience to distance ranging and locationing without adding significant design complexity and cost.
- **Smart locks:** Security will improve because the position of the person wanting to use the lock will be known more accurately. Smart locks will also benefit from robust protection against man-in-the-middle and relay attacks.
- **Tracking tags:** Contemporary tag

solutions work well, but the vibration or sound they emit when activated can be muted by things like sofa cushions or blankets. Channel Sounding will enable precise proximity alerts over long distances to overcome the limitations of sound or vibration alarms.

- **Appliances:** The physical context information provided by Channel Sounding will be useful when working with multiple devices and will assist safety features that activate control functions only when the user is near the device.

Lucrative Returns

Bluetooth Low Energy continues to diversify into new consumer, healthcare, industrial automation, and smart home applications among many others. After a slow start, the locationing sector promises lucrative returns. The Channel Sounding technology added to Bluetooth 6.0 opens a huge range of possible distance-ranging and locationing applications, including enhanced asset-tracking devices, smart locks, tags, and appliances. ◀

250416-01

About the Author

Steven Keeping gained a BEng (Hons.) degree at Brighton University, U.K., before working in the electronics divisions of Eurotherm and BOC for seven years. He then joined Electronic Production magazine and subsequently spent 13 years in senior editorial and publishing roles on electronics manufacturing, test, and design titles, including *What's New in Electronics* and Australian Electronics Engineering for Trinity Mirror, CMP, and RBI in the U.K. and Australia. In 2006, Steven became a freelance journalist specializing in electronics. He is based in Sydney.

WEB LINKS

- [1] "2024 Bluetooth® Market Update," bluetooth.com, 2024: <https://www.bluetooth.com/2024-market-update/>
- [2] "2024 Bluetooth® Market Update: Location Services," bluetooth.com, 2024: <https://www.bluetooth.com/2024-market-update/#location-services>
- [3] Steven Keeping, "Bluetooth Heads in New Direction," Mouser: <https://resources.mouser.com/explore-all/bluetooth-heads-in-new-direction>

Exploring Wireless Communication with BeagleY-AI

By Dogan Ibrahim (United Kingdom)

In this article, we explore two hands-on projects that demonstrate how devices can communicate wirelessly. First, we'll set up a TCP connection between a BeagleY-AI board and an Android smartphone to send text messages. Then, we'll explore controlling an LED connected to BeagleY-AI from a smartphone using UDP. These projects will give you a practical look at how wireless communication works in everyday tech!

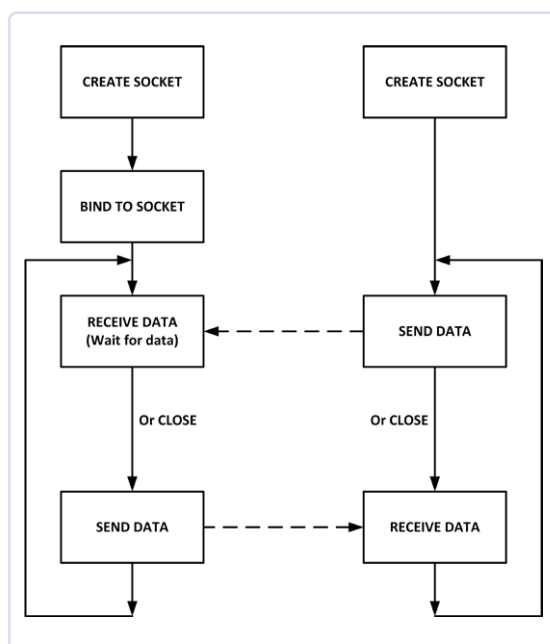
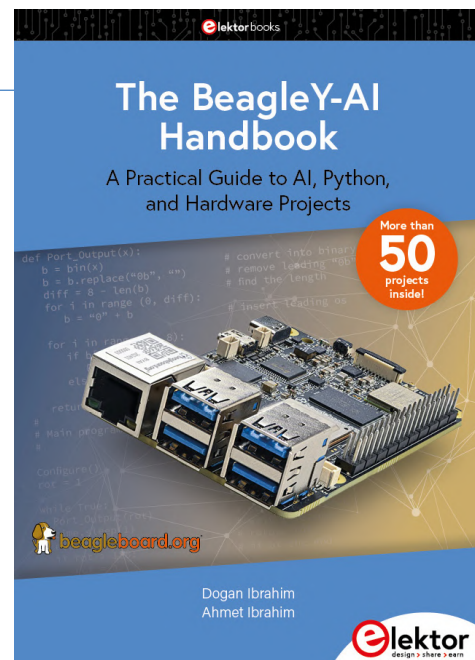


Figure 1:
UDP communication.



Editor's Note. This article is an excerpt from The BeagleY-AI Handbook. It was formatted and lightly edited to match ElektorMag's conventions and page layout. The author and editor are happy to help with queries. Contact details are in the **Questions or Comments?** box.

Three major features of BeagleY-AI are its Wi-Fi, Bluetooth communication, and AI capabilities. BeagleY-AI is equipped with a dual-band 2.4 GHz IEEE802.11ax wireless LAN module and Bluetooth Low Energy (BLE) 5.4. These built-in features mean you won't need any external network-based hardware to communicate over the Internet. Network communication is handled using either UDP or TCP protocols. In this article, we'll guide you through two projects that demonstrate how to use these protocols!

TCP and UDP

TCP is a connection-based protocol that guarantees the delivery of packets. Packets are assigned sequence numbers, and the receipt of all packets is acknowledged to ensure they don't arrive out of order. Due to this confirmation process, TCP tends to be slower but is more reliable, ensuring packet delivery. UDP, in contrast, is not connection-based. It doesn't use sequence numbers, so there is no guarantee that packets will arrive in the correct order or at all. However, UDP has less overhead compared to TCP, making it faster.

UDP Communication

Let's break down the UDP communication process between the server and client over a Wi-Fi link (this process is illustrated in **Figure 1**):



Server

1. Create a UDP socket.
2. Bind the socket to the server address.
3. Wait until the datagram packet arrives from the client.
4. Process the datagram packet.
5. Send a reply to the client, or close the socket.
6. Go back to step 3 (if not closed).

Client

1. Create a UDP socket (and optionally bind).
2. Send a message to the server.
3. Wait until the response from the server is received.
4. Process the reply.
5. Go back to step 2, or close the socket.

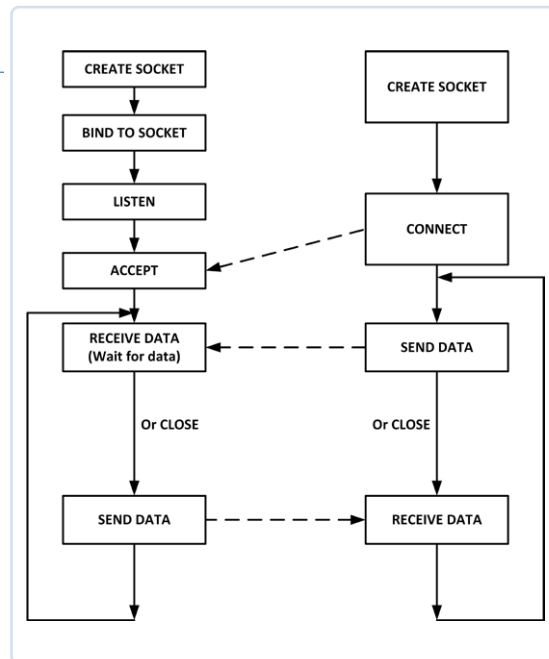


Figure 2:
TCP communication.

TCP Communication

Let's go over the steps for TCP communication between the server and client over a Wi-Fi link (this process is illustrated in **Figure 2**):

Server

1. Create a TCP socket.
2. Bind the socket to the server address.
3. Listen for connections.
4. Accept the connection.
5. Wait until the datagram packet arrives from the client.
6. Process the datagram packet.
7. Send a reply to the client or close the socket.
8. Go back to step 3 (if not closed).

Client

1. Create a TCP socket.
2. Connect to the server.
3. Send a message to the server.
4. Wait until the response from the server is received.
5. Process the reply.
6. Go back to step 2, or close the socket.

the Client node. **Figure 3** illustrates the project block diagram, showing how the BeagleY-AI and smartphone communicate over a Wi-Fi router.

Listing 1 shows the program (*tcpserver.py*). At the beginning of the program, a TCP/IP socket is created (`sock.SOCK_STREAM`) and bound to port 1500. The server then listens for a connection. While the server can accept connections from multiple clients, it can only communicate with one at a time. Once the client establishes a connection, it is accepted by the server. The server then reads a message from the keyboard and sends it to the client over the Wi-Fi link. Additionally, the `setsockopt()` statement ensures that the program can be reused without having to wait for the socket timeout of 30 s.

Testing

You should stop the firewall on your BeagleY-AI if it is running. Enter the following command to check the status of the firewall:

```
beagle@beagle:~ $ sudo ufw status
```

Project 1: Sending a Text Message to a Smartphone Using TCP

This project establishes a TCP/IP-based communication with an Android smartphone. The program reads text messages from the keyboard and sends them to the smartphone. The aim of this project is to show how TCP/IP communication can be established with an Android smartphone.

Port numbers range from 0 to 65535. Numbers from 0 to 1023 are reserved and are called well-known ports. For instance, port 23 is the Telnet port, and port 25 is the SMTP port. In this section, you will be working with port number 1500 in your program. BeagleY-AI acts as the Server node, while the smartphone serves as

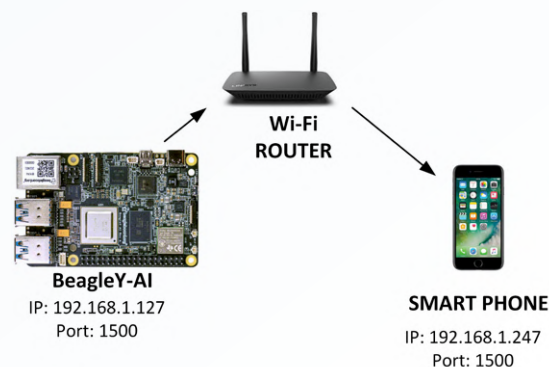


Figure 3: Block diagram
of Project 1.



Listing 1. tcpserver.py.

```
import socket

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
sock.bind(("192.168.1.127", 1500))
sock.listen(1)
client, addr = sock.accept() # accept connection
print("Connected to client: ", addr) # connected message

yn = 'y'
while yn == 'y':
    msg = input("Enter your message: ") # read a message
    msg = msg + "\n"
    client.send(msg.encode('utf-8')) # send the message
    yn = input("Send more messages?: ")
    yn = yn.lower()

print("\nClosing connection to client")
sock.close()
```

If the firewall is running, you should disable it using the following command:

```
beagle@beagle:~ $ sudo ufw disable
```

There are many TCP apps available free of charge on the Internet for smartphones. In this project, the TCP Client app by HARDCODED JOY S.R.L. is used on an Android smartphone. This app is available free of charge in the Play Store (see **Figure 4**).

The program is run as follows:

- First, run the server program:

```
beagle@beagle:~ $ python tcpserver.py
```

- Next, open the Android app and configure it as shown in **Figure 5** (click the 3-line settings icon at the top right-hand of the screen), where 192.168.1.127 is the IP address of the BeagleY-AI.
- Click **CONNECT** in the settings menu to connect to the BeagleY-AI over TCP/IP.

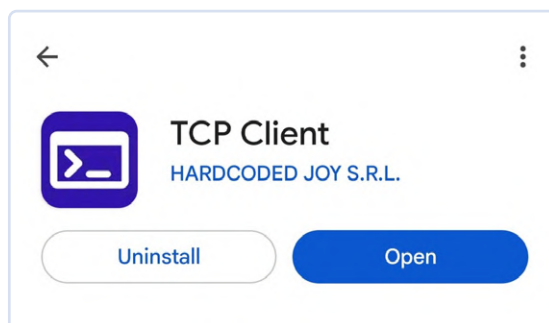
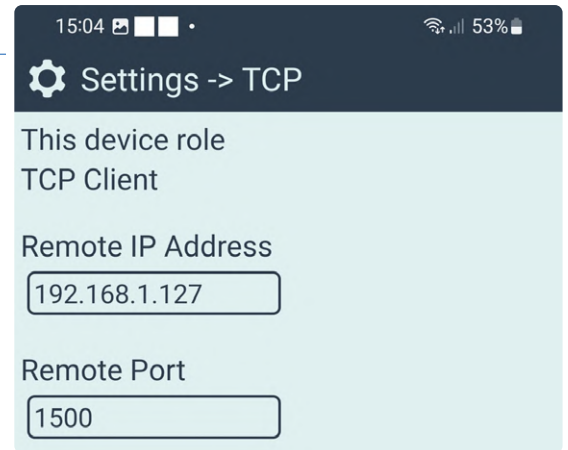


Figure 4:
Apps used in the project.



▲ Figure 5: Configuring the TCP Client App on the Smartphone.

- After connecting, you should see a connection message on your BeagleY-AI screen and also the IP address of the remote Android smartphone. Now, enter a message and press the *Enter* key. In this example, the message “HELLO FROM BEAGLEY-AI” is sent to the client (**Figure 6**). **Figure 7** shows the message displayed on the smartphone.

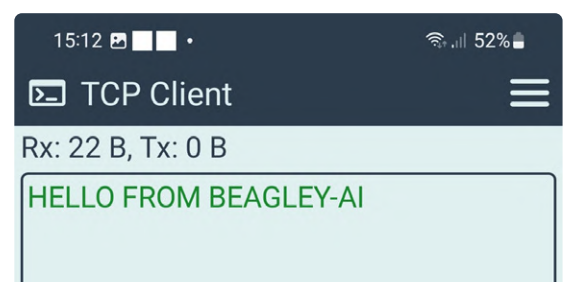
Project 2: Controlling an LED Connected to BeagleY-AI from a Smartphone Using UDP

In this project, an LED is connected to BeagleY-AI port pin GPIO21 (pin 40) through a 470-Ω current-limiting resistor. The LED is controlled by sending *On* and *Off* commands from an Android smartphone. The aim of this project is to show how an LED on BeagleY-AI can be controlled from a smartphone by sending commands using the UDP protocol over a Wi-Fi link. In this setup, BeagleY-AI acts as the server, while the smartphone serves as the client.

```
beagle@beagle:~$ python tcpserver.py
Connected to client: ('192.168.1.247', 39638)
Enter your message: HELLO FROM BEAGLEY-AI
Send more messages?: N

Closing connection to client
beagle@beagle:~$
```

▲ Figure 6: Enter the message on the keyboard.



▲ Figure 7: Message displayed on the smartphone.



Listing 2. udpled.py.

```

import socket
import gpod
from time import sleep

led = gpod.find_line('GPIO21')
led.request(consumer='beagle', type=gpod.LINE_REQ_DIR_OUT, default_val=0)
led.set_value(0)
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind(("192.168.1.127", 1500)) # Bind to Zero 2 W IP,port
data = [' '] * 10
while data != b'X':
    data, addr = sock.recvfrom(1024)
    if data == b'ON': # ON command
        led.set_value(1) # LED ON
    elif data == b'OFF': # OFF command
        led.set_value(0) # LED OFF

sock.close()
sleep(1)

```

The LED can easily be replaced with a relay, for example, to control electrical appliances from a smartphone.

Listing 2 (*udpled.py*) shows the program. A socket is created, and the server waits to receive commands from a client to control the LED. If the command is *On*, the LED is turned on. If, on the other hand, the command is *Off*, the LED is turned off. Command *X* terminates the server program.

The program can be tested using the UDP Sender/Receiver app shown in **Figure 8**.

The steps to test the program are as follows:

- > Build the circuit on BeagleY-AI with the LED.
- > Start the server program on BeagleY-AI:

```
beagle@beagle:~ $ python udpled.py
```

- > Start and configure the smartphone app.
- > Write the command *On* and press *Send* on the smartphone. The LED should turn on. Similarly, write *Off* and the LED should turn off. Sending *X* should terminate the BeagleY-AI program. ◀

250219-01

Questions or Comments?

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

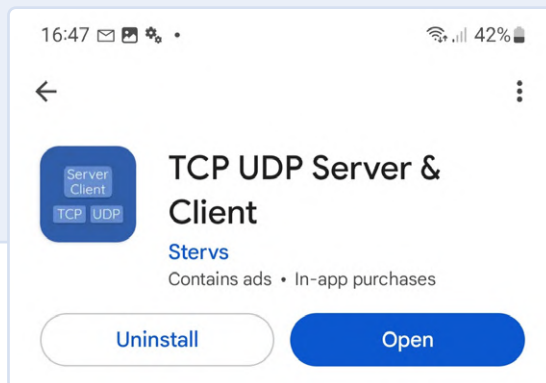


Figure 8: UDP app.



About the Author

Prof Dr Dogan Ibrahim has BSc degree in electronic engineering, an MSc degree in automatic control engineering, and a PhD degree in digital signal processing. Dogan has worked in many industrial organizations before he returned to academic life. Prof Ibrahim is the author of over 70 technical books and published over 200 technical articles on micro-controllers, microprocessors, and related fields. He is a Chartered electrical engineer and a Fellow of the Institution of the Engineering Technology. He is also a certified Arduino professional.



Related Products

- > Dogan Ibrahim, *The BeagleY-AI Handbook* (Elektor 2025)
www.elektor.com/21137
- > Dogan Ibrahim, *The BeagleY-AI Handbook* (E-Book, Elektor 2025)
www.elektor.com/21138

Err-electronics

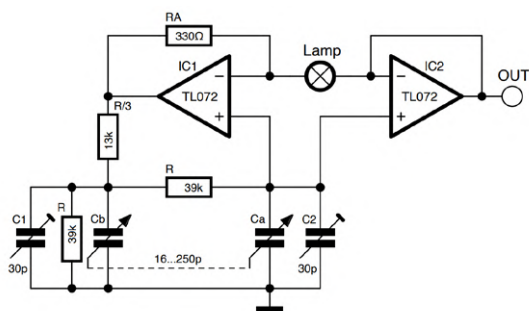
Corrections, Updates, and Readers' Letters

Compiled by Jean-François Simon (Elektor)

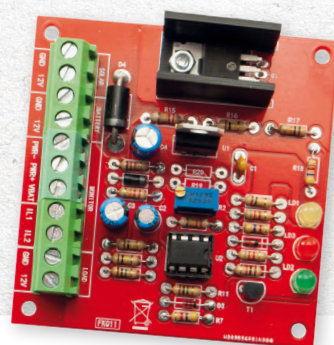
Nortonized Wien Bridge Oscillator Elektor 5-6/2025, p. 98 (240008)

We have missed a resistor in the schematic shown in Figure 7. For consistency with Figure 5, please exchange the capacitors C_a and C_b in Figure 7, and insert a 39-k Ω resistor in parallel with C_b . On both figures, C_b must be paralleled with a resistor. The corrected schematic is shown below.

On page 99, the sentence "This setup operates from 15 to 250 kHz, but adjusting I_{ABC} requires the amplitude to be in the range of a few tenths of millivolts to avoid distortion." should be corrected to "This setup operates from 15 to 250 kHz, but adjusting I_{ABC} requires the amplitude to be in the range of a few *tens* of millivolts to avoid distortion."



Source:
Elettronica In



Simple PV Power Regulator Elektor 1-2/2024, p. 68 (230586)

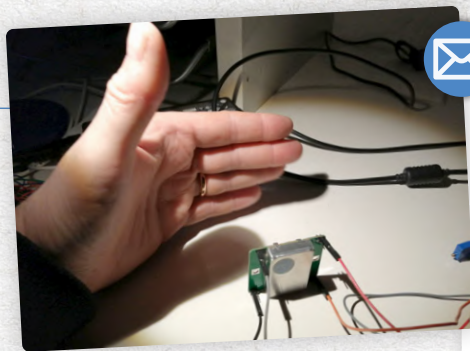
In the parts list for this controller (page 72), the values for R_{19} and R_{20} are not specified, although there is a note "see text." However, after reading through the text (which is otherwise clearly explained) several times, I cannot find any mention of R_{19} or R_{20} anywhere. I hope you can email me the missing values or the relevant part of the text, as I would very much like to build this circuit. Thank you in advance.

Gert Slijpen (The Netherlands)

Thank you for your email. We are sorry for this confusion in the article, which was probably introduced when we translated the article from Italian. In the Italian article, it is written: " R_{19} , R_{20} :" instead of "see text." From the photo, I can see that those two resistors were not used. I would guess that they were from a previous revision that the author did. The circuit should work fine without R_{19} and R_{20} .

Jean-François Simon (Elektor)

Ideas or Feedback?
Got a bright idea or valuable feedback
for Elektor? Reach out to us at
editor@elektor.com. We're eager to
hear from you!



The HB100 Doppler Motion Sensor

Elektor 7-8/2023, p. 90 (230205)

This article about using the HB100 Doppler sensor got me really interested. I bought two HB100 modules and two amplifiers, and I loaded the software onto an Arduino. However, the speed readings seem random. Sometimes it shows speeds over 100 km/h when I just move my hand in front of the HB100 module. Also, the detection range is very short (less than 20 cm). I swapped the two modules and the two amplifiers, and I adjusted the amplifier potentiometers. Still, the speed readings seem random. My goal is to build a speed sensor to check the speed of cars on my street. I'm not sure this sensor can measure speeds at a distance of 15 meters. Do you have any suggestions to get this project working properly?

Olivier Croiset (France)

If the speed readings seem random or detection range is very low, here are some possible causes and fixes:

- Ensure the HB100 signal is amplified and clean before feeding it into the Arduino. The HB100's raw signal is very weak (a few μV) and needs proper amplification.
- Try adding a Schmitt trigger (74HC14) to clean up the signal before feeding it into the Arduino.
- The HB100 has a limited range ($\sim 10\text{ m}$ max), but factors like angle and reflectivity affect performance.
- If the object moves laterally with respect to the source, detection is not ensured. The strongest signal is directly in front of the HB100's antenna. Ensure the object moves toward or away from the sensor, not sideways.
- The HB100 works best with metallic or large objects. Try using a metal plate or a large solid object. Detection is weaker for soft materials like fabric.
- If an object moves too slowly, the Doppler shift might be too low to detect.
- The HB100 is sensitive to power fluctuations, and noise from the Arduino's 5 V line can degrade performance. Use only a stable 5 V power source.

If the issue persists, check if the HB100 is producing a signal using an oscilloscope to verify the HB100's output. If you don't have an oscilloscope, try an audio amplifier and listen for a tone change when a moving object is near. Regarding the speed sensor project, I'm afraid this sensor is not the best choice. As said before, it has low range detection (less than 10 m). However, there might be a chance to improve its range by modifying its antenna and checking experimentally how far the transmitted signal can be received.

Stefano Lovati (Italy, Author of the article)



Triangular Wave Oscillator with Sine Wave Converter

Elektor 7-8/2011, p. 58 (110431)

I found this article quite interesting, but since I'm not equipped to quickly solder something together, I first looked into it using a simulation program. From the text, it appears that this circuit has been successfully used in practice; the use of a differential amplifier for sine shaping has also been seen elsewhere. Unfortunately, the simulation (I'm using LTSpice IV) doesn't show any sine-shaped voltage at all. The components used are all non-critical at the given frequency range, and the triangle oscillator behaves according to theory. So what am I doing wrong? I'm attaching the LTSpice schematic, the corresponding netlist and a screenshot in PDF. Thank you very much in advance if you find the time to take a quick look.

Günter Herth (Germany)

Thank you for your email and for the files. An interesting circuit indeed! It seems to me that the value of RP2 should be reduced. With a value around $150\ \Omega$ to $160\ \Omega$, the circuit appears to produce a roughly sinusoidal output signal. If you wish, you could try several values for RP2 using .step, and calculate the THD with .four. Have fun!

Jean-François Simon (Elektor)

Thank you very much. Your suggested values for RP2 indeed result in an approximately sinusoidal signal. I had also tried changing the resistor value, but only from $1\text{ k}\Omega$ to $5\text{ k}\Omega$, and saw no effect. Perhaps it was just an error in the schematic: $500\ \Omega$ would be more realistic than $5\text{ k}\Omega$. By the way, LTSpice's Fourier analysis is not sufficiently documented and should therefore be interpreted with caution.

Günter Herth (Germany)



From Life's Experience

Elektor 5-6/2025, p. 84 (250136)

Thank you for featuring a solar thermal collector! These days, photovoltaic systems tend to overshadow the existence of thermal solar collectors, even though these became widespread during the first oil crisis.

A 1-m² photovoltaic panel in summer yields about 200 W of electricity, while a thermal collector of the same area yields about 700 W of heat, and is cheaper. A set of two 2-m² thermal collectors can cover 50% of a household's hot water needs and pay for itself in five years. Meanwhile, a photovoltaic installation typically takes between 10 and 14 years to pay off. I wonder why so many people rush toward the less cost-effective option! Concerning wood, in my view, firewood and pellet-based systems are not truly ecological and are a major source of fine particle pollution. This is well-known in scientific circles, but rarely mentioned by wood vendors.

On the topic of energy self-sufficiency, it's important to remain cautious. The field is full of unrealistic promises, and many people are taken in by them. It's essential to request very detailed documentation of projected energy usage, and ideally have it reviewed by a specialist. That said, full energy autonomy is indeed possible, not just for tiny homes. The first known example is a fully solar-powered and autonomous house built in 1992 by the Fraunhofer Institute, Germany's largest research institute. It used seasonal hydrogen storage and is still in operation on their campus. A similar house was built in 2017 by Swedish engineer Olof Nilsson.

This kind of total solar autonomy (heating, hot water, and electricity) is technically achievable. The main barrier has been cost. At the time, such systems were prohibitively expensive, but photovoltaic prices have since dropped significantly. We are now waiting for hydrogen systems, including electrolyzers, fuel cells, and storage tanks, to become more affordable. At Onebus, we are not hydrogen specialists, but we focus on designing ultra-low-energy homes, which naturally align with such advanced systems, even if future technical solutions (like hydrogen) are still costly for now.

We are also working on components for autonomous homes, like converting standard electric water heaters into solar ones without drilling or welding, while retaining the full electric backup, and adding solar thermal collectors. We are also developing a hybrid solar/air collector made from photovoltaic cells that not only produce electricity but also recover the remaining 80% of solar energy as heat. Lastly, we are experimenting with building hot water solar collectors from off-the-shelf sheet metal, assembled using minimal tools, which almost anyone can build themselves.

*Yves Accard (France, Thermal engineer at www.onebus.fr)
Onebus is a non-profit organization. The advice provided is given on a voluntary basis.*



Source: Adobe Stock/Michal

Starting Out in Electronics...

...Concludes the Topic on Opamps

By Eric Bogers (Elektor)

Last time, we started tentatively with the equalizer. We will continue with that here, and then we will conclude the topic of opamps for the time being.

LC Filters

LC filter circuits have the great advantage of being able to manage with a minimal number of semiconductors. This means that noise and distortion can hardly cause any problems. The few equalizers that are built with “real” coils are, therefore, arts for audiophiles.

However, the required coils are a major issue. Because electromagnetic compatibility is a hot topic, fixed coils are available at relatively low prices, but their self-induction is usually far too low, and the ohmic resistance of the winding wire quickly becomes an issue. In many cases, we will, therefore, not be able to avoid winding coils manually — and that is not cheap. Such coils are also sensitive to magnetic radiation, so the device as a whole must be properly shielded. In short, the savings on semiconductors do not nearly outweigh the extra costs and effort that coils entail.

In the diagram in **Figure 1**, we have only drawn two filters, but this diagram can, in principle, be expanded with any number of filter sections. Resistors R1 and R2 are usually chosen to have the same value (for example, 10 kΩ); the value of the (slide) potentiometer is in the same order of magnitude.

The following formula can be used to calculate the resonance frequency:

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

The filter quality, or Q factor, is defined as the quotient of the intermediate frequency and bandwidth and is, therefore, the inverse

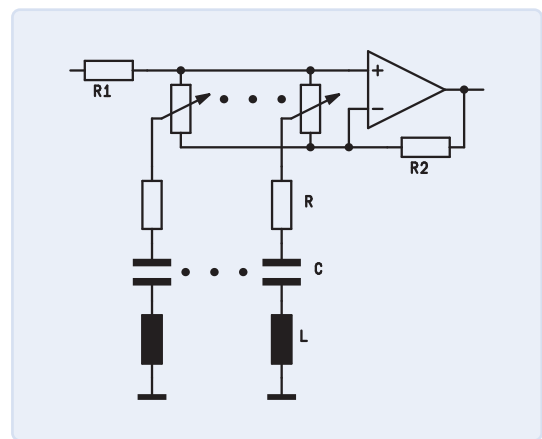


Figure 1: Equalizer with LC filters.

of the attenuation. The attenuation follows from the ratio of the resistance to the capacitance and self-induction. Here, the resistance consists of the series connection of the coil's wire resistance plus an ohmic resistance. This ohmic resistance is generally used to compensate for the unequal wire resistances of the individual inductances, and typically also to provide a useful attenuation value.

$$Q = \frac{f_M}{f_B} = \frac{1}{d}$$

$$d = \frac{R}{2 \cdot \pi \cdot f \cdot L} = \frac{R}{\sqrt{L/C}}$$

As an example, we will calculate such an equalizer filter (if you like to calculate and build a complete third-band equalizer, it is best to write a suitable computer program).

We assume that it must be a fairly narrowband filter to be able to affect the frequency characteristic without immediately creating a large gap. With a center frequency of 1 kHz, we therefore choose a bandwidth of 200 Hz.

From this data, we calculate the Q factor and the attenuation:

$$Q = \frac{f_M}{f_B} = \frac{1000 \text{ Hz}}{200 \text{ Hz}} = 5 \Rightarrow d = \frac{1}{Q} = 0.2$$

As for the resistance, we select a value of 100 Ω . That may seem rather high, but, because we want to replace the self-induction with a gyrator later on (see below), we cannot use very small values.

Now, we calculate the self-induction:

$$L = \frac{R}{2 \cdot \pi \cdot f \cdot d} = \frac{100 \Omega}{2 \cdot \pi \cdot 1000 \text{ Hz} \cdot 0.2} = 79.5 \text{ mH}$$

And then the capacity:

$$C = \frac{1}{4 \cdot \pi^2 \cdot f^2 \cdot L} = \frac{1}{4 \cdot \pi^2 \cdot (1 \text{ kHz})^2 \cdot 0.0795 \text{ H}} = 0.318 \mu\text{F}$$

By the way, when we search for a suitable fixed self-induction (which would be 68 mH) in the online catalog of a well-known mail order company, we see that availability is not a problem — but with such a coil, the ohmic resistance is already 780 Ω . The frequency in our example is only 1 kHz — we don't even want to think about what value we would end up with at 20 Hz.

One other thing: LC equalizers usually have a limit at 40 Hz, and there are reasons for that.

Gyrator

Fortunately, self-inductances can also be *simulated* with a so-called gyrator — see **Figure 2**.

In this case, R1 represents the coil's ohmic resistance (caused by capacitive losses — these are negligible in the LF range, so a value between 100 k Ω and 1 M Ω can be chosen for R1).

R2 stands for the wiring resistance of the coil: We can use 100 Ω for this (you can't really use a smaller value for the opamp — a value of 1 k Ω would be even better).

The simulated self-inductance of the gyrator now follows from:

$$L = R_1 \cdot R_2 \cdot C$$

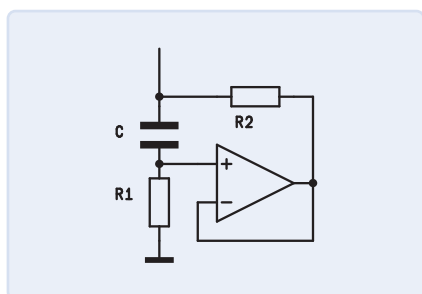


Figure 2: Gyrator.

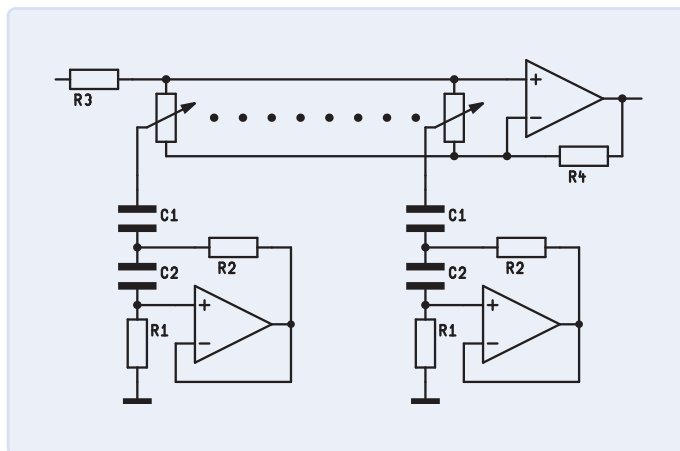


Figure 3: Equalizer with gyrators.

Then, we can calculate for our 1 kHz filter:

$$C = \frac{L}{R_1 \cdot R_2} = \frac{0.0795 \text{ H}}{220 \text{ k}\Omega \cdot 100 \Omega} = 3.6 \text{ nF}$$

In practice, using a gyrator circuit would increase the resistance to 1 k Ω ; the self-induction would then be 0.795 H and the capacity of the oscillating circuit 33 nF. However, this does not change the capacity of the gyrator capacitor.

Figure 3 shows what an equalizer with gyrators looks like. Of course, this does add another 31 operational amplifiers, but fortunately they add only cost and not noise (for the most part).

Operational Amplifiers in Practice

Except for SMD components, operational amplifiers are almost always assembled in DIL8 or DIL14 packages (**Figure 4**). DIL stands for *dual inline*, which means that there is a row of connection pins on either side of the package.

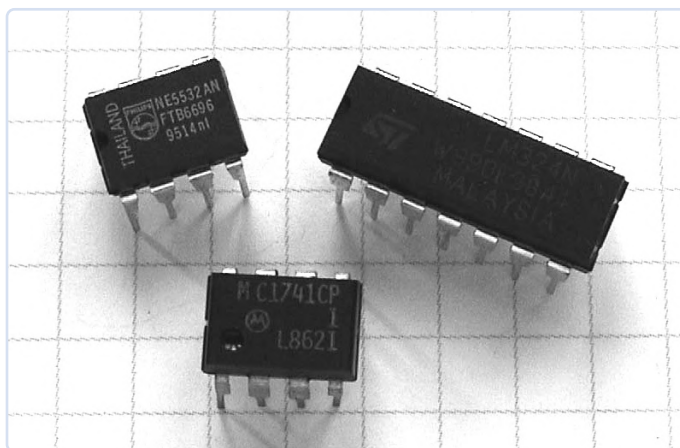


Figure 4: Some operational amplifiers.

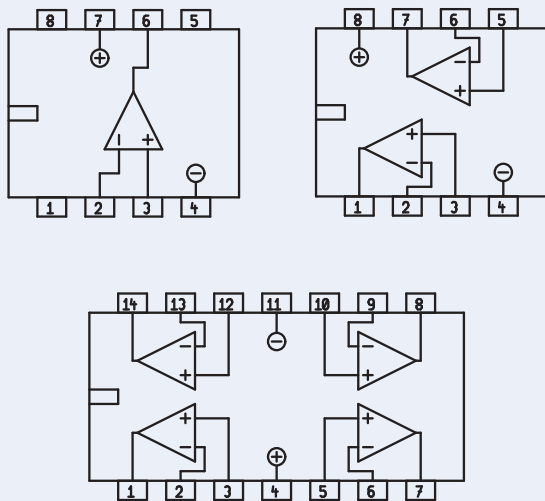


Figure 5: Pin layout of (most) opamps.

Single and dual opamps are enclosed in DIL8 packages, while quad opamps are enclosed in DIL14 packages. The pinout of these ICs is almost always as outlined in **Figure 5**.

With single opamps, pins 1, 5, and 8 are often used for frequency and offset compensation. More information about this can be found in the opamps' datasheets.

Power Supply

In the diagrams that we have shown in the last few installations, we have not included the power supply for the opamps, but that does not mean that opamps do not require one.

To guarantee optimal functioning of the opamp, decoupling capacitors of about 10 to 100 nF should be placed as close as possible to

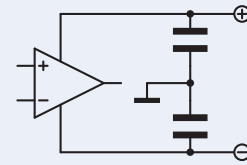



Figure 6: Decoupling the supply voltage.

the IC's power connections (**Figure 6**). If the circuit is to meet the highest standards, you can use a parallel connection of an electrolytic capacitor of approximately 10 μ F and a ceramic capacitor of approximately 1 nF. 

Translated by Hans Adams — 250271-01

Editor's note: This 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.

Questions or Comments?

Do you have technical questions or comments about his article? Send an email to Elektor at editor@elektor.com.



Related Product

- **B. Kainka, Basic Electronics for Beginners (Elektor, 2020)**
Book: www.elektor.com/19212
Ebook: www.elektor.com/19213

THE COAX FAMILY HAS EXPANDED



**WURTH
ELEKTRONIK**
MORE THAN
YOU EXPECT

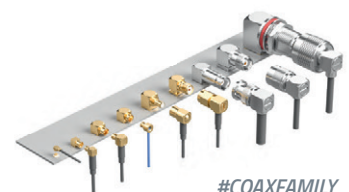
New coaxial products

Coaxial connectors and cable assemblies provide low loss paths between RF system, subassemblies or components. The coaxial product range includes frequency supporting up to 18 GHz and with various design.

www.we-online.com/coax

Highlights

- MIL-STD & IEC interfaces
- Characteristic impedance 50 Ω
- High quality data transmission
- Broad line of product portfolio with customize capability



#COAXFAMILY

A Powerful AI Code Assistant

Speed Up Your Development with Continue and Visual Studio Code

By Koen Vervloesem (Belgium)

Large language models can accelerate your software development. In this tutorial, we use the Continue extension for Visual Studio Code and enjoy the advantages of an AI code assistant that autocompletes, rewrites, and explains code.

Large language models (LLM) like OpenAI's ChatGPT [1] are often referred to as "autocomplete on steroids" for a variety of programming languages, but they offer much more. While these neural networks essentially predict and complete the text you type, training them on extensive data sets of texts and conversations has resulted in unexpected emergent capabilities. You can even run the LLM entirely on your own hardware, ensuring that you don't send confidential information to the cloud.

One remarkable capability is their proficiency in coding tasks. You can instruct an LLM to draft a shell script for a one-off task, refactor existing code, shed light on a compilation error, or even generate unit tests. LLMs are especially useful when coding in a programming language that you might not be familiar with. Overall, LLMs can speed up your development, if used correctly.

Even before ChatGPT's public release in November 2022, Microsoft offered an AI code assistant, GitHub Copilot [2], built on OpenAI's large language models. You can use this tool in the company's free code editor Visual Studio Code [3] through the GitHub Copilot extension [4]. However, GitHub Copilot operates as a cloud-based service, requiring continuous communication with Microsoft's servers. This can be problematic for developers working on confidential software, as code is sent to external servers.

Fortunately, recent years have seen the advent of smaller LLMs that you can run on your own hardware, provided it's powerful enough. An NVIDIA GPU with at least 8-GB VRAM is recommended, as these GPUs are particularly suited for running LLMs at accelerated speeds. In this article, I'll set up an Ollama [5] server for local LLM operation and show how to install and use the Continue [6] extension in Visual Studio Code to instruct the LLM in the background to assist you in various coding tasks.

Running LLMs with Ollama

Ollama is available for Windows, macOS, and Linux. Download the version for your operating system from the Ollama downloads page [7]. Ollama supports NVIDIA and AMD GPUs for accelerated inference. For AMD GPUs, it's recommended to install the latest AMD Radeon drivers [8] to have access to all features. Moreover, you need to install the AMD ROCm [9] software stack to enable GPU programming. For NVIDIA GPUs, installing the CUDA drivers [10] is necessary.

One of the best LLMs you can run locally for code generation and programming assistance is Mistral AI's Codestral 22B [11]. It's trained on a diverse dataset of 80+ programming languages, including Python, Java, C, C++, JavaScript, and Bash. Codestral can complete functions, generate tests, and complete partial code snippets. Download and run the model with 22 billion parameters using the following `ollama` command: `ollama run codestral`.

After downloading the 12-GB file, you can start chatting with the model through ollama's command-line interface. However, this sizable model requires a GPU with a minimum of 24-GB VRAM, or multiple GPUs totaling 24-GB VRAM, for optimal performance and user experience. Note that the model is distributed under the Mistral AI Non-Production License [12], which restricts its use to non-commercial purposes and research.

If you want to be able to run a coding LLM within 8-GB VRAM, DeepSeek AI's DeepSeek Coder 6.7B [13] is a commendable option.

The model is available in various sizes: 1 billion parameters (1B), 5.7 billion (5.7B), 6.7B, and 33B, allowing you to choose the size best suited to your hardware. Subsequently, DeepSeek Coder V2 [14] was released, coming in 16B and 236B sizes. DeepSeek AI's models permit commercial use.

To run the 6.7B DeepSeek Coder model, exit the previous Ollama session with `/exit` and run the following `ollama` command: `ollama run deepseek-coder:6.7b-base`. After downloading the 3.8-GB file, you can ask some code generation or explanation questions to the model and compare its speed and utility against Codestral.

You can find other code-related LLMs in Ollama's model library [15] (Figure 1). Noteworthy models include Google's CodeGemma (2B and 7B), Meta's Code Llama (7B, 13B, 34B, 70B), and IBM's Granite Code (3B, 8B, 20B, 34B). Always review the model's license to understand its terms of use.

In the next part, we'll install an excellent Visual Studio Code extension called Continue. You can configure it to use the LLM best suited to your needs. The selection of compatible LLMs is vast and includes Codestral and DeepSeek. For the purposes of this tutorial, we'll be using models other than the two just mentioned, to give you a broader overview of the possibilities.

Installing the Continue Extension

I assume that you're already running Visual Studio Code; otherwise, download it from its website [3]. Microsoft's popular code editor is available for Windows, macOS, and Linux. After this, click on the Extensions icon in the side bar (the boxes icon) and search for *continue*.

The extension *Continue - Codestral, Claude, and more* by the publisher *continue.dev* should be the first search result. Click on *Install* to add the extension (Figure 2).

After installation, the extension opens a tab in Visual Studio Code with a welcome message and useful instructions for first-time users. A new icon also appears in the sidebar to interact with Continue. The welcome message recommends moving this icon to the right, allowing you to see both Continue and the file explorer simultaneously. Simply drag the Continue icon from the left sidebar to the right and release it.

Choosing the Models

Continue offers a choice of LLMs, both cloud-based and local ones. For complete local operation, Continue recommends three models: Meta's Llama 3.1 [16] 8B for chat, BigCode's StarCoder 2 [17] 3B for code completion, and Nomic's Embed [18] for generating text embeddings to be able to query documents. First, download these three models for Ollama:

```
ollama pull llama3.1:8b
ollama pull starcoder2:3b
ollama pull nomic-embed-text
```

Once downloaded, click on the gear icon at the bottom right of Continue's side bar. This opens the extension's configuration file *config.json* in an editor tab. Here you can specify the models, following Continue's configuration documentation [19]. You'll find various JSON keys within the file. For instance, there's an empty list of models:

```
"models": [],
```

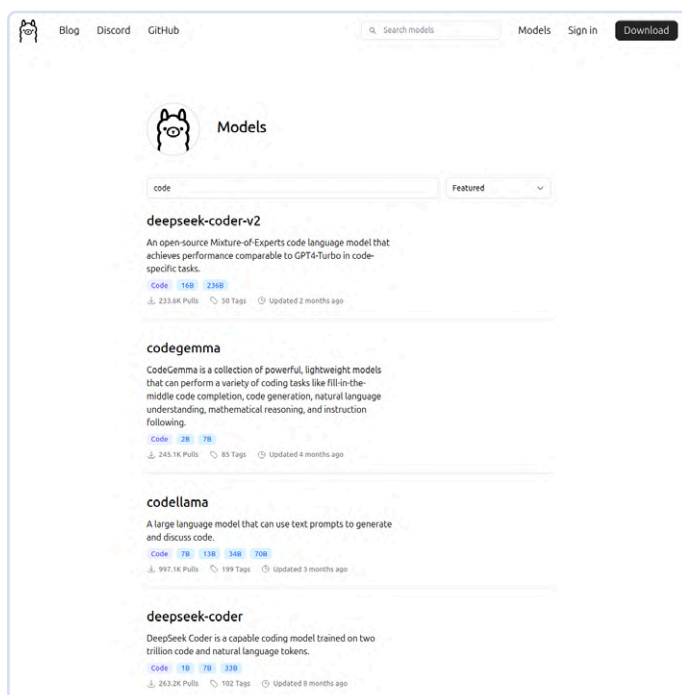


Figure 1: Find code-related LLMs in Ollama's model library.

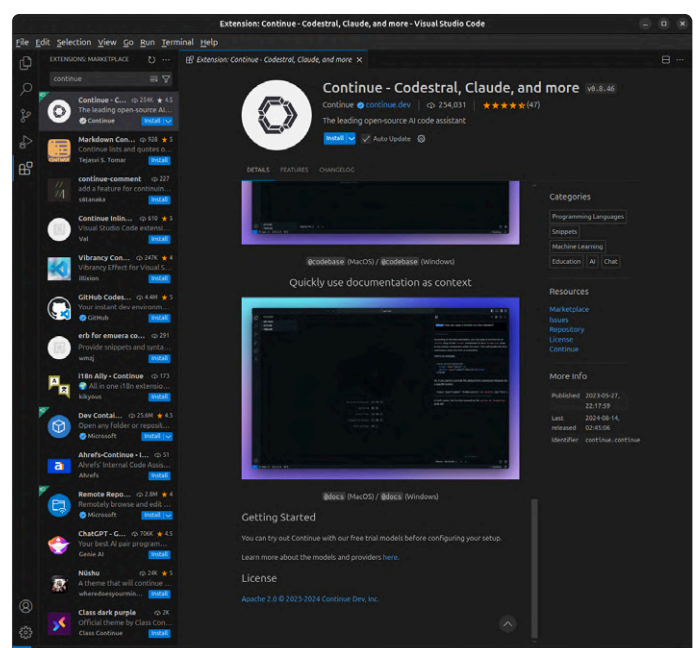


Figure 2: Install the Continue extension in Visual Studio Code.

Fitting Models in Your GPU's VRAM

If you have a GPU with 8-GB VRAM, the suggested models Llama 3.1 8B and StarCoder 2 3B can't fit simultaneously in your GPU's VRAM. Ollama will unload a model and load the other one every time you switch from asking questions to completing code, adding extra time between tasks. Using Llama 3.1 8B for both tasks isn't an option, as the models for code completion are trained with a highly specific prompt format, which allows them to respond to requests for completing code. Instead, consider replacing StarCoder 2 3B by DeepSeek Coder 1B (install with `ollama pull deepseek-coder:1.3b-base`). Tests on an NVIDIA GeForce RTX 4060 with 8-GB VRAM showed Ollama could run Llama 3.1 8B and DeepSeek Coder 1B concurrently.

Specify Ollama's Llama 3.1 model:

```
"models": [
  {
    "title": "Llama 3.1 8B",
    "provider": "ollama",
    "model": "llama3.1:8b"
  }
],
```

Alternatively, setting the model as *AUTODETECT* allows Continue to query Ollama for all available models, letting you choose the model for chat from a dropdown list. For code completion, specify the model as follows:

```
"tabAutocompleteModel": {
  "title": "StarCoder 2 3B",
  "provider": "ollama",
  "model": "starcoder2:3b"
},
```

Finally, specify the model for text embeddings:

```
"embeddingsProvider": {
  "provider": "ollama",
  "model": "nomic-embed-text"
},
```

After this, save your changes.

Regarding model selection, depending on your hardware it is possible that the suggested models Llama 3.1 8B and StarCoder 2 3B can't fit simultaneously. See the text box **Fitting models in your GPU's VRAM** for more information on that topic.

Using Continue

When you start developing in Visual Studio Code now, Continue operates in the background, enhancing your workflow. For instance, pressing the tab key while typing a line of code provides a suggestion

coming from the code completion model (StarCoder 2 if you followed Continue's recommendation).

You can also ask questions about code sections. Simply select a block of code and press **Ctrl+L** (**Cmd+L** on macOS) to add it to Continue's sidebar. You can then chat with the default model about this code (Llama 3.1 if you followed Continue's recommendations), for example, asking it to explain the code (**Figure 3**). You can always ask follow-up questions and have a conversation with the model.

If you ask the model to rewrite some parts of your code, it replies with a code block. Hovering with your mouse cursor over the code block reveals icons to copy the code, insert it at the cursor's position, or apply the suggested changes to the current file.

You can also let Continue directly edit code within the code editor. Select the part of the code you want to modify, press **Ctrl+I** (**Cmd+I** on macOS), and then instruct the model what you want to change. This is useful for refactoring, adopting a new approach, or adapting code to a new API. For each changed block of code, Continue displays a diff with buttons to accept or reject the changes. You can also press **Ctrl+I** again for additional instructions until you're satisfied with the model's edits.

The assistant can help fix errors, too. Running your code in Visual Studio Code's terminal and encountering an error? Pressing **Ctrl+Shift+R** (**Cmd+Shift+R** on macOS) lets Continue feed the question "I got the following error, can you please help explain how to fix it?" followed by the actual error message from the terminal to the LLM. The model then replies with a potential solution. Try its suggestion, and if it doesn't work, and you get an error again (the same or another one), just press **Ctrl+Shift+R** again to ask for help.

Adding Context

Until now, you've been using LLMs on selected parts of your code. However, you can also add context by one of Continue's built-in context providers, which allow you to use the LLM more effectively. During any conversation with the LLM, type `@code` to reference specific functions or classes from throughout your project, without having to select them. With `@problems` you can refer to the problems that Visual Studio Code shows for your currently opened code file, such as syntax errors.

Continue also indexes the codebase you're working on, allowing it to automatically pull in the most relevant context from throughout your workspace when answering general questions. Just type `@codebase` followed by your question to provide this context. It's best to start a new session for this, which you can do by clicking on the + icon at the top right of Continue's sidebar.

When the LLM answers your query with the `@codebase` context, it shows the context items it uses as "X context items". If you click on it, a list of the precise files and lines unfolds that the LLM is using as context for its answer. In its answer, it also refers to the relevant files. This works well for general questions, like how to add a new

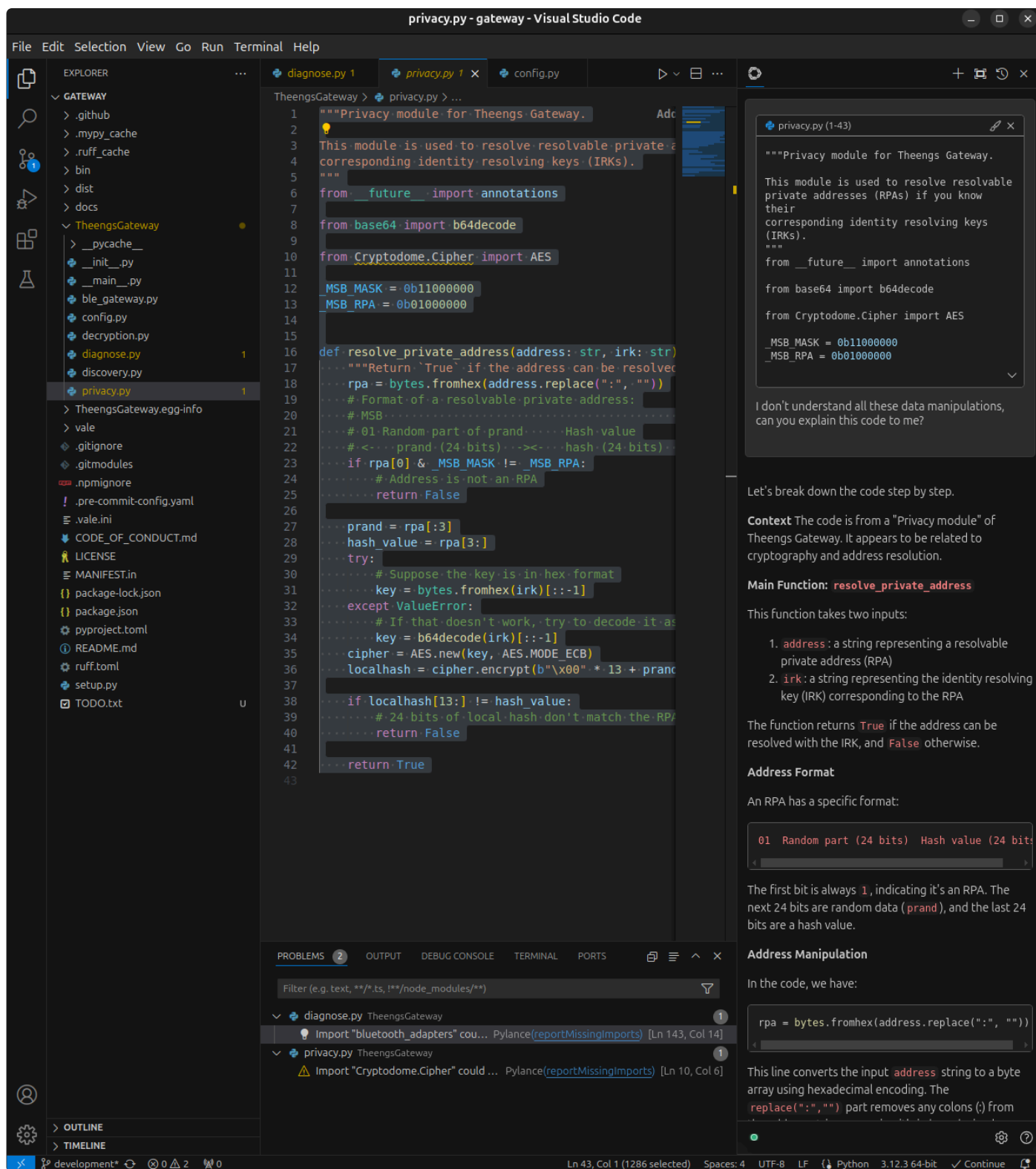


Figure 3: You can ask questions about your code in Continue's sidebar.

configuration option, how to implement a subclass using another subclass as an example, or how to add command-line parameters using Python's `argparse` module. You can also use `@folder` to ask questions about a specific folder within your project, instead of using the entire codebase as context.

If you're developing a more than trivial program, you often have multiple source files open in Visual Studio Code. You can refer to the contents of all your open files with `@open`. When asking your question within

this context, this should result in more relevant answers. However, the `@open` context provider isn't enabled by default. You need to add the following entry to Continue's `config.json` file in the `contextProviders` list:

```
{
  "name": "open",
  "params": {}
},
```

Another useful context provider is `@terminal`, which references the contents of Visual Studio Code's terminal. And with `@diff` you refer to all the changes you've made in your current Git branch. This way, you can ask the LLM for a review of your work before committing your changes to the project's Git repository.

External Context

Continue can incorporate external sources into LLM context as well. For instance, referencing a specific GitHub issue within LLM queries is possible by typing `@issue`. Enable this context provider with the following entry in Continue's configuration file:

```
{
  "name": "issue",
  "params": {
    "repos": [
      {
        "owner": "theengs",
        "repo": "gateway"
      }
    ],
    "githubToken": "github_pat_xxx"
  }
},
```

Note that you need to explicitly specify the repositories you want to be able to refer to. It's also recommended to include a GitHub personal access token [21] to avoid being rate-limited. When creating the access token on GitHub's website, giving just read-only access to public repositories is enough (**Figure 4**). Copy the access token immediately and paste it in Continue's configuration file because it's only visible once.

Another useful context provider is `@docs`, which is enabled by default. Continue pre-indexes a number of common documentation sites, most of them about web development technologies, but also the official Python documentation and Rust documentation and the C# language reference. Just start typing `@docs`, choose the documentation site from the dropdown menu or type the name of the project, and then after selecting the project ask your question with this documentation site as context.

You can also add any documentation site by selecting *Add Docs* in the dropdown menu that appears after typing `@docs`. Just enter the root URL of the documentation site and the name of the project. This is useful to add the documentation of the major libraries you're using in your project, so the LLM is aware of it when asking questions.

Built-In Commands

Continue also has some handy built-in *slash commands* [22]. Simply select a block of code, press `Ctrl+L` to start a session with the LLM, and enter `/edit` followed by instructions to edit the code. Continue shows the suggested changes in a side-by-side diff editor to accept or reject individual changes.

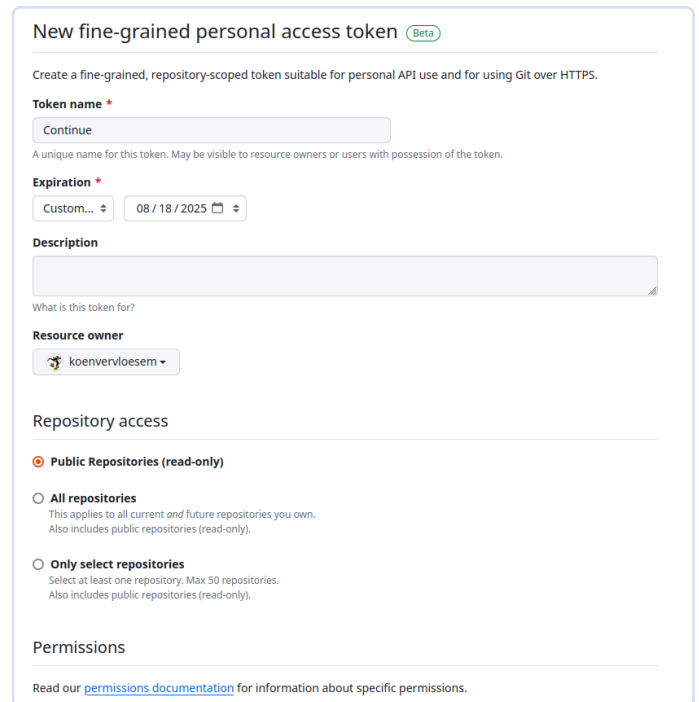


Figure 4: Create a GitHub personal access token to give Continue access to issues on your GitHub repositories.

Another command is `/comment`, which prompts the LLM to comment the selected code. The `/commit` command generates a commit message from the current Git diff with changes in your code, and `/cmd` generates a shell command based on your instructions and pastes it into Visual Studio Code's terminal. You only need to review and adapt the command and then press enter to execute it.

You can also create custom slash commands, using either TypeScript code or LLM prompts. By default, Continue includes one custom command using a prompt:

```
"customCommands": [
  {
    "name": "test",
    "prompt": "{{ input }}\n\nWrite a comprehensive set of unit tests for the selected code. It should setup, run tests that check for correctness including important edge cases, and teardown. Ensure that the tests are complete and sophisticated. Give the tests just as chat output, don't edit any file.",
    "description": "Write unit tests for highlighted code"
  }
],
```

The `{{ input }}` is a placeholder for the selected code. Then come two newlines (`\n`), followed by the prompt to the model. You can customize this to your liking. Then just select your code that you want to be tested, press `Ctrl+L` and type `/test`. Refine the results through follow-up questions if necessary.

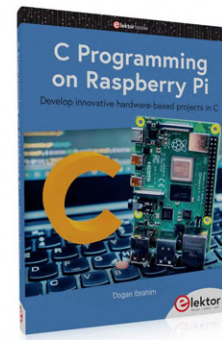
Powerful Tools at Your Disposal

Continue allows you to have a code assistant available at all times without relying on cloud services or needing to pay for API access. In my experience, having a GPU with 8-GB VRAM is really the minimum to be able to run some usable LLMs to assist with coding. LLMs are steadily improving, and even though they make errors, Continue's seamless integration with Visual Studio Code makes using LLMs as code assistants quite user-friendly. There's no need to switch between your editor, an LLM chat site, and documentation sites anymore. Experiment to find the most suitable models for your use cases and available hardware, and you'll have some powerful tools at your disposal to speed up your development. ◀

240448-01

Questions or Comments?

Do you have questions or comments about this article? Email the author at koen@vervloesem.eu, or contact Elektor at editor@elektor.com.



Related Products

- > **Dogan Ibrahim, *C Programming on Raspberry Pi* (Elektor, 2021)**
www.elektor.com/19703
- > **Daniel Situnayake, *AI at the Edge* (O'Reilly, 2023)**
www.elektor.com/20465

WEB LINKS

- [1] ChatGPT: <https://openai.com/chatgpt/>
- [2] GitHub Copilot: <https://github.com/features/copilot/>
- [3] Visual Studio Code: <https://code.visualstudio.com>
- [4] GitHub Copilot extension: <https://marketplace.visualstudio.com/items?itemName=GitHub.copilot>
- [5] Ollama: <https://ollama.com>
- [6] Continue: <https://www.continue.dev>
- [7] Ollama downloads: <https://ollama.com/download>
- [8] AMD Radeon drivers: <https://www.amd.com/en/support/download/drivers.html>
- [9] AMD ROCm: <https://rocm.docs.amd.com>
- [10] CUDA drivers: <https://developer.nvidia.com/cuda-downloads>
- [11] Mistral AI's Codestral 22B: <https://mistral.ai/news/codestral/>
- [12] Mistral AI Non-Production License: <https://mistral.ai/news/mistral-ai-non-production-license-mnpl/>
- [13] DeepSeek AI's DeepSeek Coder: <https://deepseekcoder.github.io>
- [14] DeepSeek AI's DeepSeek Coder V2: <https://github.com/deepseek-ai/DeepSeek-Coder-V2>
- [15] Ollama's model library: <https://ollama.com/library>
- [16] Meta's Llama 3.1: <https://ai.meta.com/blog/meta-llama-3-1/>
- [17] BigCode's StarCoder 2: <https://github.com/bigcode-project/starcoder2>
- [18] Nomic's Embed: <https://www.nomic.ai/blog/posts/nomic-embed-text-v1>
- [19] Continue's configuration documentation: <https://docs.continue.dev/setup/configuration>
- [20] Continue's built-in context providers: <https://docs.continue.dev/customization/context-providers>
- [21] GitHub personal access token: <https://tinyurl.com/bykfswr9>
- [22] Slash commands: <https://docs.continue.dev/customization/slash-commands>

Solar Charge Controller with MPPT (2)

The Circuit

By Roland Stiglmayr (Germany)

For the second article in this series, we convert the block diagram of the MPPT solar charge controller into a functioning circuit. As always with power electronics, the main focus is on the currents within the circuit and their implications for certain components. Calculations are required!



The first part of this article series [1] covered the basics, advantages, and functionality of an MPP tracker in a solar charge controller and ended with a block diagram showing the design of an MPPT. In part 2, the block diagram (**Figure 1**) is converted into a functioning circuit (**Figure 2**). In addition, the functionality of the sub-circuits is explained in detail.

The Internal Power Supply

Voltage V_{PV} from the solar panel is fed via fuse F1 to suppressor diode D2, which suppresses brief overvoltages. F1 trips in the event of very high fault currents caused by inductive coupling, and also makes it possible to disconnect the module from the solar panels.

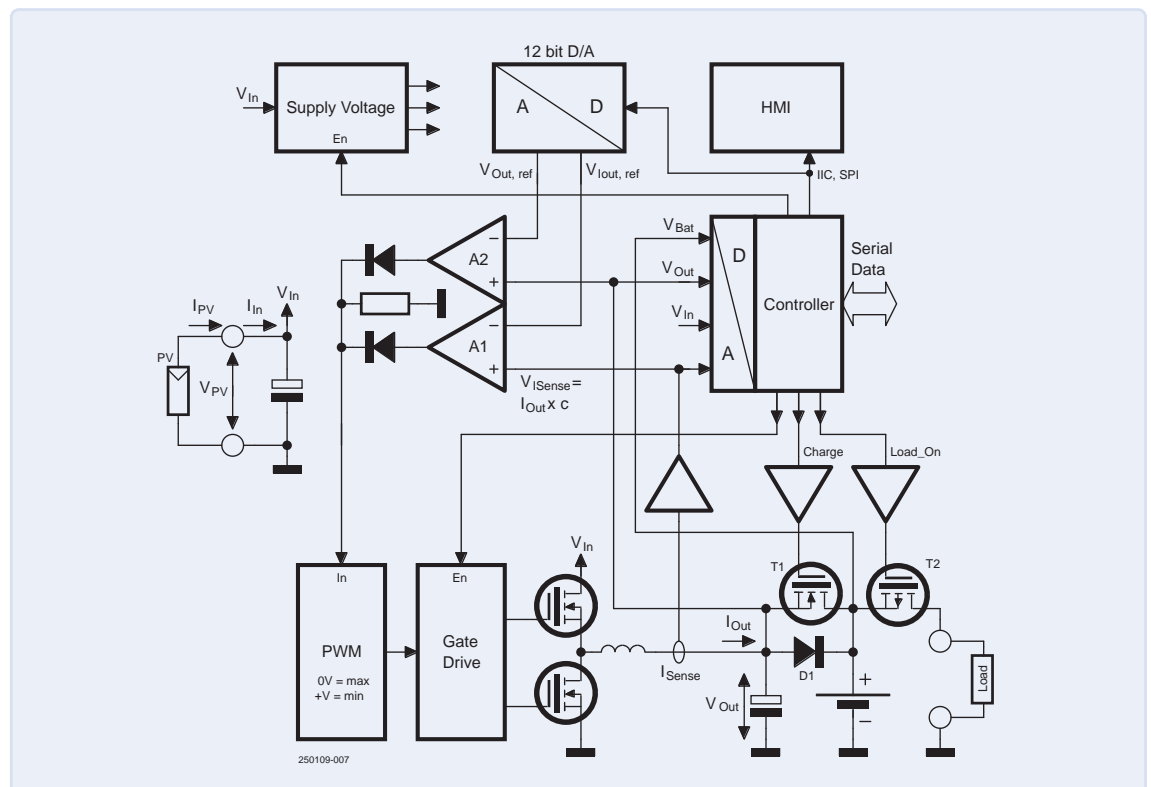


Figure 1:
Reminder: The block
diagram from the first
part of the series.

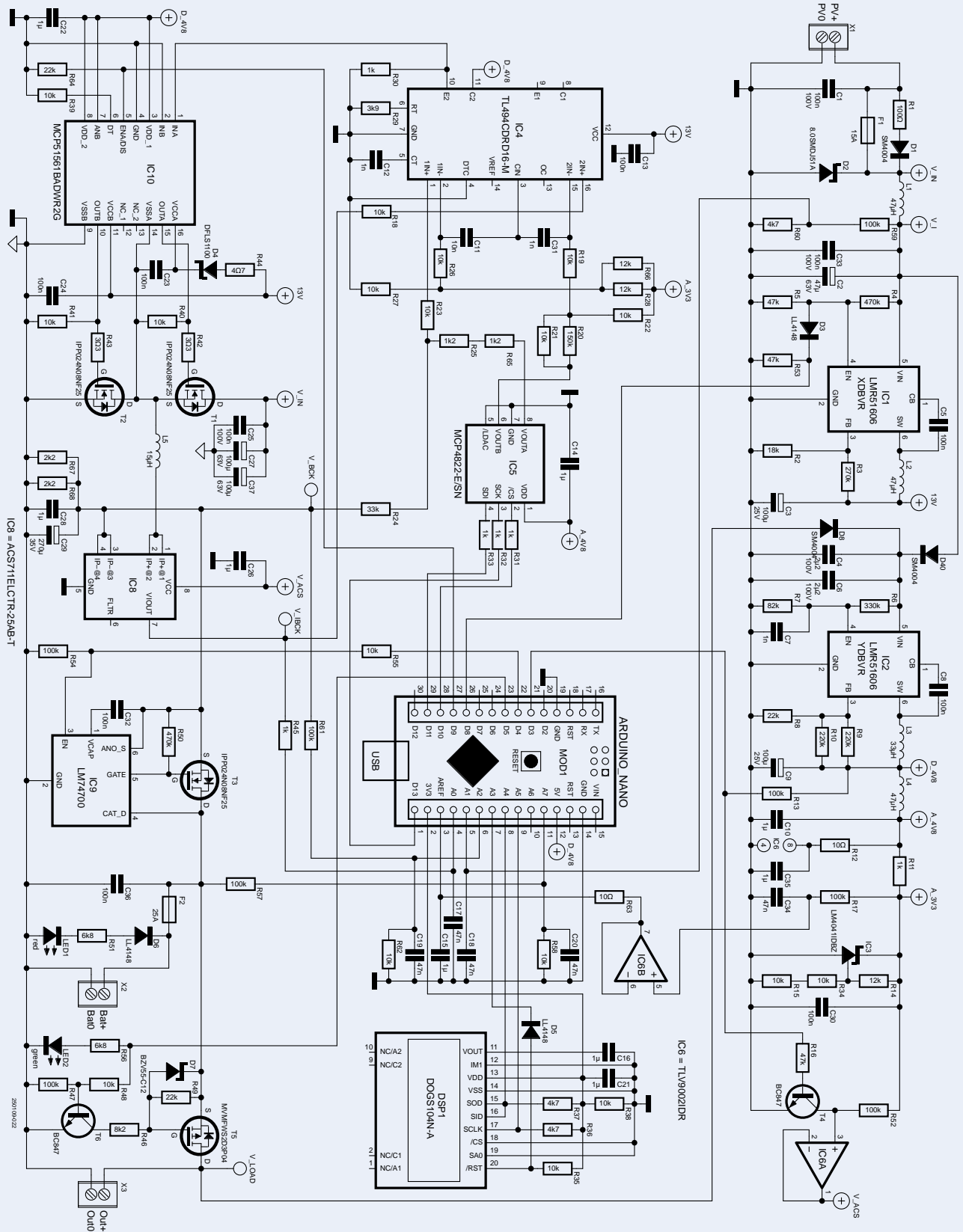


Figure 2: The complete circuit diagram of the MPPT solar charge controller.

Input voltage V_{IN} is applied to the blocking capacitors of the power section and is fed via L1 as V_I to the subsequent low-power step-down converters, IC1 and IC2. Since they also have integrated synchronous rectification, only the external storage choke is required for operation. They operate up to a maximum input voltage of 65 V. These LMR3605

components are available in several versions, which differ in terms of switching frequency and the two operating modes: fixed frequency or discontinuous (gap) operation. Due to its higher efficiency, the version with discontinuous operation is used.

Technical Specifications

Input voltage	20...60 V
Input current (max.)	15 A
Output voltage	10...29 V
Charging current (max.)	20 A
Charging power (max.)	280 W
Load current (max.)	25 A
Efficiency (230 W)	>96%
Internal power consumption (low power)	<8 mA

IC1 generates 13 V from V_{IN} to supply the gate control of the power MOSFETs. IC1 is activated by the microcontroller via diode D3 when the voltage of the PV panel is high enough to operate the power section of the tracker.

IC2 generates the 5 V supply for the microcontroller and the analog section. Its input is supplied by V_{IN} via D40 or, if V_{IN} is not available, via D8 by V_{LOAD} from the battery. The enable input (EN) is used for shutdown in case of undervoltage. It is set to the lowest operating voltage of 6 V by the voltage divider, R6 and R7.

The output voltage of IC2 is cleaned of high-frequency interference by L4 and is used to supply the analog section. The adjustable precision voltage source, IC3, generates the reference voltage of 3.3 V. Op-amp IC6B buffers this voltage and supplies it to the reference voltage input AREF of the Arduino. Op-amp IC6A uses the reference voltage as the supply voltage, V_{ACS} , for current sensor IC8. To keep the power consumption as low as possible, the controller switches off this voltage via T4 when it is dark. Op-amp IC6, a TLV9002, is characterized by its high stability in unity gain operation and must not be replaced by an arbitrary type.

The High-Power Section

In the block diagram, the function blocks *PWM*, *Gate Driver*, the two MOSFETs in half-bridge configuration, and the storage choke with the output capacitor form the high-power section of the MPP tracker. If you add the control from A1 and A2, you can see a pure buck converter with synchronous rectification. Synchronous rectification means that MOSFET T2 (**Figure 2**) takes over the role of the usual freewheeling diode. This is the only way to achieve an efficiency of over 95% at currents of 15 A and higher. To achieve this goal, the MOSFETs must have a very low $R_{DS(on)}$ and, to keep switching losses low, the semiconductor capacities must be as small as possible. Only N-channel FETs can meet these requirements.

Since the high-side FET requires a gate voltage that is higher than V_{IN} , the bootstrap circuit consisting of C23, D4, and R44 adds 13 V to V_{IN} and makes it available to the gate driver. Several aspects must be considered when dimensioning the storage choke: If a small inductance value is selected, the magnetization current increases and, in turn, the ohmic losses decrease for the same component size. The magnetization current represents the ripple current of output capacitor C29 and therefore has a major influence on its load. A quick calculation shows the range of the magnetization current.

Choke current in general: $i_L = 1/L \times \int u_L \times dt$

Since u_L only assumes two constant values ($V_{IN} - V_{BCK}$ and $-V_{BCK}$), the following applies:

$$\Delta i_L = 1/L \times u_L \times \Delta t; u_L = \text{constant}$$

When loading the choke, the following results:

$$\Delta t = 1/f \times (V_{BCK} / V_{IN})$$

$$u_L = V_{IN} - V_{BCK}$$

With $V_{IN} = 55$ V, $V_{BCK} = 12$ V, $f = 250$ kHz, $L = 4.7$ μ H, we get:

$$\Delta i_{L,4.7\mu H} = (1 / 4.7 \mu H) \times (55 \text{ V} - 12 \text{ V}) \times 0.87 \mu s = 8.0 \text{ A}$$

The ripple current is the root mean square value of Δi_L , approximately $\Delta i_L / 2 = 4$ A. This value is very high, which is why another attempt is made with 10 μ H, resulting in $\Delta i_{L,10\mu H} = 3.8$ A. As expected, the value has halved, but is still very high. We can solve this problem!

Assuming the above values, a simulation (**Figure 3**) is used to determine the voltage and current curves in the converter and calculate the load on the capacitors.

The first curve shows the voltage curves, the second shows the current, $I(L5)$, through storage choke L5 and load current $I(\text{Load})$. The superimposed alternating current is the magnetizing current applied to C29. The third curve shows the current, $I(V_{IN})$, supplied by the solar panel, the current flowing through the high-side FET, $I(T1_{DS})$, and current $I(C27)$ flowing through one of the two decoupling capacitors, C27||C37. The comparison of the two curves clearly shows how an input current of 4 A becomes an output current of 15 A. The fourth curve, which uses a different time scale, shows the energy dissipated at the equivalent series resistance (ESR) of C27 within the observation period of 7 ms. The power dissipation of the capacitor and the associated ripple current can be calculated from the energy.

$$\text{Energy: } \Delta E = P \times \Delta t; P = \Delta E / \Delta t$$

$$\text{Power: } P = (E_{8ms} - E_{1ms}) / 7 \text{ ms} = (64.4 \text{ mJ} - 63.1 \text{ mJ}) / 7 \text{ ms} = 0.19 \text{ W}$$

$$\text{Current: } I_{eff} = \sqrt{(P/ESR)} = \sqrt{(0.19 \text{ W}/17 \text{ m}\Omega)} = 3.3 \text{ A}$$

If electrolytic capacitors were subjected to such high currents, they would explode in no time! Only high-quality aluminum polymer capacitors can withstand currents of this magnitude.

Current sensor IC8 is connected in series with choke L5. The current is measured by detecting the flux density generated by the current in the sensor's internal conductor. An integrated Hall sensor measures the magnetic field. The advantage of this principle is that there are no losses. The sensor type used here is suitable for measuring ± 25 A with a sensitivity of 55 mV/1 A. Since the sensitivity depends on the supply voltage, it is supplied with the reference voltage.

What is still missing is the control of the MOSFETs by gate driver IC10. In addition to generating the high gate currents of 7 to 9 A, it separates the different reference potentials of the input and output. For this purpose, it is coupled inductively internally to achieve isolation between all stages. Furthermore, the gate driver generates the complementary output signals required to control the high-side and low-side FETs.

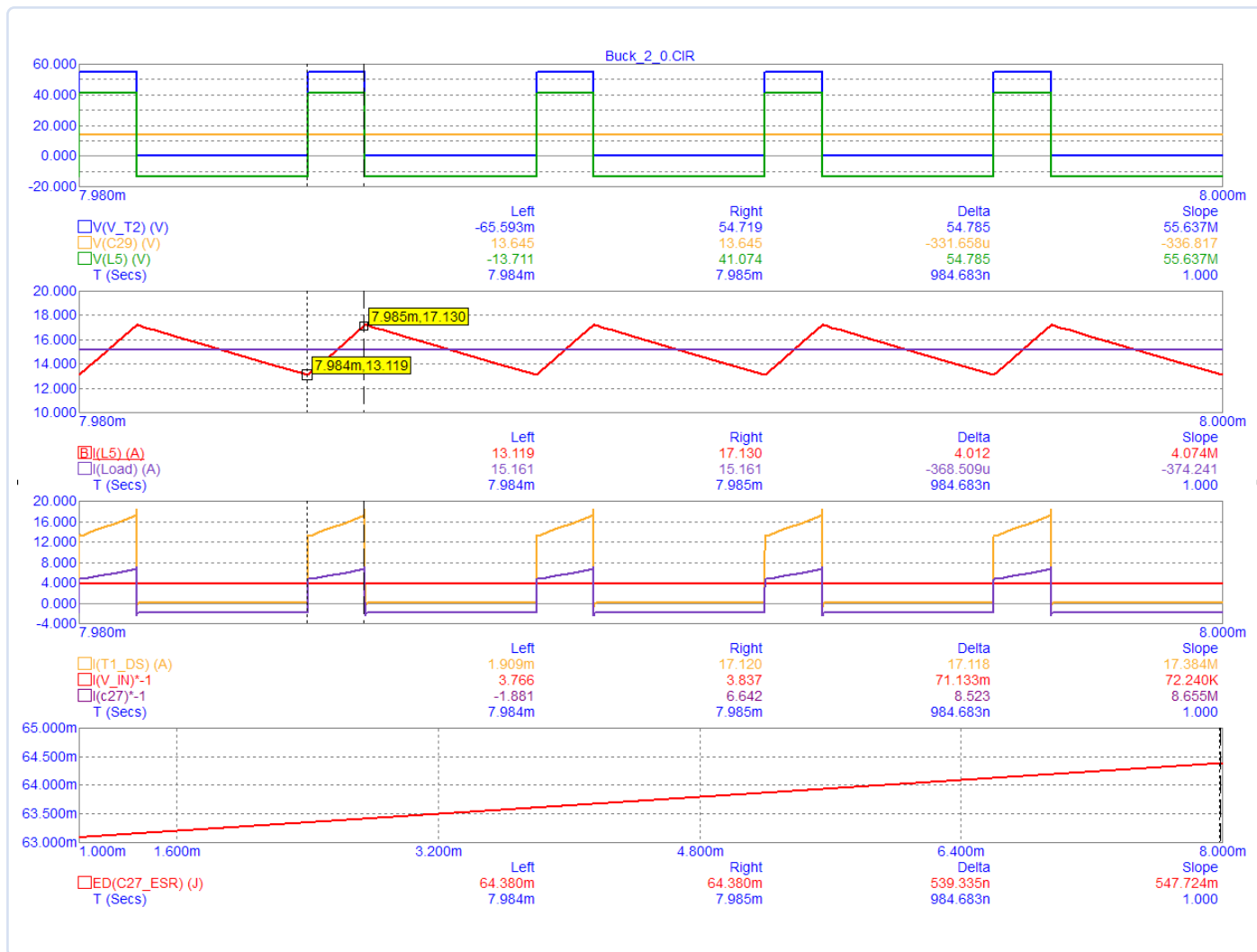


Figure 3: Simulation of the power section in steady state, with a load current of 15 A, $L5 = 10 \mu\text{H}$, and $C27 = 100 \mu\text{F}$.

To prevent the dreaded shoot-through, i.e., the simultaneous conduction of both FETs, it inserts a dead time when commutating the current from one FET to the other. The dead time is adjusted with R39. The microcontroller can disable the component by applying a low signal to the ENA/DIS pin.

The above considerations apply to every step-down-regulator and can therefore also be used very well for your own developments.

Battery and Load Management

To charge the battery, the microcontroller activates MOSFET T3 via IC9, which switches the output of the converter to the battery. It is essential to disconnect the battery from the converter; otherwise the converter will discharge the battery. Due to the polarity of body diode T3, only an N-channel FET can be used here, but this requires an increase in its gate voltage. IC9, a component originally designed for reverse polarity protection circuits, performs this task. In addition to increasing the voltage, it monitors T3's DS voltage and immediately switches off the FET if the source voltage falls below the drain voltage. This reliably prevents reverse operation, which could cause the converter to become a boost converter.

The microcontroller continuously measures the battery voltage and switches the battery to the load output when the voltage is sufficient. This is ensured by MOSFET T5, which must be a P-channel type due

to the body diode's polarity. Otherwise, the load would always be connected to the battery via the body diode. As already mentioned above, in the dark, the battery supplies the controller via diode D8, which is connected to the load output. If there is no input voltage and the controller detects that the battery voltage is too low, the load is disconnected, which leads to a complete shutdown of the system.

The Control Circuit

The most complex part of the solar controller is the power converter's control circuit. It adjusts the PWM's pulse width so that the output current corresponds to a value specified by the controller, and the maximum output voltage cannot exceed the battery's end-of-charge voltage.

The setpoints for current and voltage are provided by the 12-bit D/A converter, IC5. It has two outputs, an internal reference voltage of 2.048 V, and is controlled via an SPI interface.

IC4 contains two control amplifiers and the PWM generator. **Figure 4** shows a section of the internal circuit and the associated external circuitry. The outputs of the control amplifiers are connected to two diodes in such a way that the higher signal always dominates. Since the PWM generator reduces the pulse width as the voltage increases, the actual values must necessarily act on the non-inverting inputs. If the current or voltage rises above its setpoint value, the control

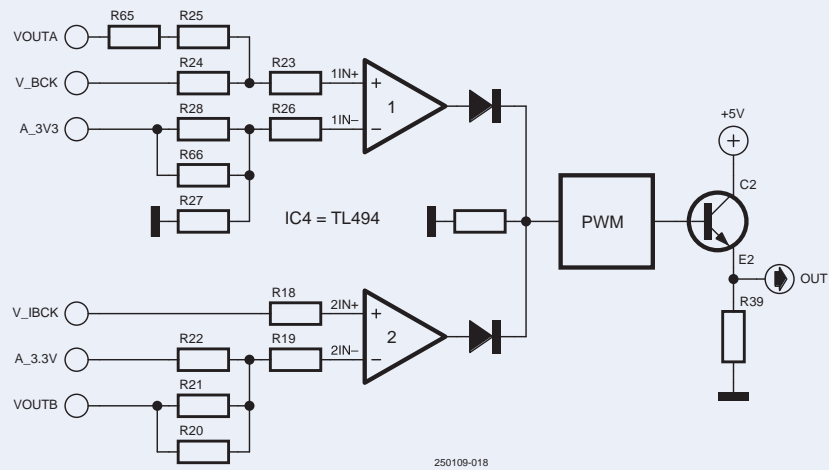


Figure 4: The control circuit with IC4 and external wiring.

amplifier's output voltage also increases and, as a result, the pulse width is reduced. This causes the deviating value to fall back to the setpoint value.

The signal from the current sensor, V_IBCK , which represents the output current, is present at the non-inverting input of control amplifier 2. Without current, the quiescent level of the sensor is half its supply voltage, i.e., $A_3V3 / 2$. V_IBCK is:

$$V_IBCK = I_BCK \times S_{Sense} + (A_3V3 / 2);$$

where $S_{Sense} = (55 \text{ mV} / 1000 \text{ mA}) = 0.055 \text{ mV} / 1 \text{ mA}$

The setpoint of the output current, which is obtained from the reference voltage A_3V3 and $VOUTB$ using a voltage divider consisting of $R20 \parallel R21$ and $R22$, is present at the inverting input. The base point of the voltage divider is present at the $VOUTB$ output of the D/A converter, which controls the potential of this point in order to set the output current. The following applies:

$$V_IBCK = ((A_3V3 - VOUTB) \times (R21 \parallel R20) / (R21 \parallel R20 + R22) + VOUTB$$

where $(R21 \parallel R20) / (R21 \parallel R20 + R22) = k = 0.48387$ and $V_IBCK = V2IN- = V2IN+$

$$V_IBCK = A_3V3 \times k - VOUTB \times k + VOUTB = A_3V3 \times k + VOUTB \times (1 - k)$$

$$VOUTB = (V_IBCK - A_3V3 \times k) / (1 - k)$$

$$VOUTB = I_BCK \times S_{Sense} / (1 - k) + A_3V3 \times ((1/2 - k) / (1 - k))$$

$$VOUTB = I_BCK \times S_{Sense} \times 1.9375 + A_3V3 \times 0.031252$$

$$VOUTB = DAC_B \times 2048 \text{ mV} / 4096 = DAC_B \times 1/2$$

$$DAC_B = VOUTB \times 2$$

Two constants, I_SetG and I_SetZ , are used in the software (they may need to be adjusted by the user to compensate for measurement errors):

$$I_SetZ = A_3V3 \times 0.031252 \times 2^{16} \times 2 = A_3V3 \times 4096$$

$$I_SetG = S_{Sense} \times 1.9375 \times 2^{16} \times 2 = 13967$$

Resulting in:

$$DAC_B = (I_BCK \times I_SetG + A_3V3 \times I_SetZ) / 2^{16}$$

The inverting input of control amplifier 1 is at a constant voltage of approximately 2050 mV, which is generated from A_3V3 via the voltage divider consisting of $R28$, $R66$, and $R27$. This voltage is the reference

for the converter's output voltage of the converter. The non-inverting input of control amplifier 1 sums part of the converter voltage, V_BCK , with $VOUTA$ from the D/A converter. The value of the setpoint that the D/A converter outputs is calculated as follows:

$$V_BCK = I_{R24} \times R24 + V1IN+$$

$$I_{R24} = (V1IN+ - VOUTA) / (R25 + R65)$$

$$V_BCK = ((V1IN+ - VOUTA) \times R24 / (R25 + R65)) + V1IN+$$

$$V_BCK = V1IN+ \times (c + 1) - VOUTA \times c$$

$$VOUTA = V1IN+ \times ((c + 1) / c) - V_BCK / c$$

where $c = R24 / (R25 + R65) = 13.75$

$$VOUTA = V1IN+ \times 1.07272 - V_BCK \times 1 / 13.75$$

$$V1IN+ = V1IN- = A_3V3 \times R27 / (R28 \parallel R68 + R27)$$

$$VOUTA = A_3V3 \times 0.625 - V_BCK \times 1 / 13.75$$

$$DAC_A = VOUTA \times 4096 / 2048 = VOUTA \times 2$$

$$V_SetZ = A_3V3 \times 0.625 \times 1.07272 \times 2^{16} = A_3V3 \times 43939$$

$$V_SetG = (1 / 13.75) \times 2^{16} = 4766$$

Resulting in:

$$DAC_A = (V_SetZ - V_SetG \times V_BCK) / 2^{15}$$

The derivation of these formulas has been described in such detail because they will be implemented later in the program code.

The frequency of the PWM is set with $R29$ and $C12$. The emitter follower at the output of the PWM operates on resistor $R30$.

Microcontroller and LC Display

For simplicity's sake, an Arduino Nano is used. It offers everything needed in a small footprint: digital I/O ports, a 10-bit ADC with eight analog inputs, SPI and I²C buses, and a USB interface. Above all, however, its programmability via the bootloader is very efficient. This is used intensively during commissioning, as the entire analog section is adjusted via parameters to be set in the program. In normal operation, the operating data is output serially via the USB interface. The use of the serial plotter is also planned.

The ADC benefits greatly from the external reference voltage, $AREF$. This results in astonishingly good accuracy. Voltage dividers limit the analog signals to the maximum input level of 3.3 V. More on this in the final part of the article.

An LC display is used due to its low power consumption. DSP1 is powered from the Nano's internal 3.3 V voltage and controlled via I²C.

Construction

This concludes the detailed description of the electronics. While we at Elektor are preparing the third and final part of the article on the function and correct configuration of the software, you can order the components and have the circuit board manufactured (if you are interested in building your own solar controller). All the necessary documents — parts list and KiCad files — can be found at [4].

The construction of the circuit is quite simple due to the generously dimensioned circuit board, provided you are familiar with soldering SMD components. Hot air works very well, but a soldering iron with a fine tip will also do the job. As always, you should start with the smallest components, then work your way up, and finally fit the large through-hole components.

The storage choke, L5, must be insulated from the vias of the circuit board using a heat-resistant film or a mica disc. The clips of the fuse holders must be well tinned before soldering, to prevent cold solder joints.

Although the Arduino can be soldered directly onto the board by sticking insulating film to its underside, it is better to provide a socket for it. Either way, it should not be fitted until commissioning. The display is also mounted in a socket for better visibility.

The MOSFETs, T1, T2, and T3, must be cooled. Either place the module on a heat sink as shown in the title image and screw the transistors onto this plate in an insulated manner, or mount the insulated MOSFETs on a heat sink with a thermal resistance of less than 3 K/W. T1 and T2 should not be fitted until commissioning.


In Conclusion

The solar charge controller is ideal for use with today's standard solar panels with 2x72 half cells. Another cost-effective option is to use four small panels with 36 cells each, connecting two in series and then connecting the two series connections in parallel. This creates an ideal arrangement in terms of shading [2]. In any case, the specifications of the tracker must be observed.

Due to its robustness and low price, a lead-acid battery [3] is the best choice for the battery. Lithium batteries can only be operated with an additional battery management system that monitors voltage and current and balances the cell voltage. It should be noted that the charging current should not be limited due to an unfavorable battery selection, as otherwise the maximum available power will not be available to the load.

When installing the tracker, the connection to the load should be equipped with a disconnect device and a fuse rated for the load current. This precautionary measure serves to reduce the risk of fire in the event of a short circuit in the load.

The third and final part of this article series will focus almost exclusively on the MPPT software in the Arduino Nano, describing its functionality in detail. The solar charge controller will then be put into operation — this will also require certain parameters to be entered into the software.

The author is aware that the above is "heavy going." However, since many sub-areas have been discussed that may also be useful in other projects (analog data acquisition, chargers, power supplies with digital setpoint specification, and current and voltage control in power electronics), this project is sure to be of interest to many readers. 

Edited by Rolf Gerstendorf, translated by Jörg Starkmuth — 250109-B-01



About the Author

Roland Stiglmayr studied information technology in the 1970s and has over 40 years of experience in research and development. His work has focused on the development of computer mainframes, fiber optic-based data transmission systems, RRHs for mobile communications, and contactless energy transmission systems. Today, he works as a consultant. He is particularly passionate about sharing his knowledge.

Questions or Comments?

Do you have questions or comments about this article? Email the author at 1134-715@online.de, or contact Elektor at editor@elektor.com.



Related Products

> **Waveshare Solar Power Management Module**
www.elektor.com/20488

> **Waveshare Solar Power Manager (C)**
www.elektor.com/20490

WEB LINKS

[1] Roland Stiglmayr, "Solar Charge Controller with MPPT," Part 1, Elektor 5-6/2025: <https://elektormagazine.com/250109-01>

[2] Elektor Special: Solaranlagen und Photovoltaik [German]: <https://elektor.de/20597>

[3] Walter Ribbert, "Pimp My Car Battery Charger (Part 1)," Elektor Circuit Special 2024: <https://elektormagazine.com/240040-01>

[4] Download: <https://www.elektormagazine.com/labs/solar-charge-controller-with-mpp-tracking>

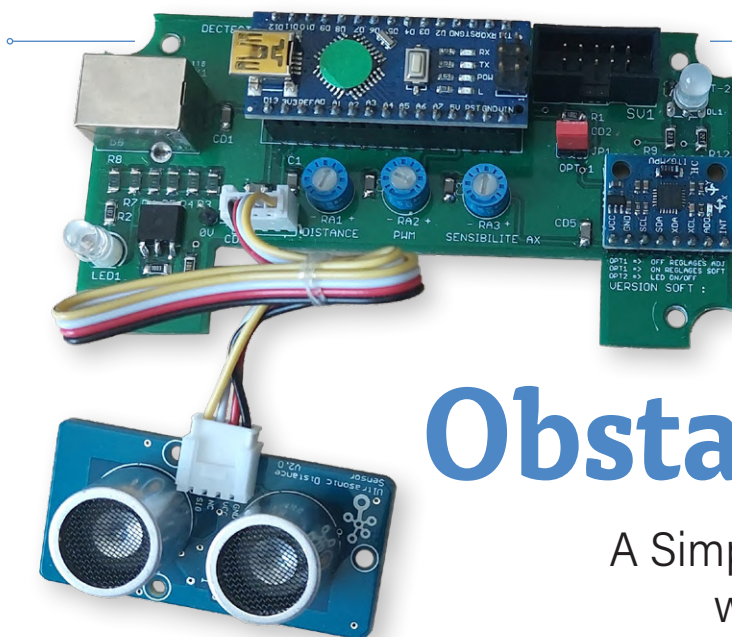


Figure 1: A neat PCB to hold the Arduino Nano.

Ultrasonic Obstacle Detector

A Simple Project to Help Those with Impaired Vision

By Pascal Rondane (France)

Read how a simple ultrasonic sensor, a vibrating motor, and one or two microcontrollers turned into a practical upgrade for a walking frame. This ultrasonic obstacle detection module is capable of identifying objects up to four meters away.

This project originated with a request from my uncle, who lived in a retirement home and experienced significant vision problems. He used a walking frame, but had difficulty detecting obstacles in his path, which were a daily challenge. To help him get around as best as possible, I designed an ultrasonic obstacle detection module. It's capable of identifying objects up to 4 m away, with a detection angle of approximately 20°. When an obstacle is detected, a red LED lights up, and a micro motor activates to generate vibrations on the walker's frame, which can be felt by the user as a warning.

First Version

To keep the project simple and modular, an Arduino Nano was used for this project, as well as a ready-made Grove ultrasonic sensor module from Seeed Studio (**Figure 1**). To generate vibrations is easy: simply attach an off-center mass to the shaft of a small motor. Here, to make the task even easier, I used the Vibration Motor Unit (N20) module from M5Stack [1], which, for just under three euros, provides a ready-made unit. Besides the motor and eccentric mass, it includes a flyback diode, a MOSFET to control the motor from a microcontroller GPIO pin, all housed in a plastic enclosure that's easy to mount near the handles of the walking frame using the two mounting holes (**Figure 2**).



Figure 2: The vibration motor is bolted near the handle. A piece of PVC pipe hides the wiring.

Schematic

The schematic is shown in **Figure 3**. A custom PCB was designed to hold the Arduino Nano and provide connectors to ease the wiring of the vibration motor and the ultrasonic sensor module. Both the vibration intensity and detection range are adjustable using two potentiometers RA1 and RA2. A jumper (JP1) allows quick selection of a default detection distance of one meter and maximum motor power. The PCB connects to the micro motor module via a 10-pin connector, which is wired to a 3.5 mm, panel-mount stereo TRS jack on the enclosure. From there to the motor, a standard 3.5 mm stereo cable is used.

Power is supplied by a USB cable connected to a 5 V USB power bank. A 10 Ah battery offers about one week of autonomy, and recharging is made convenient using a standard phone charger. A known issue

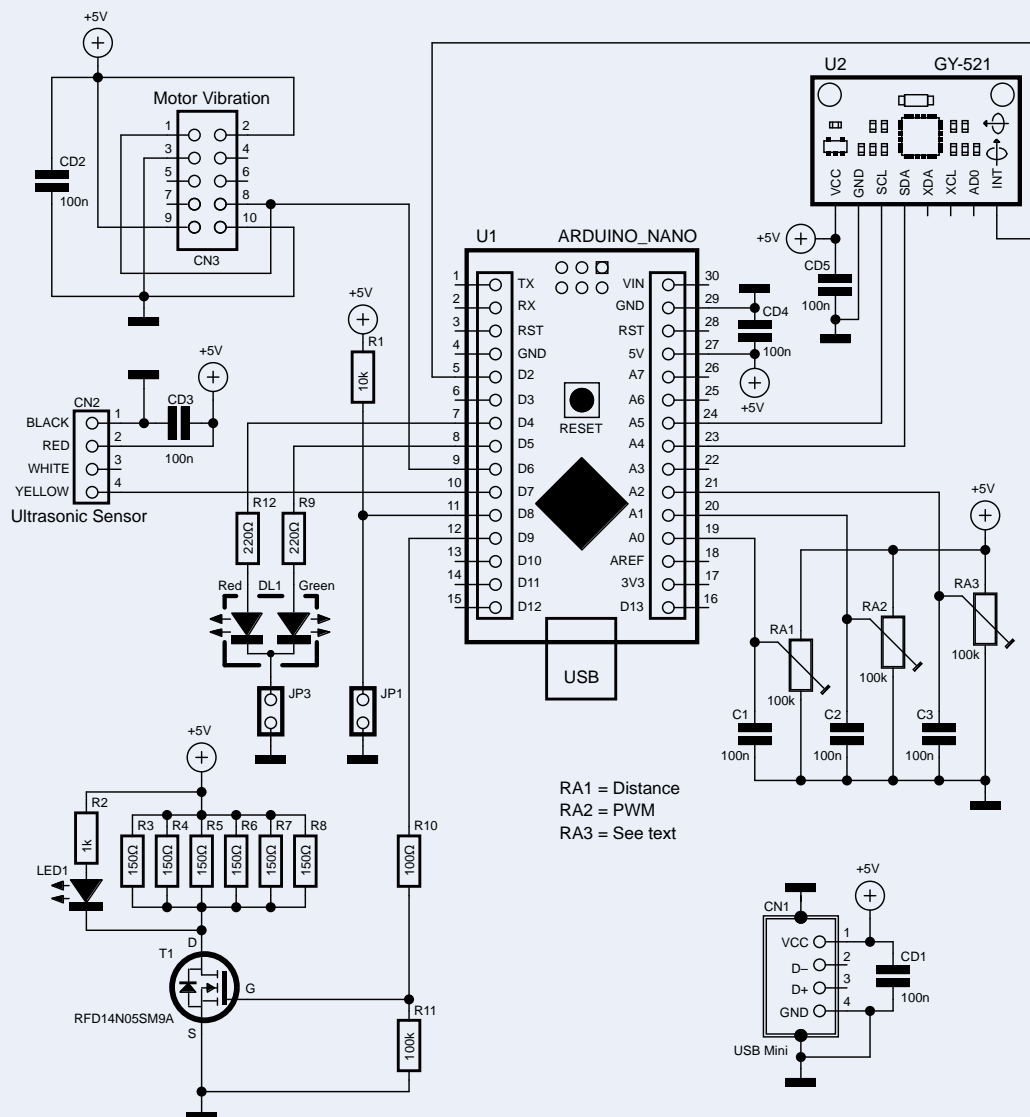


Figure 3: Schematic of the Arduino board.

with these USB power banks is their tendency to shut down when current draw is too low. To prevent this, the circuit uses a MOSFET (T1) to periodically activate load resistors (R3 to R8) for a few milliseconds, drawing a short current pulse to keep the power bank awake. LED1 gives a visual indication of these pulses. This approach is inspired by the "USB Battery Interface" project published in Elektor [2]. The potentiometer RA3 controls the interval to keep the USB battery from switching off.

A bi-color LED (DL1) signals system status: green for normal operation, red when an obstacle is detected. If desired, the LED can be disabled with the JP3 jumper. On the PCB, I have added a footprint for an 8-pin GY-521 module, using an MPU6050 accelerometer to be added in the future to detect motion and automatically wake the system. In the end, this feature hasn't been used.

Practical Build

The PCB and the ultrasonic transducer fit in a translucent Hammond enclosure, which was mounted under the seat of the walker, with the sensor pointing forward. The control box, battery and cables are secured under the seat (**Figure 4**) using zip-ties and small, custom-made sheet



Figure 4: Mounting everything under the seat.

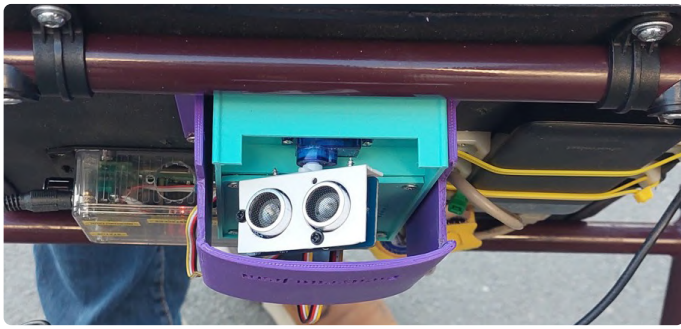


Figure 5: The sensor on its pivot.

metal brackets. Of course, very short screws must be used, so that they don't go all the way through the wooden board.

Arduino Software

The software is quite simple; the detection distance, the vibration intensity of the vibrations (i.e., the motor speed) and the delay period used to prevent the USB battery from switching off can be configured. Depending on the specific battery model you are using, the exact timing of these pulses may need to be adjusted. A debug mode is also available to display real-time debug information via the serial monitor.

The *Ultrasonic.h* library needs to be included to enable the use of the Grove ultrasonic sensor: `#include "Ultrasonic.h"`. Several constants are then defined to manage the input and output pins, as well as to set default values for various parameters. In `setup()`, the relevant input and output pins are configured, the PWM output as well as the serial port are initialized.

In the main `loop()`, the program continuously measures the distance to an obstacle using the ultrasonic sensor with a builtin function in the *Ultrasonic* library (`RangeInCentimeters = ultrasonic.MeasureInCentimeters();`) and reads the analog voltages on the potentiometer inputs. When an object is detected within the configured range, a red LED is illuminated and a PWM signal is sent to activate the vibration motor. Finally, the main loop also keeps the USB power bank active by repeatedly switching the load resistors on for a few milliseconds and then off again.

The Project Evolves

Following real-world tests, my uncle (who was a particularly discerning user) pointed out that the 20° detection angle of the ultrasonic sensor was insufficient, especially for detecting obstacles on the sides of the walker. To increase the detection range, several approaches could be used; I chose to mount the ultrasonic sensor on a bracket, and make it pivot using a small RC servo (Figure 5).

As I wanted to avoid modifying the existing PCB and Arduino software, I used this as an opportunity to build a second PCB to control the servo, this time using a small XIAO

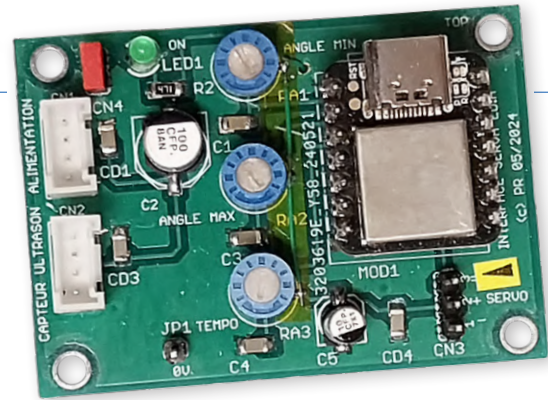


Figure 6: Secondary board to drive the servo.

SAMD21 module from Seeed Studio [3] that I had in stock. You can see the result in Figure 6. Three potentiometers are provided to adjust minimum angle, maximum angle and sweep rate. This allows a sweeping scan of up to 180° if needed. The schematic is shown in Figure 7.

This secondary SAMD21 board is powered by connecting it to the first Arduino board described above, via connector CN1 (which connects to CN2 on the first board). In this way, the two PCBs and the ultrasonic sensor are daisy-chained, with the ultrasonic sensor connected to CN2 of the SAMD21 module. This way, the wiring is kept simple; the primary Arduino board remains in charge of dealing with the sensor. The servo and sensor are protected by a small 3D-printed enclosure, for which the 3D model file is available on Elektor Labs [4].

SAMD21 Software

The Arduino environment was also used to program the XIAO module. After including the *Servo.h* and *Arduino.h* libraries, the servo is attached to pin 9. The potentiometer inputs are configured, and serial communication is started at 9600 baud. In the main loop, the potentiometer values are read and converted into angles (between 0 and 180 degrees) and a sweep duration (up to 3000 ms). The `servo.write()` function is used; this function takes the desired angle for the servo as an argument. For a smooth movement, I chose to move in 20 ms steps.

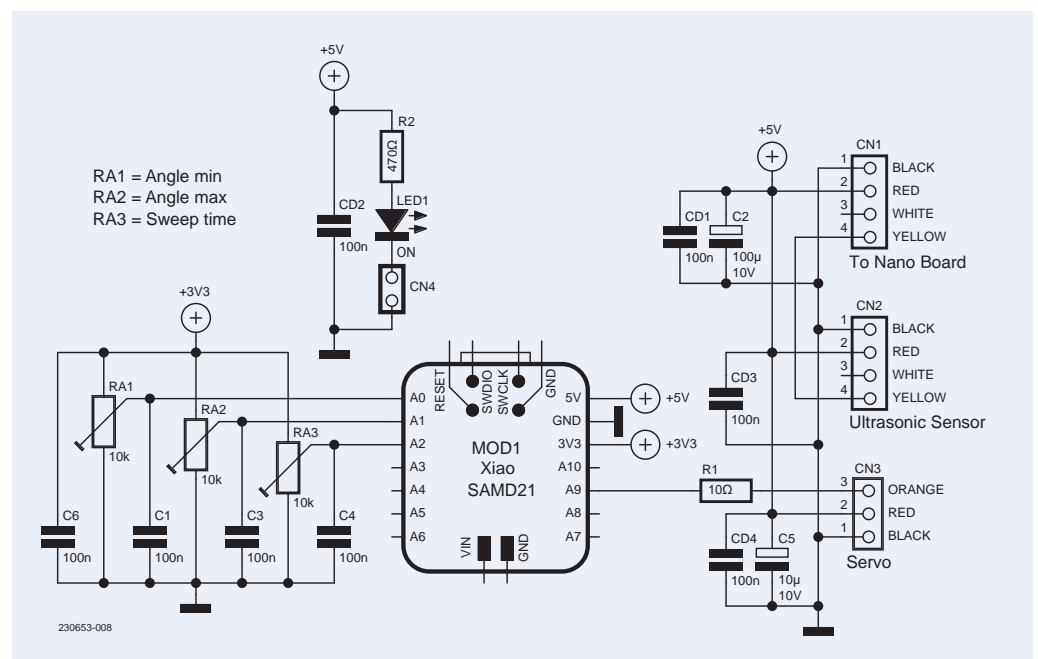


Figure 7: Schematic of the additional servo control module.



Listing 1: Snippet of the servo control code.

```
void smoothControl(int angleStart, int angleEnd, int duration)
{
    int steps = duration / 20; // Divide duration into 20 ms steps
    float increment = (angleEnd - angleStart) / (float)steps;
    float angle = angleStart;


    // Forward transition
    for (int i = 0; i <= steps; i++) {
        angle = angleStart + increment * i;
        monServo.write(angle);
        delay(20); // Wait 20 ms between each step
    }

    int durationForward = duration; // Use the calculated duration

    // Return transition with the same duration
    for (int i = steps; i >= 0; i--) {
        angle = angleStart + increment * i;
        monServo.write(angle);
        delay(durationForward / steps); // Use the same duration for the return transition
    }
}
```



Hence, the sweep duration (in milliseconds) is divided by 20 to give the number of steps needed. The angle the servo needs to move in each step is the total sweep angle `angleEnd - angleStart` divided by the number of steps. A for loop is used to go through all the angles, step by step. Then a similar loop is called, counting in reverse, for a smooth sweep in the opposite direction, as can be seen in **Listing 1**.

The sweeping sensor made it easier to detect obstacles on the sides, and turned out to be a useful addition. I'd like to think of this article as a small tribute to my late uncle, whose request started it all; I'm glad this project could help him regain a degree of independence. I hope this will inspire you with other ideas to help take care of your loved ones! Many thanks to Vincent, Bastian, and Cyril for their valuable help. 

230653-01

Questions or Comments?

Do you have technical questions or comments about this article? Feel free to contact the author at pascal.tours@gmail.com or Elektor at editor@elektor.com.

About the Author

Pascal Rondane's hobby has been electronics since he was a teenager, and he has owned every issue of Elektor since the beginning of the French edition in 1978. He trained as an electronics technician and worked for 20 years in a company that maintained Motorola radio equipment, and repaired electronic boards for IBM France. He then worked for 22 years in after-sales service and test bench design in a major French group manufacturing road signs. He's been retired for a year, which leaves him more time for his favorite hobby, as well as allowing him to take part in the activities of the *Association du Centre Historique de la Diffusion Radiophonique (ACHDR)*, dedicated to the safeguarding of the French audiovisual heritage.



Related Products

- > **Seeed Studio Grove Ultrasonic Distance Sensor**
www.elektor.com/20027
- > **Ultrasonic Radar Robot Kit**
www.elektor.com/20941

WEB LINKS

- [1] M5Stack Vibration Motor Unit: <https://shop.m5stack.com/products/vibration-motor-unit>
- [2] Pascal Rondane, "USB Battery Interface," Elektor Circuit Special 2024: <https://www.elektormagazine.com/230652-01>
- [3] XIAO SAMD21 by Seeed Studio: <https://tinyurl.com/yrtxyaza>
- [4] Downloads: <https://www.elektormagazine.com/labs/elektor-articles-software-downloads>



2025: An AI Odyssey

Mid-Year Review

By Brian Tristam Williams (Elektor)

From autonomous AI agents to multimodal systems that seamlessly blend text, image, and code, the first half of 2025 has transformed AI from experimental novelty to everyday reality. But as capabilities soar, questions about productivity gains and sustainability persist. What does this mean for makers and electronics enthusiasts?

As we begin the second half of 2025, the first half has brought remarkable developments in artificial intelligence, solidifying what many predicted would be a year of transition from experimentation to widespread practical implementation.

The pace of innovation has become so rapid that it's increasingly difficult to stay current with all the developments, so let's review some AI trends that have emerged so far this year and how they're impacting both the technology landscape and our community.

The Rise of Agentic AI

Perhaps the most transformative trend of 2025 has been the rise of agentic AI systems that can autonomously perform complex tasks with minimal human supervision. Google's Agentspace platform [1], launched

in January, has enabled enterprises such as Brazil's Banco BV to securely connect AI agents across critical systems and data sources while maintaining compliance. Meanwhile, IBM reports that organizations are increasingly deploying AI agents to handle workflows and routine tasks, though their research indicates that "most organizations aren't agent-ready" yet [2].

This shift toward agentic systems represents a fundamental evolution from last year's focus on content generation to autonomous reasoning and action. While the full capabilities promised by vendors aren't yet realized, the first half of 2025 has shown that AI can increasingly work independently on meaningful tasks rather than merely generating content under direct human guidance.

Multimodal Integration

The barriers between different types of data and processing have continued to dissolve in 2025. AI systems that seamlessly work across text, images, audio, and video are now standard rather than exceptional. According to PwC, "multimodal AI is revolutionizing product design and R&D processes" [3] in manufacturing, allowing engineers to rapidly generate and test designs that might have been overlooked through traditional methods.

Microsoft's expanded Copilot capabilities now include Copilot Daily, which provides personalized news and weather updates in your preferred voice each morning while respecting privacy and security [4]. This integration of AI across modalities and everyday activities reflects how AI is becoming more embedded in our daily routines, both professional and personal.

Global AI Competition Accelerates

The first half of 2025 has seen a dramatic narrowing of the performance gap between American and Chinese AI models. According to Stanford University's 2025 AI Index, the performance gap between top US and Chinese models has shrunk from 9.26% in January 2024 to just 1.70% by February

2025 [5]. This trend extends across various benchmarks for reasoning, mathematics, and coding capabilities.

This accelerating global competition has been accompanied by increased attention to semiconductor manufacturing and AI chip development. As organizations grapple with computational constraints, companies such as Nvidia have faced growing competition from alternative chip designs optimized for specific AI workloads.

The Productivity Reality Check

Despite the continued hype around AI's potential to revolutionize productivity, 2025 has brought a more nuanced assessment of real-world impact. While companies are implementing AI at scale, very few are "actually measuring productivity gains carefully or figuring out what the liberated knowledge workers are doing with their freed-up time," according to the MIT Sloan Management Review [6].

McKinsey's research reveals that despite significant investments, nearly half of C-suite leaders describe their AI initiatives as "still developing or expanding," suggesting the full economic benefits remain unrealized for many organizations. This aligns with what we've observed in the electronics and maker communities — AI tools are widely available, but integrating them effectively into existing workflows remains challenging.

Sustainability and Resource Constraints

An emerging theme in 2025 has been the growing recognition of AI's resource demands and the push for more sustainable practices. "AI requires so much energy that

there's not enough electricity (or computational power) for every company to deploy AI at scale," PwC notes [5]. This constraint has driven innovations in more efficient models and computing approaches.

Interestingly, AI is simultaneously becoming a key tool for addressing sustainability challenges. Microsoft reports that AI tools are helping address pressing global concerns such as climate change [4], while Google's AI-powered tools are reducing the time required for complex scientific research — decreasing both costs and environmental impact.

Coming Up in 2025

As we move into the second half of 2025, several developments appear on the horizon. The focus on measurement and optimization will likely intensify as organizations push to demonstrate concrete returns on their AI investments. Regulatory frameworks will continue to evolve, with significant variations between regions creating both challenges and opportunities for global companies.

For our Elektor community, the democratization of advanced AI capabilities presents exciting possibilities. The ability to deploy sophisticated AI on edge devices, including the latest generation of Raspberry Pi and specialized AI accelerators, puts powerful tools in the hands of hobbyists and small-scale developers that were unimaginable just a few years ago.

What AI developments have caught your attention in 2025? How are you incorporating these technologies into your projects? Let me know — you can find contact details in the **Questions or Comments?** box! ▶

230181-N-01

Questions or Comments?

If you have questions or comments, brian.williams@elektor.com is my email address. You can also catch me on Elektor Engineering Insights each month on YouTube, and you can find me @briantw on X.



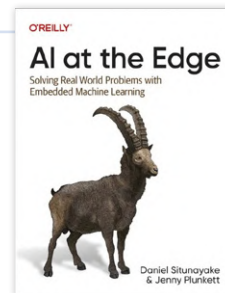
About the Author

Brian Tristam Williams has been fascinated with computers and electronics since he got his first "microcomputer" at age 10. His journey with Elektor Magazine began when he bought his first issue at 16, and since then, he's been following the world of electronics and computers, constantly exploring and learning. He started working at Elektor in 2010, and nowadays, he's keen on keeping up with the newest trends in tech, particularly focusing on AI and single-board computers such as Raspberry Pi.



Related Product

► **D. Situnayake, J. Plunket:**
AI at the Edge
(O'Reilly Media, 2023)
www.elektor.com/20465



WEB LINKS

- [1] Google: 2025 and the Next Chapter(s) of AI: <https://tinyurl.com/google2025ai>
- [2] IBM: AI agents in 2025: Expectations vs. reality: <https://tinyurl.com/ibmaiagents>
- [3] PwC: 2025 AI Business Predictions: <https://www.pwc.com/us/en/tech-effect/ai-analytics/ai-predictions.html>
- [4] Microsoft: 6 AI trends you'll see more of in 2025: <https://tinyurl.com/ibm2025ai>
- [5] IEEE Spectrum: 12 Graphs That Explain the State of AI in 2025: <https://spectrum.ieee.org/ai-index-2025>
- [6] MIT Sloan: Five Trends in AI and Data Science for 2025: <https://tinyurl.com/sloan2025ai>
- [7] McKinsey: Superagency in the workplace: Empowering people to unlock AI's full potential: <https://tinyurl.com/mckinseywpai>



Raspberry Pi Standalone MIDI Synthesizer (3)

Making It Smarter and Adding a User Interface

By Brian Tristam Williams (Elektor)

In Parts 1 and 2 of this series [1][2], we transformed a Raspberry Pi into a standalone MIDI synthesizer and added basic intelligent enhancements. Now, we'll take a leap forward by integrating actual AI capabilities and a user interface.

With the hardware for these experiments having been configured and plugged in, this final installment is quite code intensive. The actual code will still be available at the GitHub repository for this project [3].

Implementing Machine Learning Models for Music

Google's Magenta project [4] offers powerful open-source tools for music generation using machine learning. In Google's own words, it is "an open-source research project exploring the role of machine learning as a tool in the creative process." I highly recommend checking out their *Demos* page to get some creative inspiration.

Let's integrate some of these capabilities into our Raspberry Pi MIDI system.

Setting Up the Environment

First, we need to install the necessary libraries. Open a terminal on your Raspberry Pi and run:

```
sudo apt-get update
sudo apt-get install -y python3-pip libasound2-dev
libjack-jackd2-dev
pip3 install magenta
pip3 install python-rtmidi pip3 install tensorflow==2.4.0
```

Note that we're using a specific TensorFlow version that's compatible with the Raspberry Pi.

Loading Pre-Trained Models

Magenta provides several pre-trained models. For our purposes, we'll use a melody RNN model that can generate continuations of melodies we play:

```
# Create a directory for models
mkdir -p ~/magenta_models
cd ~/magenta_models
```

```
# Download a pre-trained melody RNN model
curl -O https://storage.googleapis.com/magentadata/
models/melody_rnn/basic_rnn.mag
```

Building the AI MIDI Bridge

Create a new Python script called *ai_accompaniment.py* (Listing 1) that will listen to your MIDI input, process it through the model, and output generated accompaniment.

This script listens to your MIDI input, collects it into a sequence, and after you pause playing, it generates a continuation of your melody. The AI-generated response is then played through your MIDI output port.

Creating a Musical Style Transfer System

Let's create another AI capability: transforming your playing into different musical styles using a simpler approach that's better suited for the Raspberry Pi's processing power.

Create a new file called *style_transform.py* (Listing 2). This script transforms your playing by applying stylistic elements like different scales, chord extensions, and swing feels based on the selected musical style. This happens in real-time in the following order:

1. When you play a note on your MIDI keyboard, the script intercepts it.
2. It applies style-specific modifications (adjusting velocity, mapping to the style's scale, potentially adding chord extensions).
3. It may adjust timing to create a swing feel.
4. It outputs the transformed note to the MIDI output.



Listing 1: ai_accompaniment.py

```
import magenta.music as mm
import tensorflow as tf
import threading
import numpy as np
import time
import rtmidi

from magenta.models.melody_rnn import melody_rnn_sequence_generator
from magenta.music import sequences_lib
from magenta.protobuf import generator_pb2
from magenta.protobuf import music_pb2

# Set up MIDI input and output
midi_in = rtmidi.MidiIn()
midi_out = rtmidi.MidiOut()
in_port = 0
out_port = 1
midi_in.open_port(in_port)
midi_out.open_port(out_port)

# Initialize the model
tf.compat.v1.disable_eager_execution()
bundle = mm.sequence_generator_bundle.read_bundle_file('~/.magenta_models/basic_rnn.mag')
generator_map = melody_rnn_sequence_generator.get_generator_map()
melody_rnn = generator_map['basic_rnn'](checkpoint=None, bundle=bundle)
melody_rnn.initialize()

# Buffer to store incoming MIDI notes
note_sequence = music_pb2.NoteSequence()
note_sequence.tempos.add(qpm=120)
current_notes = {}
last_note_time = 0

def generate_response(input_sequence):
    # Set generation parameters
    generator_options = generator_pb2.GeneratorOptions()
    generator_options.args['temperature'].float_value = 1.0 # Higher = more random
    generator_options.generate_sections.add(
        start_time=0,
        end_time=4.0 # Generate 4 seconds of music
    )

    # Generate the sequence
    generated_sequence = melody_rnn.generate(input_sequence, generator_options)

    # Convert the generated sequence to MIDI messages and play
    for note in generated_sequence.notes:
        if note.start_time >= input_sequence.total_time: # Only play the new notes
            # Calculate when to play this note
            delta_time = note.start_time - input_sequence.total_time
            time.sleep(delta_time)

            # Send note-on message
            midi_out.send_message([0x90, note.pitch, note.velocity])

            # Calculate note duration
            duration = note.end_time - note.start_time
            time.sleep(duration)
```

continued on next page


```

        # Send note-off message
        midi_out.send_message([0x80, note.pitch, 0])

def midi_callback():
    global last_note_time
    while True:
        message = midi_in.get_message()

        if message:
            midi_msg, delta_time = message

            # Process note-on messages
            if midi_msg[0] & 0xF0 == 0x90 and midi_msg[2] > 0: # Note on with velocity > 0
                note = midi_msg[1]
                velocity = midi_msg[2]
                current_time = time.time()
                last_note_time = current_time

                # Add note to sequence
                note_sequence.notes.add(
                    pitch=note,
                    velocity=velocity,
                    start_time=current_time - note_sequence.total_time,
                    end_time=current_time - note_sequence.total_time + 0.5, # Default duration
                    instrument=0,
                    program=0
                )
                current_notes[note] = len(note_sequence.notes) - 1

            # Process note-off messages
            elif (midi_msg[0] & 0xF0 == 0x80) or (midi_msg[0] & 0xF0 == 0x90 and midi_msg[2] == 0):
                note = midi_msg[1]
                if note in current_notes:
                    note_idx = current_notes[note]
                    note_sequence.notes[note_idx].end_time = time.time() - note_sequence.total_time
                    del current_notes[note]

            # If we've collected enough notes and there's a pause, generate a response
            if len(note_sequence.notes) >= 8 and time.time() - last_note_time > 1.0:
                # Update the total time of the sequence
                note_sequence.total_time = time.time() - note_sequence.total_time

                # Generate a response in a separate thread
                threading.Thread(target=generate_response, args=(note_sequence,)).start()

                # Reset for next input
                note_sequence = music_pb2.NoteSequence()
                note_sequence.tempos.add(qpm=120)
                current_notes = {}

        time.sleep(0.001) # Small sleep to prevent CPU hogging

# Start the MIDI callback in a separate thread
midi_thread = threading.Thread(target=midi_callback)
midi_thread.daemon = True
midi_thread.start()

try:
    print("AI Accompaniment running. Play your MIDI keyboard to generate responses.")
    print("Press Ctrl+C to exit.")
    while True:
        time.sleep(1)

```

continued on next page

```
except KeyboardInterrupt:
    print("Exiting...")
    midi_in.close_port()
    midi_out.close_port()
```



Listing 2: style_transform.py

```
import rtmidi
import time
import random
import numpy as np

# MIDI setup
midi_in = rtmidi.MidiIn()
midi_out = rtmidi.MidiOut()
midi_in.open_port(0)
midi_out.open_port(1)

# Style definitions
STYLES = {
    'jazz': {
        'chord_extensions': [9, 11, 13], # 9th, 11th, 13th extensions
        'swing_ratio': 0.67, # Swing feel
        'velocity_variance': 15, # More dynamic expression
        'scale': [0, 2, 3, 5, 7, 9, 10] # Dorian scale (for jazz feel)
    },
    'classical': {
        'chord_extensions': [7], # Simpler harmonies
        'swing_ratio': 0.5, # No swing
        'velocity_variance': 10, # Moderate dynamics
        'scale': [0, 2, 4, 5, 7, 9, 11] # Major scale
    },
    'electronic': {
        'chord_extensions': [4, 7], # Power chords and octaves
        'swing_ratio': 0.5, # Straight timing
        'velocity_variance': 5, # Less dynamic variance
        'scale': [0, 2, 4, 7, 9] # Pentatonic scale
    }
}

# Current style
current_style = 'jazz'

# Note tracking
active_notes = {}
note_history = []
last_timestamp = time.time()
beat_interval = 0.5 # 120 BPM (in seconds)
def apply_style_to_note(note, velocity):
    """Apply the current style's characteristics to a MIDI note"""
    style = STYLES[current_style]

    # Add velocity variance
    new_velocity = min(127, max(1, velocity + random.randint(-style['velocity_variance'],
                                                                style['velocity_variance'])))

    # Map to scale if needed
    scale = style['scale']
```

continued on next page

```

octave = note // 12
pitch_class = note % 12
closest_scale_pitch = min(scale, key=lambda x: abs(x - pitch_class))
new_note = (octave * 12) + closest_scale_pitch

# Occasionally add a chord extension
if random.random() < 0.3: # 30% chance
    extension = random.choice(style['chord_extensions'])
    extension_note = new_note + extension
    if 0 <= extension_note <= 127:
        midi_out.send_message([0x90, extension_note, new_velocity // 2])
        active_notes[extension_note] = time.time()

return new_note, new_velocity

def adjust_timing(timestamp):
    """Apply swing feel based on current style"""
    style = STYLES[current_style]
    global last_timestamp, beat_interval

    # Calculate position in beat
    beat_position = (timestamp - last_timestamp) % beat_interval
    beat_fraction = beat_position / beat_interval

    # Apply swing if on off-beat
    if 0.25 < beat_fraction < 0.5:
        swing_adjustment = beat_interval * (style['swing_ratio'] - 0.5) * 0.5
        return timestamp + swing_adjustment

    return timestamp

def process_midi():
    """Process incoming MIDI messages and apply style transformations"""
    while True:
        msg = midi_in.get_message()

        if msg:
            message, timestamp = msg
            command = message[0] & 0xF0

            # Note on
            if command == 0x90 and message[2] > 0:
                note, velocity = message[1], message[2]
                styled_note, styled_velocity = apply_style_to_note(note, velocity)
                styled_time = adjust_timing(timestamp)

                # Small delay to implement swing if needed
                time_diff = styled_time - timestamp
                if time_diff > 0:
                    time.sleep(time_diff)

                midi_out.send_message([0x90, styled_note, styled_velocity])
                active_notes[styled_note] = time.time()
                note_history.append(styled_note)

                # Keep history manageable
                if len(note_history) > 32:
                    note_history.pop(0)

            # Note off

```

continued on next page


```

        elif command == 0x80 or (command == 0x90 and message[2] == 0):
            note = message[1]
            # Find any transformed version of this note
            for active_note in list(active_notes.keys()):
                if abs(active_note - note) <= 2: # Allow for slight pitch mapping
                    midi_out.send_message([0x80, active_note, 0])
                    if active_note in active_notes:
                        del active_notes[active_note]

            # Other MIDI messages (pass through)
            else:
                midi_out.send_message(message)

            # Release hung notes after 5 seconds
            current_time = time.time()
            for note in list(active_notes.keys()):
                if current_time - active_notes[note] > 5.0:
                    midi_out.send_message([0x80, note, 0])
                    del active_notes[note]

            time.sleep(0.001)

def cycle_style():
    """Change to the next available style"""
    global current_style
    styles = list(STYLES.keys())
    current_index = styles.index(current_style)
    next_index = (current_index + 1) % len(styles)
    current_style = styles[next_index]
    print(f"Style changed to: ")

# Set up a separate thread for button handling if using the Pirate Audio buttons
try:
    import RPi.GPIO as GPIO

    # Pirate Audio button pins
    BUTTON_A = 5
    BUTTON_B = 6
    BUTTON_X = 16
    BUTTON_Y = 24

    GPIO.setmode(GPIO.BCM)
    GPIO.setup(BUTTON_A, GPIO.IN, pull_up_down=GPIO.PUD_UP)
    GPIO.setup(BUTTON_B, GPIO.IN, pull_up_down=GPIO.PUD_UP)
    GPIO.setup(BUTTON_X, GPIO.IN, pull_up_down=GPIO.PUD_UP)
    GPIO.setup(BUTTON_Y, GPIO.IN, pull_up_down=GPIO.PUD_UP)

    def button_callback(channel):
        if channel == BUTTON_A:
            cycle_style()

    GPIO.add_event_detect(BUTTON_A, GPIO.FALLING, callback=button_callback, bouncetime=300)

    print("GPIO buttons configured. Press Button A to cycle through styles.")
except:
    print("GPIO setup failed. Button functionality not available.")
if __name__ == "__main__":
    print(f"Style Transformer running. Current style: ")
    print("Play your MIDI keyboard to hear the transformation.")
    print("Press Ctrl+C to exit.")

```

continued on next page

```
try:
    process_midi()
except KeyboardInterrupt:
    print("Exiting...")
    midi_in.close_port()
    midi_out.close_port()
try:
    GPIO.cleanup()
except:
    pass
```

Creating an Interactive User Interface

Let's utilize the Pimoroni Pirate Audio's ST7789 display to show the current AI mode and provide visual feedback. Of course, any other display may be utilized with suitable code adjustments, but I happen to have the Pirate Audio one (**Figure 1**), for which the `midi_display.py` is at [3].

This script creates a user interface on the Pirate Audio display showing the current AI mode, MIDI activity, and button functions. The Pirate Audio board has four buttons (A, B, X, Y) in this implementation:

- Button A: Cycles through modes on the main screen, navigates up in menus, increases values when editing settings.
- Button B: Opens settings menu on the main screen, navigates down in menus, decreases values when editing settings.
- Button X: Save functionality (saves presets), edits selected settings, confirms selections.
- Button Y: Load functionality (loads presets), navigates back, cancels editing.

Button A cycles through the available modes (FluidSynth, AI Melody, Jazz Style, Classical, Electronic). The interface shows the current mode and status on the display.

Combining Everything with an Auto-Start Script

Let's create a master script that will start our entire MIDI AI system at boot. Create a file called `start_midi_ai.sh` (**Listing 3**).



Figure 1: I decided to switch up from the Raspberry Pi Zero 2 W to a beefier Raspberry Pi 5 with 16 GB of RAM.

Using crontab make this script executable and set it to run at startup. Crontab (short for "cron table") is a time-based job scheduler in Unix-like operating systems, including the Raspberry Pi's Linux distribution. It allows users to schedule commands or scripts to run automatically at specified times or intervals.

Use this line to enter crontab:

```
chmod +x ~/start_midi_ai.sh crontab -e
```

And then add this line to the crontab:

```
@reboot /home/pi/start_midi_ai.sh &
```

Finally, exit the nano editor by using `Ctrl-X`, `Y`, then `Enter`.



Listing 3: start_midi_ai.sh

```
#!/bin/bash

# Start FluidSynth as a service
systemctl restart fluidsynth

# Connect MIDI devices after a short delay
sleep 5
aconnect 24:0 128:0 # Adjust these numbers to match your hardware

# Start the display interface
python3 /home/pi/midi_display.py &

# Log startup
echo "MIDI AI System started at $(date)" >> /home/pi/midi_ai.log
```



Performance Optimization for AI Models

Running machine learning models on the Raspberry Pi can be resource-intensive. Here are some tips to optimize performance:

- 1. Use TensorFlow Lite:** Instead of the implementation in this project — especially for lower-power devices — we could use TensorFlow Lite, as it's designed for edge devices.
- 2. Quantize TensorFlow models:** This reduces model size and speeds up inference.

```
# Add to your Python setup script
converter = tf.lite.TFLiteConverter.
    from_saved_model(saved_model_dir)
converter.optimizations = [tf.lite.Optimize.DEFAULT]
quantized_tflite_model = converter.convert()
```

- 3. Reduce model complexity:** When using Magenta models, opt for smaller network configurations to decrease computational demands and memory usage, especially on resource-constrained devices like the Raspberry Pi. For Magenta network configurations, the best concise explainer is probably the Melody RNN model config page in the Magenta repo at [5].

- 4. Offload computation:** If we really want to branch out to more complex tasks, we could consider offloading to a more powerful computer on our network, but the Raspberry Pi 5 with 16 GB RAM works well enough for me — for now.

Final Thoughts

We've now transformed our simple MIDI setup into an AI-powered musical companion. The system can:

- Respond to our playing with generated melodies
- Transform our input into different musical styles
- Provide visual feedback on the Pirate Audio display
- Start automatically at boot

As computing power on edge devices continues to improve, and as AI models become more efficient, we'll be able to run increasingly sophisticated musical intelligence directly on edge devices such as the Raspberry Pi. Future enhancements could include full band accompaniment (perhaps even on an array of Raspberry Pis!), learning from your specific playing style, or even voice-controlled music generation. ◀

240714-C-01

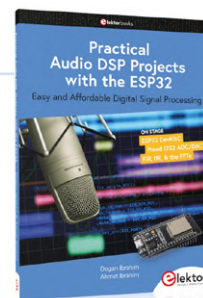
Questions or Comments?

If you have questions or comments, email me at brian.williams@elektor.com. You can also catch me on Elektor Engineering Insights each month on YouTube, and you can find me @briantw on X.



Related Products

- G. Spanner, *Machine Learning with Python for PC, Raspberry Pi, and Maixduino* (E-book, Elektor, 2022) www.elektor.com/20150
- D. Ibrahim, A. Ibrahim, *Practical Audio DSP Projects with the ESP32* (Elektor, 2023) www.elektor.com/20558



elektor TV

See the project
in action here!



WEB LINKS

- [1] Brian Tristam Williams, "Raspberry Pi Standalone MIDI Synthesizer (1)," Elektor 3-4/2025: <https://elektormagazine.com/240714-01>
- [2] Brian Tristam Williams, "Raspberry Pi Standalone MIDI Synthesizer (2)," Elektor 5-6/2025: <https://elektormagazine.com/240714-B-01>
- [3] GitHub repository for these experiments: <https://github.com/briantw/pi-midi>
- [4] Magenta Project: <https://magenta.tensorflow.org/>
- [5] Magenta's Melody RNN README.md — GitHub repository: <https://tinyurl.com/melodyreadme>

Meshtastic: A Demo Project

An Intelligent Mesh of LoRa Radios

By Bera Somnath (India)

Despite their scalability, long range, low power consumption and security with data encryption, LoRa modules were not very suitable for mesh networking. That was before the arrival of Meshtastic, an open-source platform that filled this gap. Meshtastic allows a very effective mesh network to be created with LoRa modules, and most importantly, at low cost!

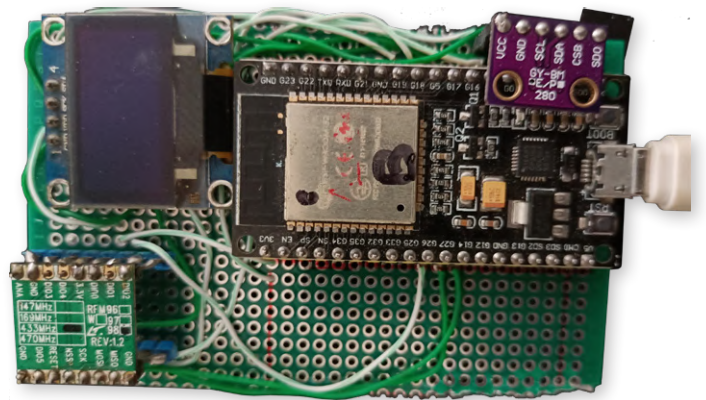


Figure 2: The finished prototype of the homemade LoRa board, ready for firmware flashing.

Ever since Long Range (LoRa) radios hit the market, lots of IoT devices are appearing in the market now and then. Besides LoRa, there is one more standard which is already occupying the market segment with various applications, and that is ZigBee. Although ZigBee devices are substantially costly, still they are in use due to their exceptionally easy meshing capability. ZigBee radios can be placed in mesh configurations, where every ZigBee will be connected as a node with each other (Figure 1), such that if one or more nodes fail, the network will still be available.



Figure 1: A typical mesh network. (Source: Adobe Stock)

Whilst the ZigBee network is highly scalable, the LoRa network is not lagging much behind. Besides, LoRa has a very long range, low-power consumption, highly encrypted and is immune to electromagnetic interferences.

Meshtastic Network

Meshtastic [1] has brought an open-source software to create a LoRa mesh network. LoRa works great on Line of Sight (LOS). Therefore, if one or two nodes can be placed at a strategically high place, then all nodes will easily talk to each other. There are already many works available on the Internet. Here, I just tried to create a practical working network for readers, where sensor data, GPS position and triggering alarms can be easily worked out.

The Project

To make a workable mesh network for testing, we need to build three or more LoRa nodes, with an ESP32 board and a LoRa module (Figure 2), or purchase ready-made ESP32 LoRa boards from the market, for example the WiFi LoRa 32 [2] from Heltec (Figure 3). Furthermore, at least one I²C sensor from the following list has to be procured:

- > BME280
- > BME680
- > MCP9808



Figure 3: An "off-the-shelf" Heltec LoRa module.

- > INA260
- > INA219
- > SHTC3
- > SHT31
- > PMSA0031

Wiring Diagram

The wiring needed to link all the modules included in this project is shown in **Figure 4**. To flash an ESP32 board with Meshtastic software (see also the textbox **Compatible Boards**), connect the board to your computer's USB port and then go to the Meshtastic online flasher

Modules Needed

- > Heltec WiFi LoRa 32 with I²C OLED & SPI LoRa (3 sets)
- > BME280 (Temperature, Pressure, Humidity) -or-
- > MCP9808 (Temperature)
- > u-blox NEO-6M or NEO-7M GPS Sensor (1 piece)

Instead of the ready-made Heltec LoRa module, you can also buy these modules separately:

- > ESP32 Dev Module - Google: esp32-38pin-development-board-wifi-bluetooth-ultra-low-power-consumption-dual-core
- > I²C OLED 0.96 Inch - Google: 0-96-inch-yellow-yellow-blue-oled-lcd-led-display-module
- > SX1278 868 MHz LoRa Radio - Google: lora-module-rfm95w-868s2r

webpage [3]. This tool will only work in Chrome or Edge browsers. However, the command line interface (CLI) method of uploading does not have this limitation.

The website with the Meshtastic flashing tool described above does not provide flash facility for Heltec V1 or V2 boards. It starts working with the Heltec V3 board upwards (and for higher level boards like T-Beam, T-Echo, T-Deck, etc.). The flash software *firmware-heltec-v1-2.2.12.092e6f2.bin* available from [4] will flash the Heltec LoRa board, or the finished prototype of a ESP32 LoRa homemade board shown in Figure 2.

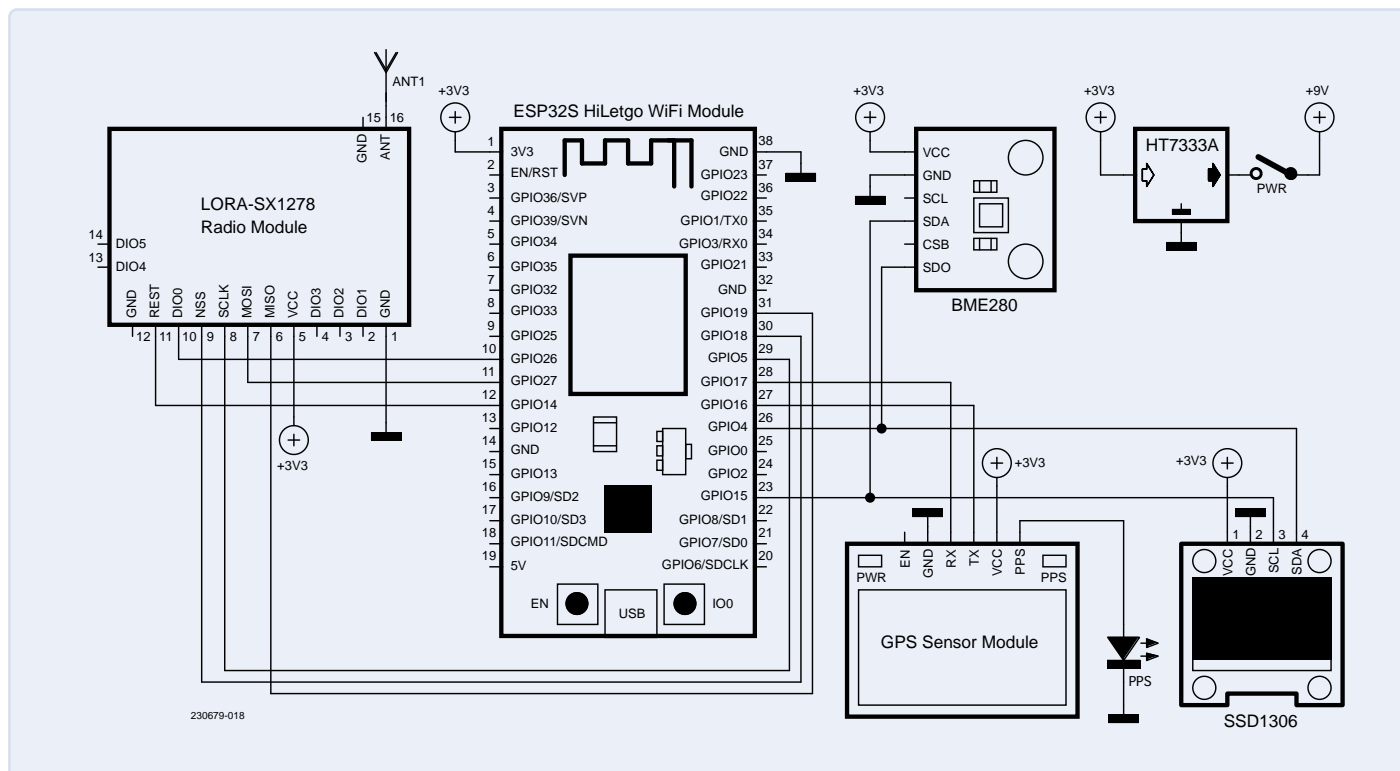


Figure 4: Schematic: A generic GPS sensor (see text) is added. The PPS (Pulse Per Second) output signal is routed to an LED.

On the first screen (**Figure 5a**), select the firmware version you want to flash (*Heltec V1*). Then, on the next screen (**Figure 5b**), select the CP2102 USB to UART bridge — this is the interface hardware that sits between the MCU and the computer. Next, follow the instructions that appear on the screen — like the warning about erasing the entire device, etc. Ensure the Internet connection is stable during this crucial 4 to 5 minutes period. For the original Heltec board (V3 hardware version), you may need to hold down the *boot* button for the entire 4 to 5 minutes of the upload phase.

Uploading Software Through the Command Line

I'm providing this option as well, since I've found that the browser-based firmware update doesn't work reliably. Many times it indicates that it has completed the task rapidly, but actually, it did not. It takes about 4 to 5 minutes to install/update the firmware. That's why I have provided the firmware and the CLI method in the download [4]. Also note that the software will probably evolve many times from now on. Therefore, having a stable copy around may be useful during a later setup.

MacOS commands/Linux commands:

```
pip3 install --upgrade esptool
// upgrade esptool first
esptool.py chip_id
// you will get a long output with chip id and chip
// mac id
cd ~/Downloads/firmware/
// firmware-heltec-v1-2.2.12.092e6f2.bin
./device-install.sh -f firmware-heltec-v1-2.2.12.092e6f2.
bin
// for installing
./device-update.sh -f firmware-heltec-v1-2.2.12.092e6f2.
bin
// for updating
```

Compatible Boards

The Meshtastic site is evolving continuously. When this project was developed about two years back, it was only ESP32 varieties of boards where Meshtastic was working. Today the flasher [3] knows a variety of boards where the software can be installed, including boards from RAK Wireless, Seeed, Nordic (nRF52) and Raspberry Pi (Pico). The tool has even an auto-detect button to automatically discover the board.

Windows commands:

```
device-install.bat -f firmware-heltec-v1-2.2.12.092e6f2.
bin
// for installing
device-update.bat -f firmware-heltec-v1-2.2.12.092e6f2-
update.bin
// for updating
```

Project Set Up and Testing

To set up and test the project, we need to install the Meshtastic app on our Android phone, iPhone or tablet. There are both Android and iOS versions available in the respective software stores. For three nodes to work with, I would request to fix one mobile/tab/Apple computer for each one node to play with. Following this way, you will actually avoid many setup jolts, which I learned the hard way. The application is absolutely free, so no problem!

First, power on the LoRa node. Next, on your mobile/tablet/Apple/PC, switch on the Bluetooth functionality and pair your device with the LoRa node. The six-digit pairing number will appear on the OLED screen. Once connected, open the Meshtastic app, and you will see that the app is connected (look for the check mark at the top right) to the board with the board's MAC address. You can set a name for the board in the wireless configuration by pressing the three dots on the right top of the app.

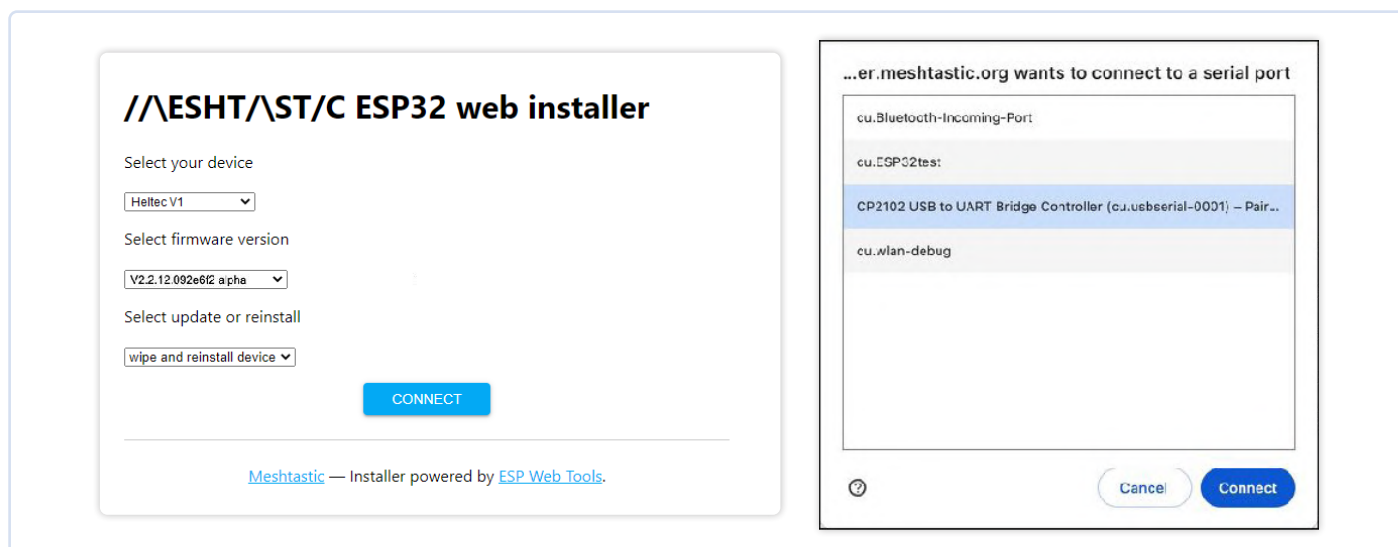


Figure 5a and 5b: First steps to install the Meshtastic software on each ESP32 board.

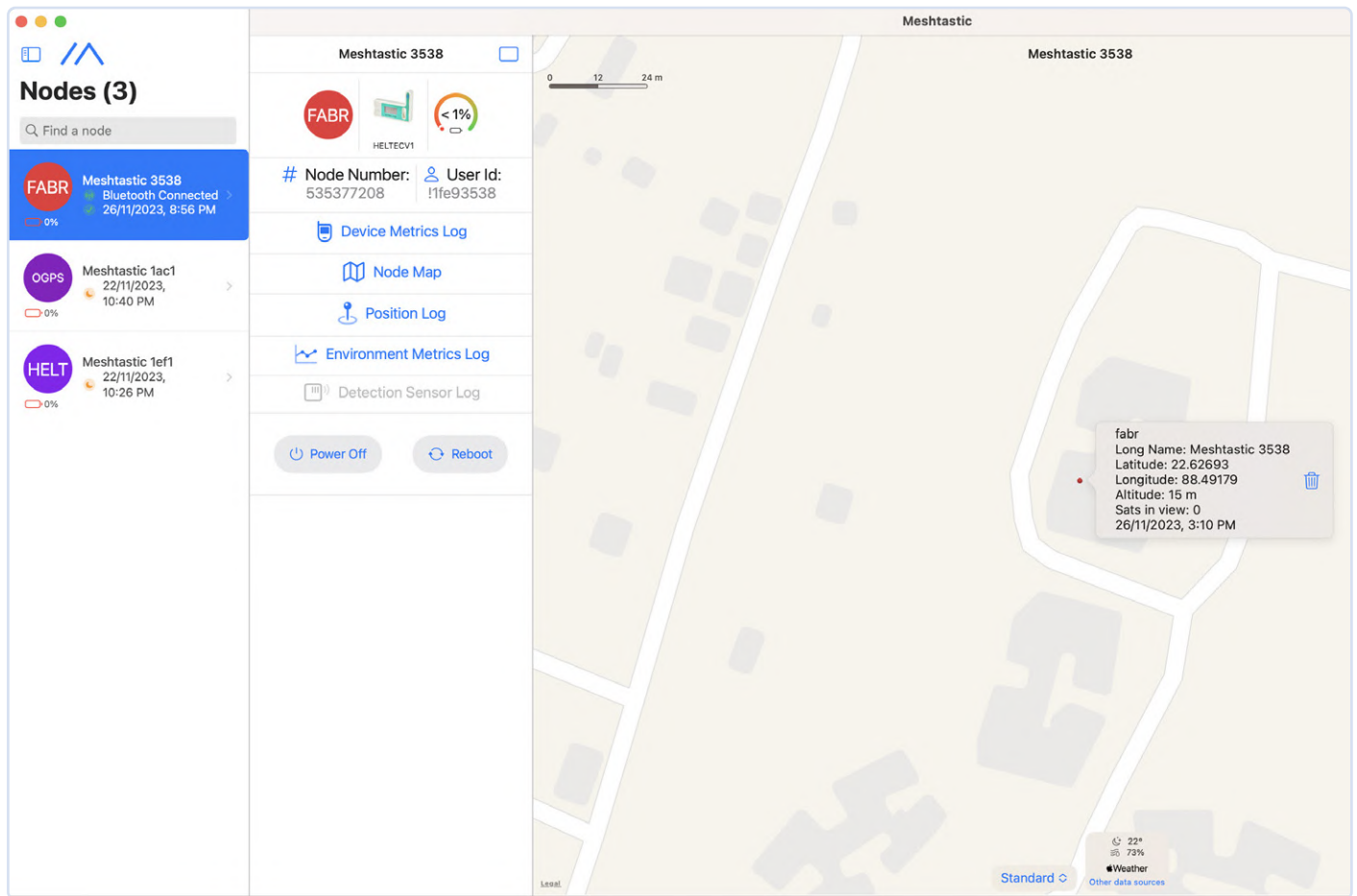


Figure 6: Homemade Heltec-like LoRa board *fabr* is now operational and connected. The other two nodes are still inactive.

Attaching Sensors

Only I²C sensors with the following addresses are allowed: 0x76, 0x77, 0x78, 0x18, 0x40, 0x41, 0x5D, 0x5C, 0x70, 0x44, 0x12, and 0x3C (fixed for OLED). More sensors will hopefully be added in the future. To enable detection, navigate to *Radio Configuration* → *Detection Sensor* → *Detection Sensor enabled*.

You may provide a name for the sensor, such as *BME* or *Temp_Ind*. In the same window, you can also designate a GPIO pin to monitor for high or low signals. The detection will be monitored periodically every 900 s (default value) for high or low signals. Please note that this GPIO pin status is entirely independent of the sensor value.

The homemade Heltec-like node is now up and running (**Figure 6**). I named it *fabr* (short for Fabricated One). The other two devices are not operational yet. **Figure 7** shows a screenshot of the Meshtastic app.

Role Configuration

In *Radio Configuration* → *Device* → *Role*, you can assign the role for transmitting sensor values:

- **SENSOR:** Used for devices that gather and send sensor data. Optimized for low-power consumption and periodic data transmission. These are strategically used for remote sensor data on solar panels or battery power.
- **CLIENT:** Used for end-user devices that primarily consume and display data. These devices can also send commands but do not relay messages within the network.

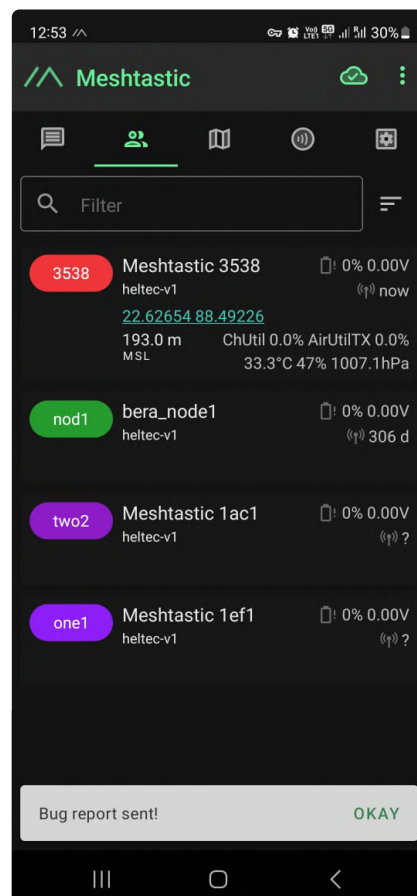


Figure 7: Screenshot of the Meshtastic app. The LoRa node at the top is up. The two at the bottom are not up. That's why question marks are appearing against their names.

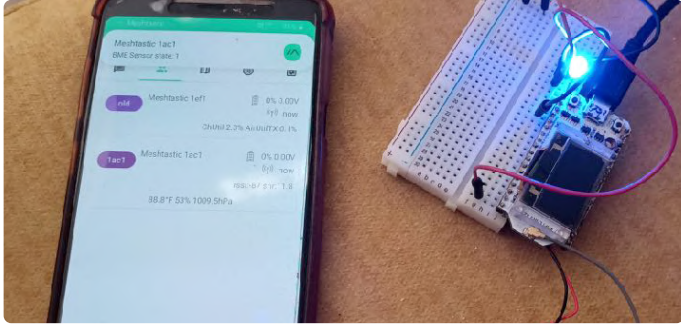


Figure 8: LED alarm fires up! On the app display, the sensor values are also visible.

- **ROUTER_CLIENT:** Acts to relay messages across the network. Typically configured with higher power settings and kept stationary to maintain network stability and reach.
- **REPEATER:** Acts as a signal booster, enhancing the range of the mesh network by boosting the signal. Strategically placed to cover larger areas or to connect distant nodes that are out of direct communication range. Like routers, they typically do not sleep to ensure consistent signal boosting.

Remote Hardware

This section allows you to send signals from remote sensors. You must first define a GPIO pin. However, only digital read and digital write functions are enabled here.

External Notifications

To enable external notifications, navigate to *Radio Configuration* → *External Notification* → *Enable*. You also need to set the output GPIO pins that will receive the notifications. You can assign an LED, alarm buzzer, or alarm vibrator to three different GPIO pins, along with the duration of the alarm. In my *fabr* unit, I have set the alarm on GPIO16 to light an LED for 2000 ms. The sensor values will appear on the Android/iOS screen and on the OLED screen (**Figure 8**).

Adding a New Node

There is no special rule to set up a LoRa node. Just set up the node, power it on and it will be joined into the network.

GPS Location

If the board has a GPS locator (**Figure 9**), it will deliver the location automatically. Otherwise, in case GPS is not fitted, if the Android/iOS device has location information turned on, it will provide its own GPS location, which will be visible and used for that node in the app window. The app has the facility to attach a separate GPS sensor, for which you must manually set the GPIO pins for TX and RX in *Radio Configuration* → *Position*. After setting the GPS sensor, the location data (latitude and longitude) will come from the GPS sensor. Currently, only u-blox and GLONASS sensors are supported. My old VKEL-GPS sensor also worked when connected. The PPS (pulse per second) is connected to an LED, which can be disabled to preserve battery life.

Wi-Fi Network Connection

Navigate to *Radio Configuration* → *Network* → *Enable WiFi*. Enter the SSID and PSK for your network. Note: By doing this, your node will be inaccessible via Bluetooth. You will have to access it from now on through Wi-Fi or via the command line using Meshtastic, as explained below (note: this method is not very user-friendly).

To control a node via Wi-Fi, you need to know the IP address of the node. Follow these steps:

- Go to your Wi-Fi DHCP list.
- Find the IP address and port of the node (the port is usually *nil*, the default).
- From the command line of your Meshtastic installation (on PC or Mac, see below), type the IP you've just found and a message that you want to be broadcasted by your node.

```
$> meshtastic --host <device-ip> --info
$> meshtastic --host <device-ip> --sendtext "Your message here"
```

For reversing the configuration, simply reinstall the firmware. Despite trying various methods, reinstalling the firmware is the only solution that consistently works.

Store & Forward Configuration

To enable *Store & Forward*, navigate to *Radio Configuration* → *Module Configuration* → *Store & Forward*. This feature is only available for ESP32. It allows you to store a few records in the event of disconnection from a node. However, please note that Meshtastic does not normally recommend using this feature.

Serial Terminal

I absolutely love the serial terminal! If you have sensors that are not I²C type, you can attach them to an Arduino and send the data over the Serial port into the serial terminal of a Meshtastic node. The data will be read and transmitted over the nodes. Additionally, you can attach a serial monitor on two GPIO pins to monitor the serial data output directly. I have created an "ESP32 Serial Terminal Viewer," which you can try. You can find it on Elektor Labs [5].



Figure 9: The prototype with ESP32, LoRa and OLED screen. GPS module and BME280 sensor are added.

Channel Configuration

At some point, you may need to separate your network from other networks operating on the same frequency band. Meshtastic networks can be secured by setting a custom channel key, a shared secret 32-byte key used to encrypt messages. Only devices with the same channel key can join and communicate within the network. To configure this, navigate to *Radio Configuration* → *Channel Configuration*; enter your secret key and repeat it for all nodes. Instead of a secret key, you can also set a unique channel name to distinguish your network from others.

The secret key should be a 32-byte encoded key. There is a small circular arrow on the right of the *Channel Key* window. Pressing it will generate a random 32-byte encrypted channel key. Alternatively, you can use a small Python script (*Channel-key.py*) to create your own channel key.

```
import base64
key = "abcdefghijklmnopqrstuvwxyz123456"
encoded_key = base64.b64encode(key.encode()).decode()
print(encoded_key)
```

Command Line Meshtastic

The app is convenient for many users. But, as I previously said, there is also a CLI for Meshtastic to work with the hardware. Connect your device to the USB port to get started.


```
$> pip install --upgrade pip [or pip3]
$> pip install meshtastic [or pip3]
$> meshtastic --version
$> meshtastic --help
$> meshtastic --port /dev/usbserial-xxxx --info
    [usbserial-xxx is the port number ]
$> meshtastic --setkey abcdefghijklmnopqrstuvwxyz123456
    [to be encrypted channel key]
$> meshtastic --setch-name "S_Bera_SecureMesh"
    [Set the channel name]
```

Typical Use

In a forest area, a LoRa mesh network can be used to monitor forest fires, with each node tracking air temperature, wind velocity, and humidity in its zone. In an industrial setting, this technology is effective for monitoring fire hazards. In a city, turning all vehicles into LoRa nodes allows for monitoring vehicle speed and position. For a township or housing complex, V53LOX laser sensors can be employed to monitor

water tank levels, and operating lifts equipped as LoRa nodes can be overseen from a central monitoring center. Additionally, Meshtastic LoRa networking enables communication with the nodes and can utilize MQTT to extend connectivity to other cities.

Benefits and Fun

The Meshtastic software demonstrates how a LoRa mesh can deploy LoRa radios to achieve the best networking benefits, such as low power consumption, great range, and security. It's fun to experiment with the network. Once the nodes are set up, you can disconnect the Android or iOS devices, and the nodes will continue to operate automatically. 

230679-01

Questions or Comments?

Do you have technical questions or comments about this article? Feel free to contact the author by email at berasomnath@gmail.com or the Elektor editorial team at editor@elektor.com.



About the Author

Somnath Bera, a mechanical engineer from Jalpaiguri Govt. Engg. College, India, worked for 36.5 years at different capacity at NTPC, the largest power producer in the country. He has a profound passion for electronics, evidenced by his 60+ innovative projects on Elektor Labs, over 12 of which have been featured in Elektor. His projects are often focused on problem-solving in areas like waste and natural resource management. Somnath likes to use innovative approaches and platforms like Arduino, Raspberry Pi, and ESP32 coupled with various kinds of sensors and wireless systems to create efficient and cost-effective solutions.



Related Products

- > **Claus Kühnel, *Develop and Operate Your LoRaWAN IoT Nodes* (Elektor, 2022)**
www.elektor.com/20147
- > **0.96" OLED Display for Arduino (128x64)**
www.elektor.com/18004

WEB LINKS

- [1] Meshtastic website: <https://meshtastic.org/>
- [2] WiFi LoRa 32(V3), ESP32S3 + SX1262 LoRa Node, Heltec Automation: <https://heltec.org/project/wifi-lora-32-v3/>
- [3] Meshtastic flasher: <https://flasher.meshtastic.org>
- [4] Software download at Elektor Labs: <https://tinyurl.com/2psrnvdv>
- [5] "ESP32 Serial Terminal" at Elektor Labs: <https://tinyurl.com/95rak5up>

Upgraded T-962 v2.0 Reflow Soldering Oven (Elektor Version)

This upgraded version 2.0 (available exclusively from Elektor) contains the following improvements:

- Enhanced protective earthing (PE) for furnace chassis
- Extra thermal insulation layer around furnace to reduce odors
- Connection to a computer, allowing curve editing on a PC



Price: ~~€279.00~~

Special Price: €239.00

www.elektor.com/20346

SEQUIRE HT140 (2-in-1) SMD Soldering & Desoldering Tweezers

The SEQUIRE HT140 is a highly versatile 2-in-1 soldering tool that combines the functionality of hot tweezers and a soldering iron. It is specifically engineered for precise SMD soldering and desoldering tasks.



Price: ~~€119.95~~

Special Price: €99.95

www.elektor.com/21167

The Arduino Collection (USB Stick)

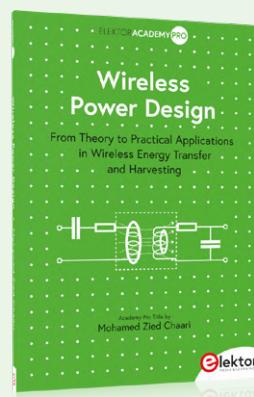


Price: ~~€49.95~~

Special Price: €39.95

www.elektor.com/21181

Wireless Power Design



Price: €39.95

Member Price: €35.96

www.elektor.com/21177



Geekworm KVM-A3 Kit for Raspberry Pi 4 (+ FREE E-book)



Price: ~~€119.95~~

Special Price: €89.95

www.elektor.com/21139

Elecrow All-in-One Starter Kit for Raspberry Pi Pico 2



Price: €49.95

Member Price: €44.96

www.elektor.com/21144

FNIRSI DSO-TC4 (3-in-1) Oscilloscope (10 MHz) + Transistor Tester + Signal Generator



Price: ~~€89.95~~

Special Price: €69.95

www.elektor.com/21146

Andonstar AD210 10.1" Digital Microscope



Price: ~~€179.95~~

Special Price: €99.95

www.elektor.com/20802

Analog Audio Frequency Generator

High-Quality Adjustable Frequency Sine Wave Generator

By Alfred Rosenkränzer (Germany)

Last year, we published a high-quality variable-frequency audio notch filter. To complete a typical test lab setup for measuring audio equipment performance, this article introduces a high-performance audio frequency generator with continuously adjustable output frequency control. The key component in both these designs is something called a State Variable Filter.

A *State Variable Filter* is at the core of this design and consists of analog integrators together with a carefully controlled feedback coupling to create a band-pass filter with a very high quality (Q) factor. The level of performance achieved by this filter would be difficult to attain using alternative, more basic circuit designs. The band-pass filter is made to oscillate by positive feedback. By combining this with active amplitude control in the feedback loop, we can build a high-quality, low-distortion adjustable audio frequency generator.

The Block Diagram

The band-pass filter, shown in the block diagram of the adjustable audio tone generator (**Figure 1**), shows that its passband frequency can be switched between three ranges using a front-panel mounted switch—just like the notch filter published in *Elektor* 9-10/2024 [1].

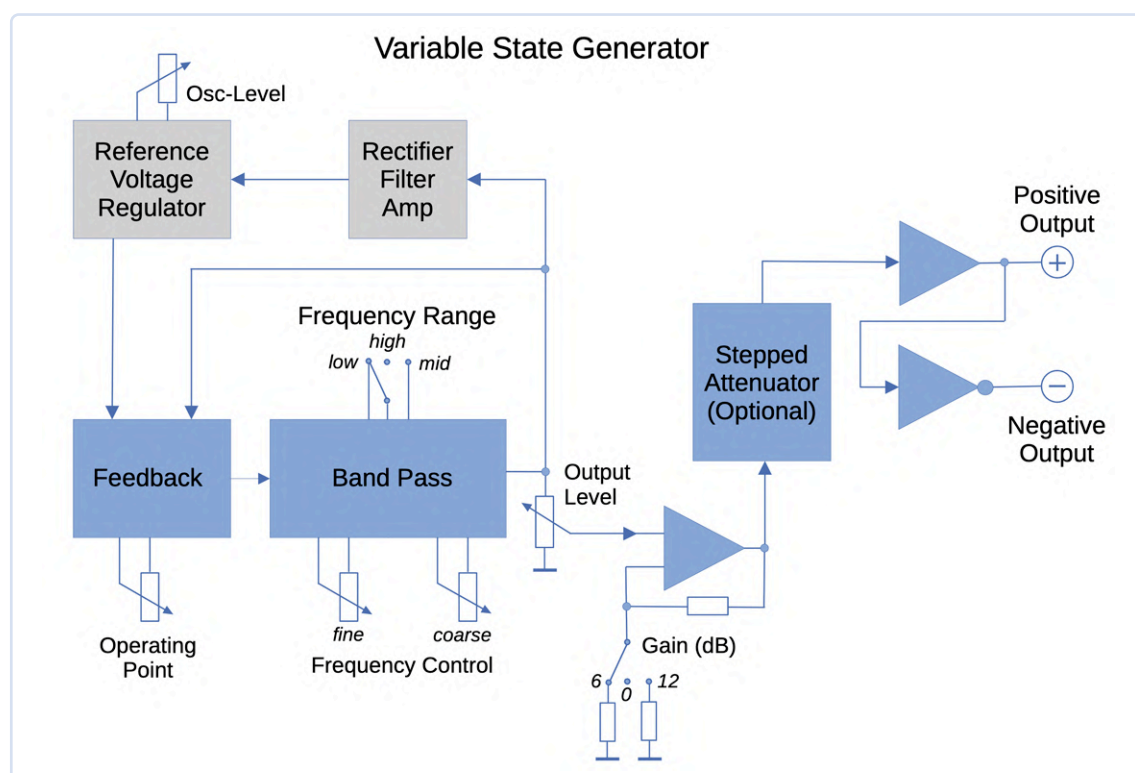


Figure 1: Block diagram of the audio frequency generator.



Component List

Resistors

Unless otherwise specified:

MELF 0204 or thin film SMD 1206 outline, 1% tolerance

R1 = 100 Ω

R2 = 510 Ω

R3 = 150 Ω

R4, R6, R8, R12 = 2k87

R5 = 200 Ω , multi-turn upright trim pot

R7 = 2k2

R9 = 0 Ω , SMD 0603

R10, R21 = not fitted

R11, R22 = 2k2

R13 = 1k5

R14, R64 = 0 Ω

R15 = 4k22

R16 = 2k15

R17 = 1 k Linear dual gang pot 4 mm shaft

R18 = 10 k Linear dual gang pot 4 mm shaft

R19, R20, R26, R27, R44, R45 = 4k7, SMD 0603

R23, R24, R28, R29, R46...R49 = 10 k, SMD 0603

R25, R53, R56, R60...R62 = 220 Ω

R30, R31 = 1 k, SMD 0603

R32, R33 = 47 Ω , SMD 1206

R34 = 3k3

R35, R36, R40 = 8k2, SMD 0603

R37 = 470 Ω , SMD 0603

R38 = 100 Ω , SMD 0603

R39 = 100 k, SMD 0603

R42 = 100 k, multi-turn upright trim pot

R43 = 27 k, SMD 0603

R50 = 2k2, SMD 0603

R51 = 5k1, SMD 0603

R52 = 10 k, linear potentiometer 4 mm shaft

R54, R55, R57...R59 = 3k3

R63 = 1k1

R65 = 4k7, hand wired*

Capacitors

C1, C7 = 39 n, COG, SMD 0805

C2, C8 = 100 n, COG, SMD 1206

C3, C11 = 1n2, COG, SMD 0603

C4, C12 = 33 n, COG, SMD 0805

C5, C9 = 680 p, COG, SMD 0603

C6, C10 = 15 n, COG, SMD 0603

C13, C14 = 2,200 μ / 16 V, Electrolytic 5 mm lead pitch, \varnothing 13 mm,

C15...C18, C22...C28, C30...C32, C34, C35 = 100 n, X7R, SMD 0603

C19, C20, C33, C36...C38, C40...C43 = 22 μ / 20 V, Tantalum, SMC B

C21 = 220 μ / 16 V, Electrolytic 5 mm lead pitch, \varnothing 10,5 mm

C44 = 3n3, X7R, SMD 0603

C45, C46 = 10 μ / 25 V, SMD 1206

C47 = 1 μ / 25 V, SMD 1206

C48 = 47 p, SMD 0603

C50 = 10 μ / 25 V, SMD 1206, hand wired*

Semiconductors

B1 = Bridge rectifier, B40C1000, DIL

D1...D5 = 1N4148, DO214AC

D5...D7 = SK34, DO214AC

D8, D9 = BAV199 oder BAV99, SOT23

IC1...IC3, IC9 = OPA2210, SOIC8

IC4 = 7812, TO220

IC5 = 7912, TO220

IC6 = 7805, TO220

IC7, IC8 = TL072, SOIC8

IC10 = NSL-32SR3*

LED1, LED2 = LED, SMD 0805

Q1...Q5 = 2N3904, SOT23

Q6 = 2N3904, SOT23, hand wired*

VR1 = TL431, TO92

Miscellaneous

F1 = PCB mounted 20 mm fuse holder 250 mA.

JP1, JP3, JP4, JP7, JP8 = 2 way pin header strip 0.1" pitch

JP2, JP5 = 4-way pin header strip. 0.1" pitch

JP6 = 3-way pin header strip, 0.1" pitch

K1, K2, K3, K4 = SMD relay, 5 V, Coto Technology 9802-05-00

TR1 = PCB mounted mains transformer, 2 x 12 V / 3.6 VA, Gerth 387.24.2

X1 = 3-way terminal strip, 5 mm pitch, AK300/3

X2 = 2-way terminal strip, 5 mm pitch, AK300/2

X3, X4 = PCB mounted BNC socket, e.g., AMP 227161 BNC

2x 1-pole. 3-position switch

3x control knobs for 4 mm shaft pots

PCB

* see text

Coarse and fine frequency control potentiometers are mounted on the front panel. The signal amplitude control consisting of a rectifier, filter, amplifier, reference voltage, control amplifier, and optocoupler has been borrowed from the "Analog 1 kHz Generator" circuit featured in *Elektor* 7-8/2024 [2]. The output stage also comes from that circuit but has been slightly modified here.

The output level is adjusted via a potentiometer. The three-position selector switch sets the gain to 0, 6, or 12 dB (corresponding to x1, x2, and x4). An optional stepped attenuator can be placed between the amplifier's output and the input to the final output stage. This can be

implemented using the relay-controlled "Attenuator for Audio Signals" featured in the *Elektor Circuit Special* 2024 [3] or more simply with a rotary switch. Attenuation step sizes of 6 or 10 dB would be ideal. If no additional attenuator is used, the corresponding connections are bridged using a 0- Ω resistor (R64).

In the first version of the audio generator, I used an OPA1632 differential op-amp to generate differential output signals but it introduced too much distortion. By the fifth iteration of the circuit (Version E), it was replaced with a conventional dual op-amp. In theory, you could leave this out entirely, but that would result in a single, unbalanced output.

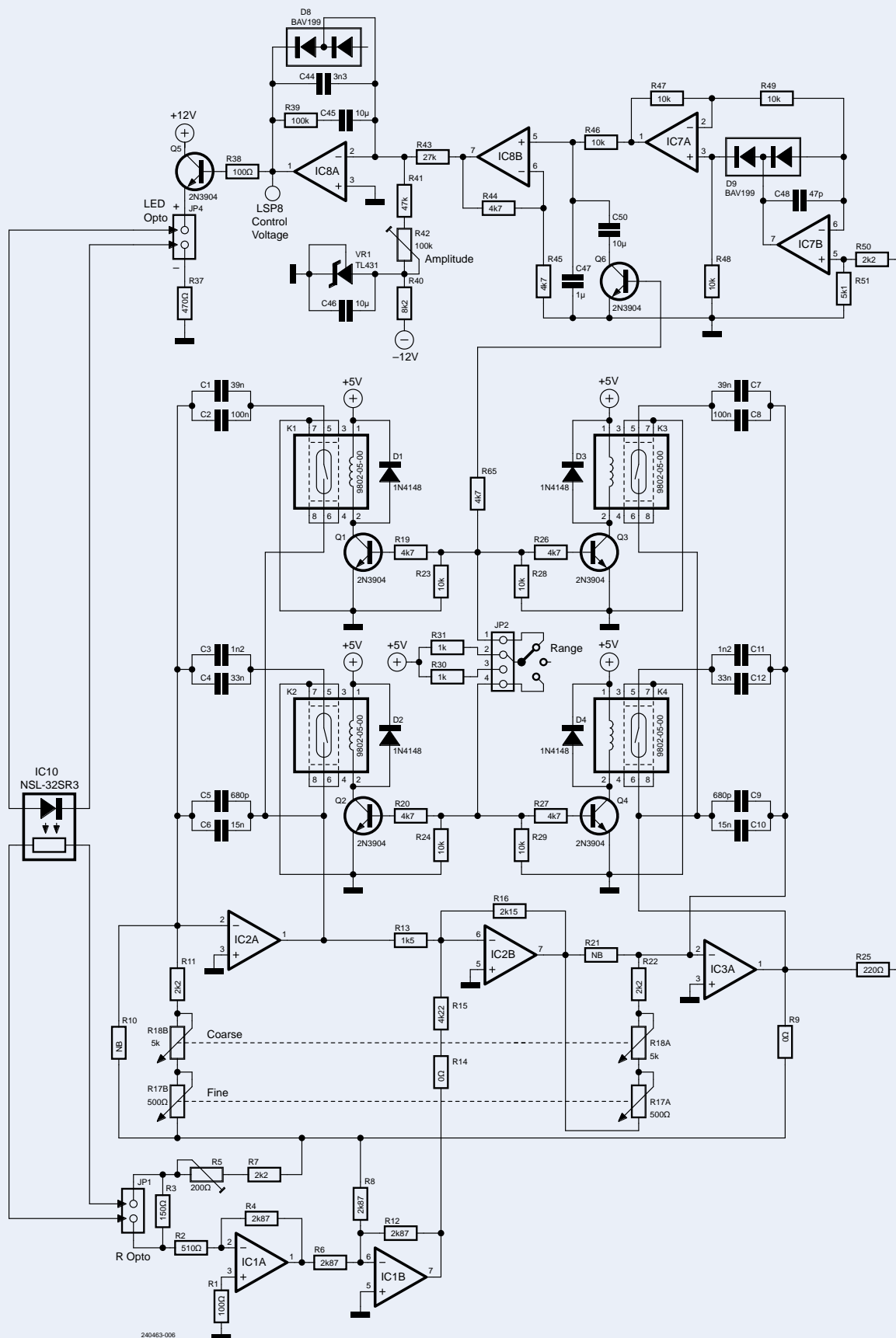
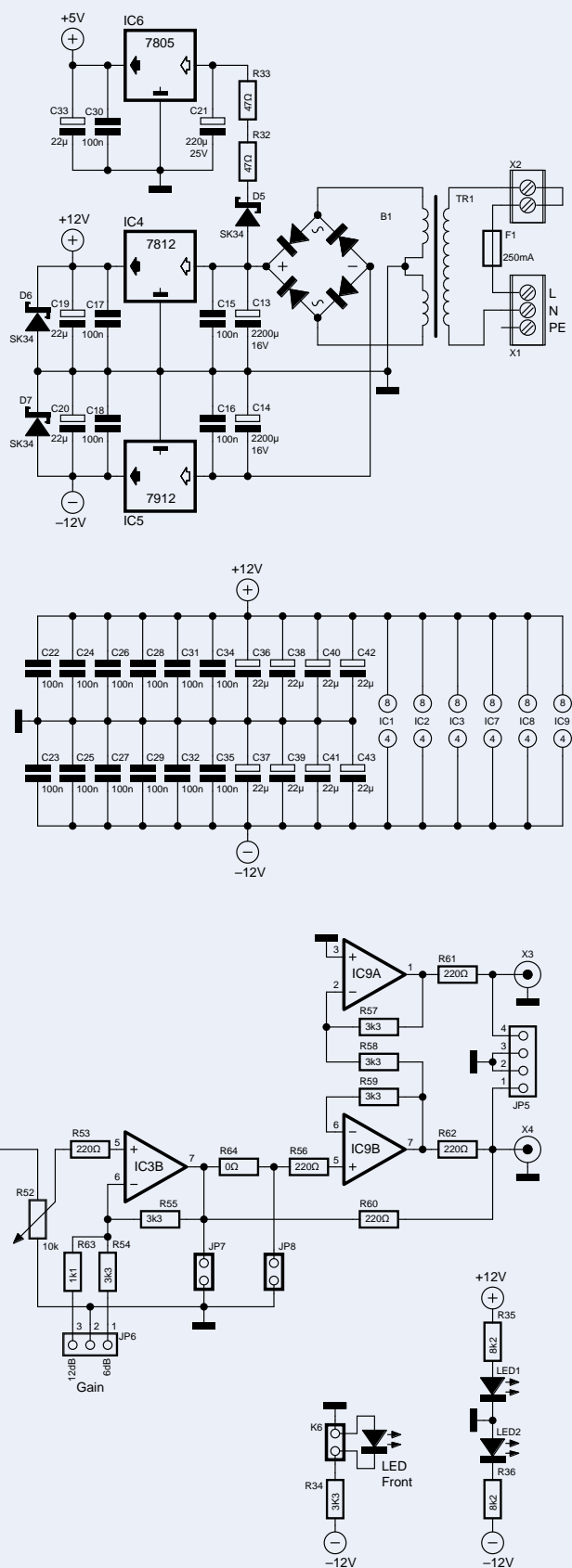


Figure 2: The audio signal generator circuit diagram and some elements of the design are taken from my previous projects.



The complete power supply design has also been borrowed from the previously mentioned notch filter design [1].

Details

Figure 2 shows the circuit diagram for the most recent Version E of the design. The *State Variable Filter* is built around four op-amps: IC1B, IC2A, IC2B, and IC3A. Just like the circuit used in the notch filter, the frequency range can be switched between three bands using four relays that control the frequency-determining parallel wired capacitors: C1||C2, C3||C4 and C5||C6 as well as C7||C8, C9||C10 and C11||C12. In this design, the achievable frequency range extends from 175 to 5500 Hz. Two twin-ganged potentiometers mounted on the front panel allow continuous control of the output signal frequency: R18 for coarse and R17 for fine adjustments. Feedback control used by the system is provided by IC1A, via an optocoupler (the variable resistance element of the optocoupler IC10 at JP1).

The sine wave output from IC3A drives the active rectifier around IC7, which is part of the amplitude regulation circuit. Its output signal level is smoothed by R46 and C47 and fed to the input of IC8B which has a fixed voltage gain of 2. The core regulation stage, built around IC8A, compares this rectified and filtered output signal amplitude with the reference voltage from VR1. The resulting voltage level defines the current through the opto coupler's LED via transistor Q5 which influences the value of its resistive element to control the gain of IC1 and close the feedback loop. This section of the circuit is almost identical to that used by the 1 kHz audio generator design [2].

To expand the amplitude control range, R3 (in parallel with the resistor in optocoupler IC10) has been increased to 150 Ω , and R2 changed to 510 Ω . Resistor R37 has also been swapped for a 470 Ω to allow for a higher current through the optocoupler LED. Extra components – C50, Q6, and R65 – have been added to increase the smoothing capacitor effect at low frequencies. There wasn't enough space on the PCB to accommodate these components, so they were wired by hand.

The amplitude of the output signal is continuously adjustable using potentiometer R52 and is buffered by IC3B. If needed, a switch at JP6 can be fitted to provide additional amplification by a factor of 2 or 4 (connect JP6 pin 2 to pin 1 for 6 dB or to pin 3 for 12 dB gain). The additional gain will however result in increased output signal distortion. A switchable output attenuator can also be inserted between J7 (input) and J8 (output). If an external attenuator is used, R64 (a 0-ohm resistor) must not be fitted.

For the differential output stage, I originally intended to use the OPA1632 ultra-low distortion differential audio amplifier but its performance in the prototype was disappointing. In the final design, I opted for the OPA2210 dual opamp for IC9A and IC9B to provide the differential output stage. Pads on the PCB are provided to mount right-angle BNC connectors (X3 and X4) for the differential output signals. Alternative output connectors can be wired to the output signals available at header strip JP5 if required.



Figure 3: The prototype PCB (version A) fitted into its case.

The entire power supply design was taken from the notch filter [1]. Its configuration is completely conventional and doesn't really need any explanation.

Figure 3 shows the populated PCB (this is still the first prototype Version A) installed inside its case. The PCB layout files can be downloaded in Eagle format from the Elektor Labs website [4]. **Figure 4** shows the front panel labeling. The output stage gain switch is missing from this version. The stepped attenuator option is also not included in this first prototype. For calibration and setup, refer to the corresponding section in the 1 kHz generator article [2].

Quality Components Are Key

The quality of the output signal doesn't just depend on a well-designed circuit — it's crucial that high quality components are used, especially when you are designing high-performance linear or audio circuits. For this reason, all resistors used in the signal path are Mini MELF 204 format, as they offer the same high-quality performance as leaded metal film resistors. Thin-film resistors in the 1206 SMD format can also be used. It is **important to use** COG-type capacitors in this design. The performance of the entire circuit would be compromised if a single X7R-type capacitor were substituted for any of the COG types. The high-performance OPA2210 opamp is used throughout the circuit for processing audio signals. ◀

Translated by Martin Cooke — 240463-01



Figure 4: Front panel of the prototype design. The gain switch and stepped output attenuator are not fitted in this version.

Questions or Comments?

Do you have questions or comments about this article? Email the author at alfred_rosenkraenzer@gmx.de or contact the Elektor editorial team at editor@elektor.com.



About the Author

Alfred Rosenkränzer worked for many years as a development engineer, initially specializing in professional television technology. Since the late 1990s, he has been designing digital high-speed and analogue circuits for IC testing. His special interests are in all things audio.



Related Products

- > **OWON SDS1102 2-ch Oscilloscope (100 MHz)**
www.elektor.com/18782
- > **Douglas Self, *Small Signal Audio Design* (4th Edition, Focal Press)**
www.elektor.com/18046

WEB LINKS

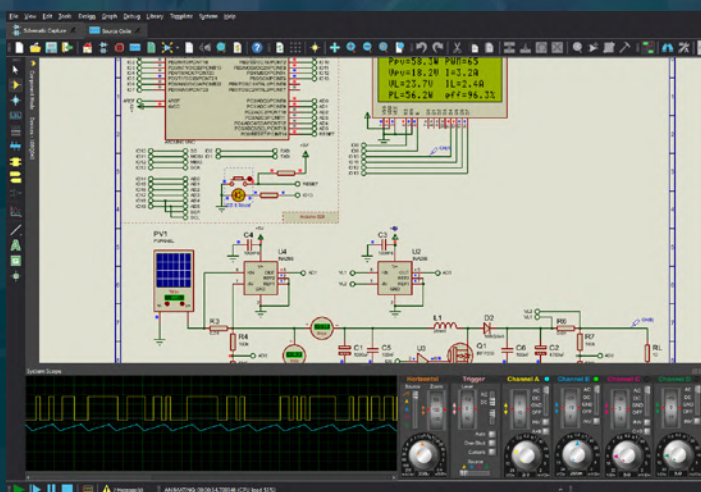
- [1] Alfred Rosenkränzer, "Audio Notch Filter with Adjustable Frequency," Elektor 9-10/2024:
<https://www.elektormagazine.com/magazine/elektor-354/63208>
- [2] Alfred Rosenkränzer, "Analogue 1 kHz generator," Elektor 7-8/2024:
<https://www.elektormagazine.com/magazine/elektor-346/62910>
- [3] Alfred Rosenkränzer, "Attenuators for Audio Signals," Elektor Circuit Special 2024:
<https://www.elektormagazine.com/magazine/elektor-350/63040>
- [4] Analogue Audio Generator, Elektor Labs: <https://tinyurl.com/ysns2nut>



**Brand new, high performance,
64-bit application framework
with full support for international
character sets and theming.**

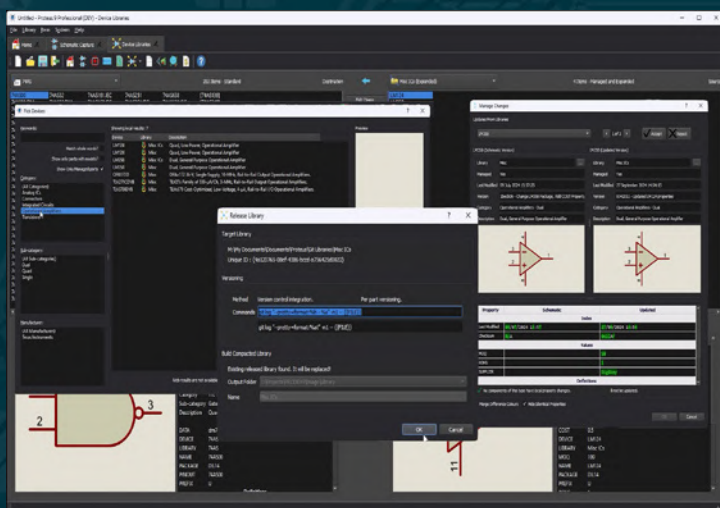
PROTEUS 9

...THE START OF A NEW ERA



Proteus VSM Simulation

- System Scope with live circuit probing.
- Live results on graph based simulations.
- Visual Designer for MicroPython.
- IoT Builder for MicroPython.



Proteus PCB Design

- Versioning of managed libraries with either repository or local folder.
- Software wide support for restricting use to managed libraries.
- Lightning fast, live power plane regeneration.
- Via Inspector tool for board analysis.



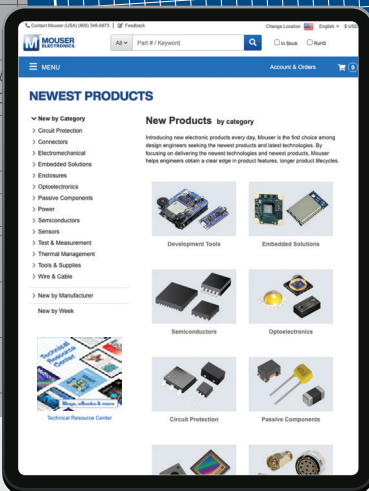
labcenter  **www.labcenter.com**
Electronics
Tel: +44 (0)1756 753440

Discover

Design

Develop

mouser.co.uk



Order - with - Confidence



**MOUSER
ELECTRONICS**