£9.95 | €10.95 | $12.95

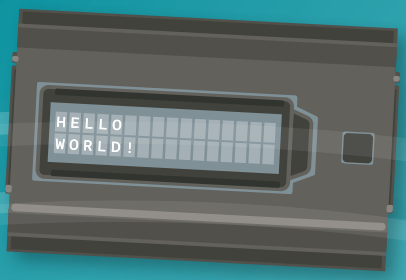
elektorMAG
design > share > earn

518
* SINCE 1961 *
DEC. 2022 & JAN. 2023
ELEKTORMAGAZINE.COM

Special edition
guest-edited by

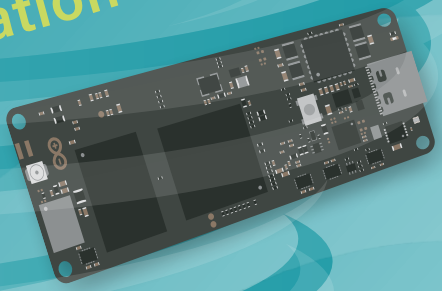
ARDUINO

140 Pages



Arduino **Projects**

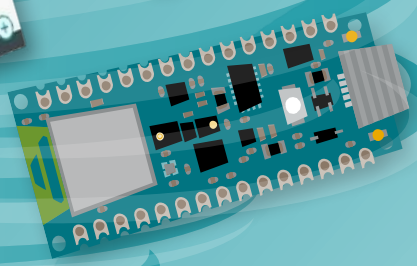
Home **Automation**



Connectivity **Simplified**

Go Professional
with Arduino Pro

Arduino **Portenta**
Machine Control and
Arduino **Portenta H7**



Prototyping to **Production**

Art with **Arduino**

In this issue

and much more!

Taking over Elektor!

Building Future Skills
with Arduino
p. 90

Save the Planet
with Home Automation
p. 62

Getting to know Arduino
with Fabio, Massimo, and David
p. 32

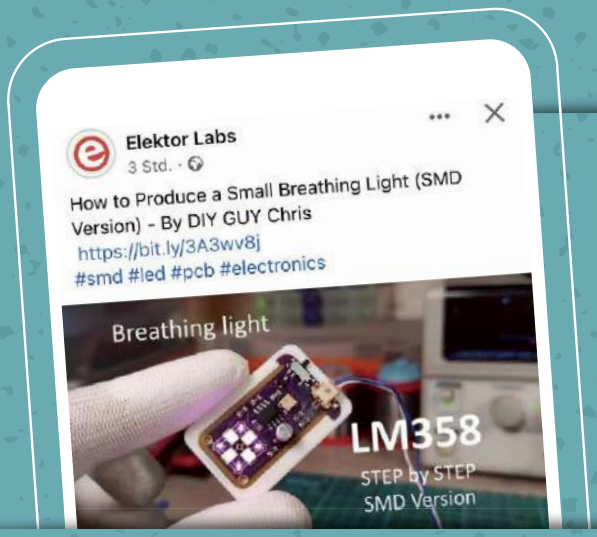
EP-0518

New *Arduino* or *Electronics* Project?

Share it with our community!

Follow us on:

www.elektor.com/TW



www.elektor.com/Intsta

www.elektor.com/FB

elektor

Elektor Magazine, English edition

Volume 48, No. 518
December 2022 & January 2023
ISSN 1757-0875 (UK / US / ROW distribution)

www.elektor.com
www.elektormagazine.com

Elektor Magazine, English edition
is published 8 times a year by
Elektor International Media

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

Memberships:
E-mail: service@elektor.com
www.elektor.com/memberships

Advertising & Sponsoring:
Raoul Morreau
Phone: +31 (0)6 4403 9907
E-mail: raoul.morreau@elektor.com

www.elektor.com/advertising
Advertising rates and terms available on request.

Innovation **with Arduino**

As most of you remember, last year's 60th anniversary edition of Elektor Magazine included a great interview with David Cuartielles, co-founder of Arduino. The indelible impact Arduino has had on our field, and thus on Elektor, made an in-depth conversation inevitable. The interview with David led to the idea of inviting Arduino to take on the challenge of becoming the 2022 guest editor of Elektor Magazine. We are now excited to present you with the fruit of the hard labor from both the Arduino and Elektor teams.

What can you expect from this super-thick, special edition of Elektor Magazine? Over the past several months, our engineers and editors have worked closely with David and his colleagues at Arduino to prepare projects, interviews, and tutorials about the many Arduino solutions currently at your disposal, from the Arduino Uno to Arduino Cloud to Arduino's newest professional products, including the Portenta family. In typical Elektor fashion, we have worked hard with our friends at Arduino to bring you a magazine packed not only with innovative projects like David's soil-monitoring system, but also insights from creative artists such as Jacob Remin who are using Arduino in innovative ways. Whether you are maker looking to start a new DIY project or a professional engineer designing a new product, you are sure to find the project articles, tips, and tutorials in this issue helpful and inspiring. Enjoy!

C.J. Abate (Content Director, Elektor) &
Jens Nickel (Editor in Chief, Elektor)

My passion for electronics goes back to the moment I discovered — the hard way — that the small light bulb in my flash lamp could not be connected to the mains for long without it blowing up the fuses in the whole apartment. I was 5 and saw electronics as a mystery to solve. I could not imagine how relevant that moment would become for my future. As I grew older, I figured out that electronics were not — always — built in the solitude of a nerdy teenage room. It was through Elektor that I discovered a whole community of like-minded people. Inventors willing to share their knowledge and projects in the pages of the Elektor Magazine. This concept of community became the other fundamental pillar of my professional career.

Years passed and we — as in I and my partners Massimo, Dave, and Tom — built Arduino together with a huge community of collaborators. This special issue of Elektor Magazine is a homage to our community, to everyone who ever took an Arduino in their hands to make a project, to those who spent their time teaching others about the importance of digital technology, to the artists, designers, engineers, and scientists aiming at making a great job by means of an Arduino board, and — why not — to the Elektor community who got us in and helped amplify our message throughout the years.

In this issue you will find interviews with members of the Arduino community, special projects we — at Arduino — do in our spare time, and real-life examples of how the Arduino community is helping make the connected world a better place. This is just a tiny portion of everything that has happened since Arduino started in Ivrea in 2005. We hope you will like the stories and projects we hand-picked for you. And remember: sharing is caring!

David Cuartielles (Co-Founder, Arduino)

The Team

THIS EDITION

Volume 48, No. 518
December 2022 & January 2023

Connected Projects, Simplified

Dive Into the Arduino Cloud

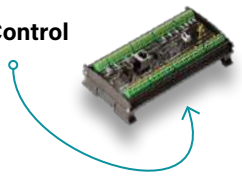
18

ALTAIR 8800 SIMULATOR

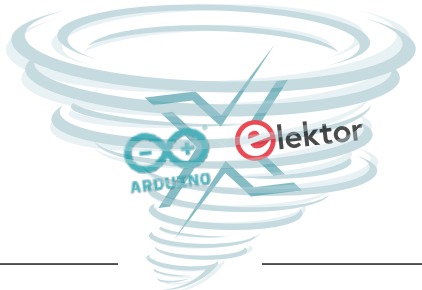
46

Hardware Simulation of a Vintage Computer

Regulars



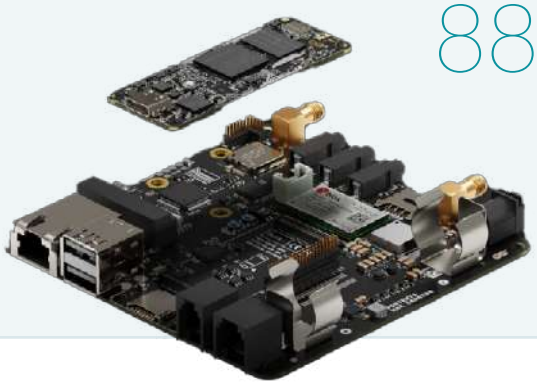
Features



Grow It Yourself

A Digitally Controlled, Single-Box Solution for Indoor Farming 54

The New Portenta X8 (with Linux!) and Max Carrier Redefine What's Possible



88

Projects

Next Edition

Elektor Magazine Edition 1-2/2023 (January & February 2023)
As usual, we'll have an exciting mix of projects, circuits, fundamentals and tips and tricks for electronics engineers and makers. We will focus on audio and video electronics.

From the contents:

- Video output with microcontrollers
- ESP32 Camera
- 32-Ω Headphone amplifier
- Workshop: ESP32 Audio Development Framework
- Tube amplifier
- Poor man's ChipTweaker
- USB True random number generator
- ATX Power supply for Raspberry Pi

And much more!

Elektor Magazine edition 1-2/2023 (**January & February 2023**) will be published around January 5, 2023. The arrival of printed copies for Elektor Gold Members is subject to transport. Contents and article titles are subject to change.

Want more content from Elektor and Arduino? In the coming weeks, we will publish a bonus edition of Elektor Mag — also guest-edited by Arduino — that is packed with more Arduino-related projects and articles: a controller for Spotify, retro gaming with Arduino, and more! Subscribe to the Elektor E-Zine (www.elektormagazine.com/ezine), and we'll deliver the bonus edition directly to your inbox!

Arduino **Portenta Machine Control** and Arduino **Portenta H7**

A CAN-to-MQTT Gateway Demo Project

Figure 1: Arduino Portenta Machine Control.

By Mathias Claussen (Elektor)

Is the Arduino Portenta Machine Control an alternative to a programmable logic controller (PLC)? It's Arduino's foot in the industry door. Here's a brief overview of the Arduino Portenta Machine Control and the Arduino Portenta H7.

Arduino in industrial environments: this is the path that the Arduino Pro series would like to take. With the Arduino Portenta Machine Control (**Figure 1**), a platform is now available that has I/Os with the typical 24-V or 4-to-20-mA compatibility on board. In addition, Ethernet, CAN-FD, RS-232, RS-485, and RS-422 are provided as bus systems. For further connectivity, Wi-Fi, Bluetooth, and USB are available. Of course, I^2C and connections for thermocouples are not forgotten. The connections are very easy to reach on the Arduino Portenta Machine Control through terminal strips, as you can see in **Figure 2**.

Anyone who now feels reminded of the Siemens S7 connections is not wrong. The Arduino Portenta Machine Control is intended to create options for developers in existing plants — options that would not be achievable with classic PLCs. It's also not surprising that the Arduino Portenta Machine Control is equipped with a functional DIN rail housing.

Arduino Portenta Machine Control

What makes Arduino Portenta Machine Control different from a classic PLC? If you search a bit, you will find information about the inner workings of a Siemens S7-1200 [1]. But, what makes the Arduino Portenta Machine Control better? There is no Raspberry Pi or Beagle Bone Black built in. The heart of the machine is the Arduino Portenta H7 and its STM32H747XI microcontroller (MCU). The MCU is an asymmetric dual-core type with an ARM Cortex-M7, which can be clocked at up to 480 MHz, and an Arm Cortex-M4, which can be clocked at 240 MHz. In addition to the STM32's internal 2 MB Flash and 1 MB SRAM, another 8 MB of SD-RAM and 16 MB of QSPI-Flash (Quad Serial Peripheral Interface Flash [2]) are available.

The STM32 Dual-core MCU on the Arduino Portenta H7 thus provides enough resources to run more than just simple PLC applications. In this constellation, the system is best equipped to implement neural networks and to enable new applications, thanks to machine learning. The Arduino Portenta H7 is directly supported by Edge Impulse for an easy entry into machine learning. Best of all, the Arduino Portenta H7 has a USB-C port that can not only operate as a USB host or USB slave, but, thanks to clever design, is capable of providing up to 720p (1280×720

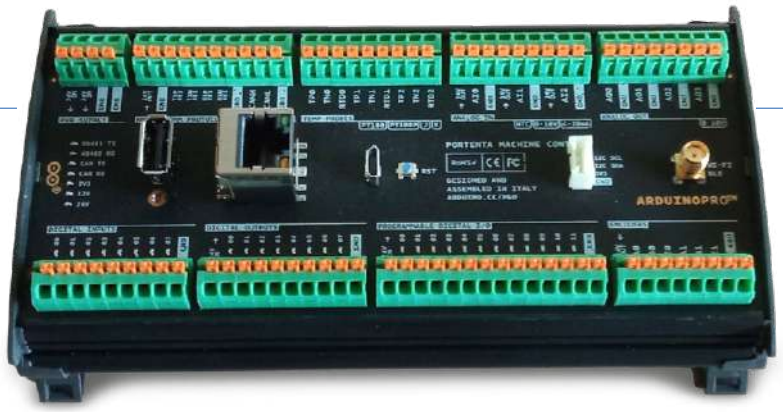


Figure 2: Arduino Portenta Machine Control's terminal strips.

pixels) video output. So, a monitor can be implemented as an human-machine interface (HMI) directly with the Arduino Portenta H7. Unfortunately, the USB-C port that provides the video signal is not directly accessible on the Arduino Portenta Machine Control.

With all these new options, you will likely have ideas for your own applications. In this article, I will show you how to tackle a small project with the Arduino Portenta H7 and the Arduino Portenta Machine Control.

Software Framework and IDE

For the Arduino Portenta H7, the Arduino framework is available and can be used with the Arduino IDE in version 1.x or the new IDE in version 2.x. As for the editor, version 1.x is not the best editor in its class, and version 2.x brings significant improvements. The user interface of version 2.x is shown in **Figure 3**.

To use the Arduino IDE to control the Arduino Portenta H7 or Arduino Portenta Machine Control, only the appropriate board package (**Figure 4**) must be installed for you to start developing.

You also can take a look at PlatformIO. [3] There, the Arduino Portenta H7 is also directly supported and Visual Studio Code is used as an editor. The Arduino IDE and application programming interface (API) hide other background software machinations. That would be the Mbed OS [4] as the operating system and its hardware abstraction library (HAL) for the hardware. In addition, there is the openAMP library [5], which can be used for communication between the processors. For the hardware-specific drivers, the STM32Cube from STMicroelectronics is used, so that proven code directly from the manufacturer is in use. The software layers can be seen in **Figure 5**.

Useful Arduino Libraries for Arduino Portenta Machine Control and Arduino Portenta H7

In addition to the core for the Arduino IDE, the board package, there are a number of libraries that enable accelerated development. For the Arduino Portenta Machine Control, the library named Arduino Portenta Machine Control is available. This library provides suitable functions for the Arduino Portenta Machine Control temperature sensor interface, suitable control for the RS-485 driver modules, and the 24-V GPIOs. In addition, the control of the CAN interface is provided here.

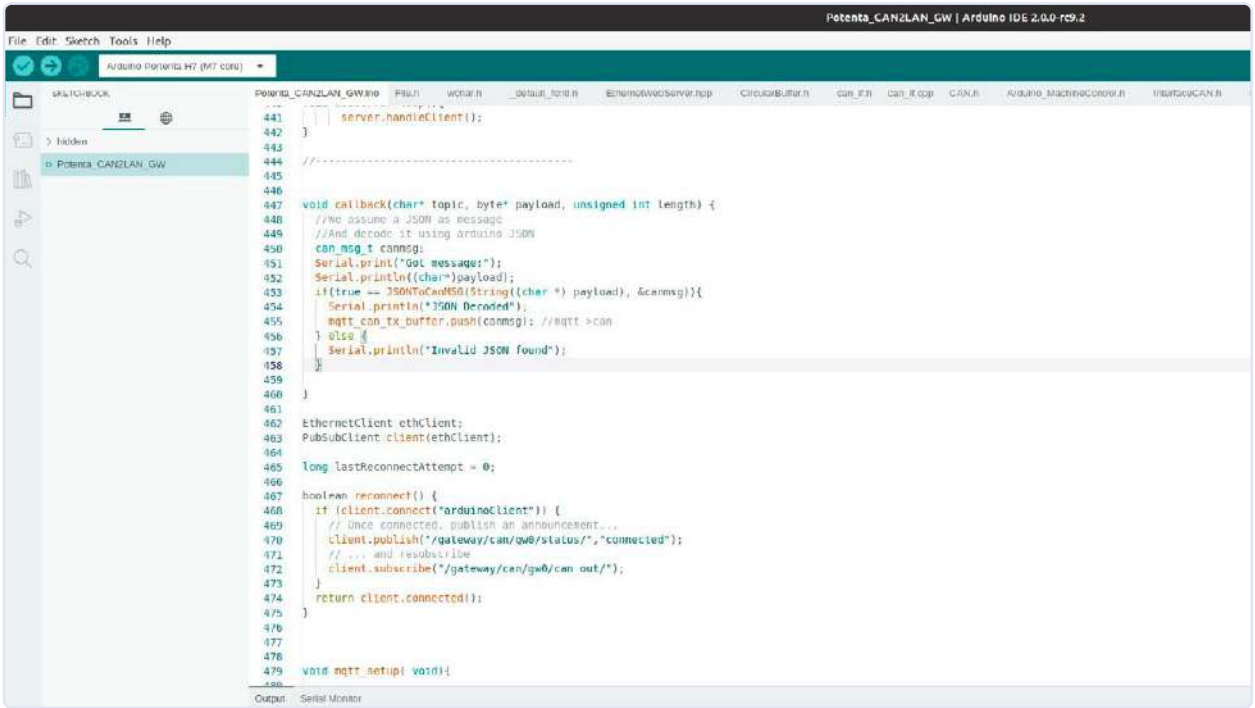


Figure 3: Arduino IDE version 2.

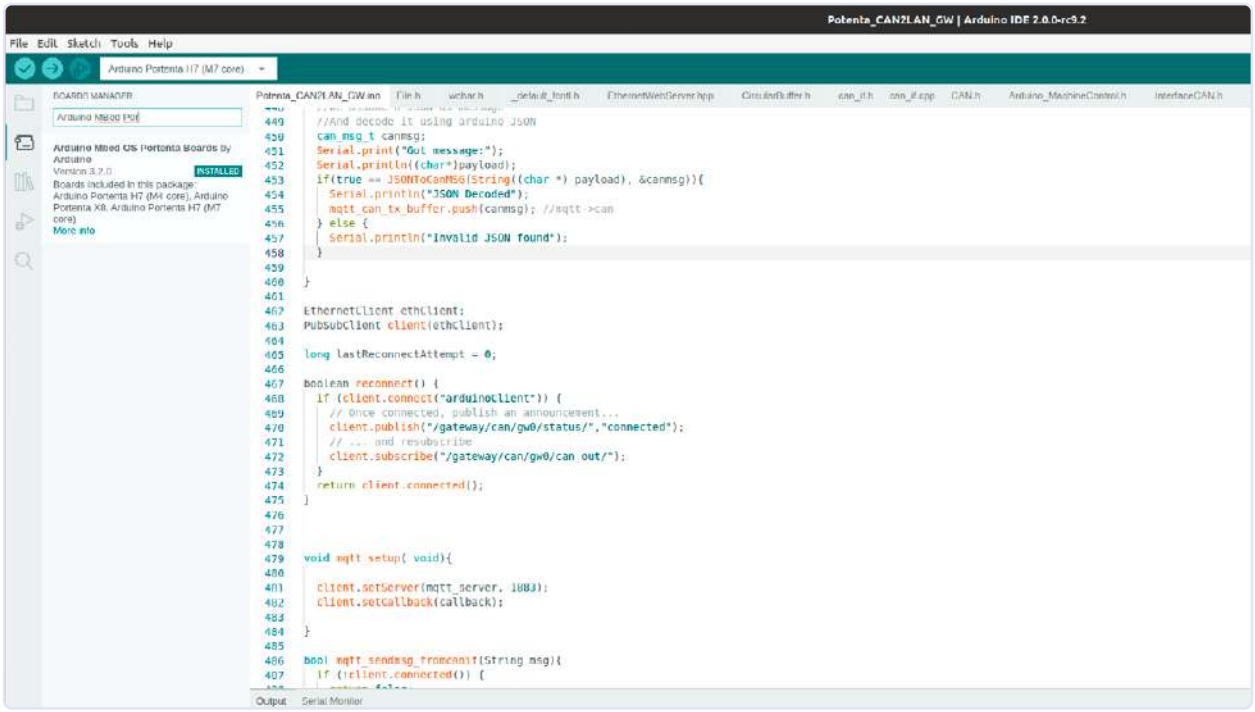


Figure 4: Board in the Board Manager.

If you want to use the integrated Wi-Fi to install new firmware remotely, you can use the Arduino_Portenta_OTA library. This can be used to write firmware updates to the external flash or an SD card. The next time the bootloader is started, it will update the firmware.

Since Bluetooth Low Energy (BLE) is also supported, it still requires a BLE stack. This is called ArduinoBLE and can be downloaded via the Library manager.

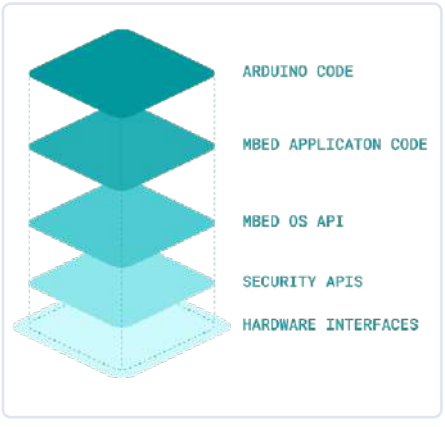


Figure 5: Softwares layers for the Arduino Portenta H7. (Source: Arduino, https://bit.ly/arduino-mbed-stack)

When it comes to graphics output, the Arduino Portenta H7 relies on a library that arranges pixels suitably. The Little Versatile Graphics Library (LVGL) is such a library and in its current version a good basis for your own HMI development.

Almost a Classic PLC: Arduino Portenta Machine Control and IEC 61131-3

At the time of writing, support for IEC 61131-3 [6] was still in beta and the screenshots are from a beta version (**Figure 6** and **Figure 7**). But, what is IEC 61131-3, and why is it good that the Arduino Portenta Machine Control has support for it?

Anyone who works with classic PLCs such as an S7-1200 from Siemens will already have come into contact with IL (instruction list), LD (ladder diagram) or ST (structured text). Creating controllers with these programming languages has become a standard in the industry. These languages and procedures are described in IEC 61131-3. With the support of the Arduino Portenta Machine Control, users can thus fall back on existing knowledge and implement proven concepts without much effort.

One MCU, Two Processor Cores

The programming of the STM32 MCU differs from the other Arduinos. Here, we have two processors that work independently, but can access the same resources. In the Arduino IDE you can now choose if the project should run on the Arm Cortex-M7 or -M4 and how the available flash is divided between the processors. For example, the real-time control of the machine can run on one of the cores, while a MicroPython instance does its job on the other core. Also, you can have your AI application compute on the Arm Cortex-M7 core and use the M4 core for other communication, such as TCP/IP, CAN or Modbus. A block diagram of the STM32 MCU can be seen in **Figure 8**.

External RAM, Flash and Secure Element

As I mentioned at the start, 8 MB of SD-RAM is connected, and can be accessed by both CPUs. Not only is access to data possible, but also the execution of program code from the external RAM. In addition, there is the 16-MB QSPI Flash, which the STM32 could also mount directly in its memory area. This would enable the MCU to have the 16-MB Flash available like internal memory and also execute code directly from there (XiP — Execution in Place), but this option is not provided in the Arduino

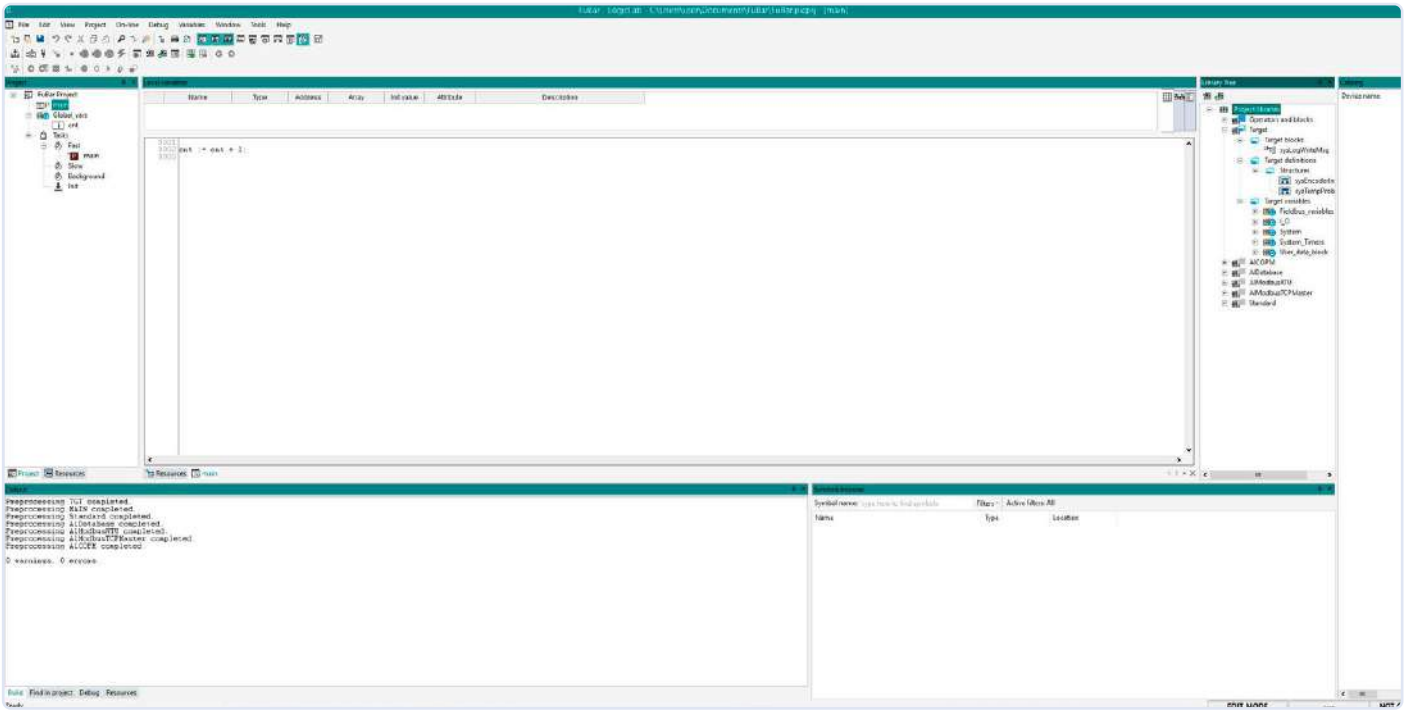


Figure 6: Simple counter in the Arduino PLC editor.

software stack. The 16-MB flash is treated as external QSPI flash and used as mass storage with a matching flash file system. This allows up to 16 MB of data to be stored in memory, which can be very handy for over-the-air (OTA) updates or other data.

Something to consider when developing code is where it's placed. Execution from internal flash or SD-RAM will cause the MCU to wait for code and data for varying lengths of time. Clumsy placement can result in significant loss of processing power.

The Arduino Portenta H7 has a Secure Element. This is an NXP SE050 Secure Element, whose smaller offshoot, the SE050E, has a review that can be found on the Elektor website. [7]

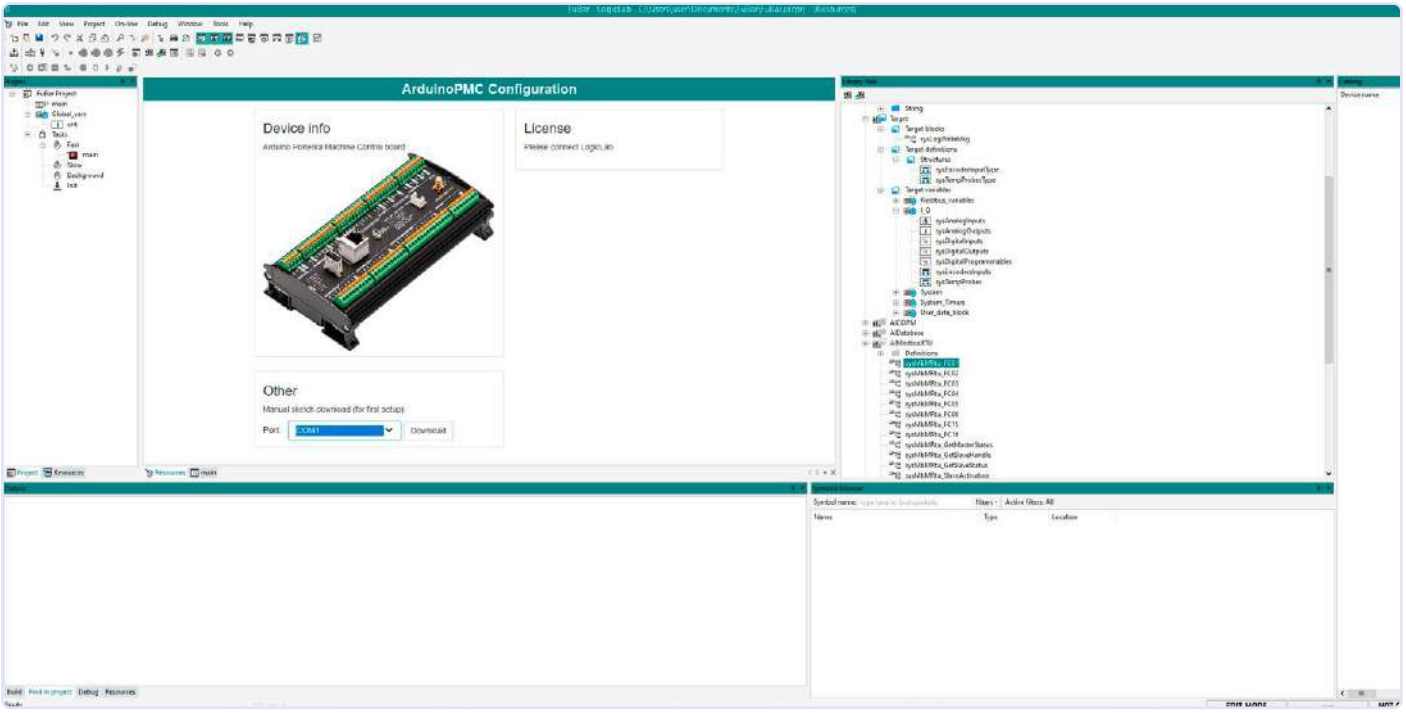


Figure 7: Project overview in the Arduino PLC Editor.

Example Project Using the Arduino Portenta Machine Control

Now that the hardware has been introduced, a small example project will show you how to write software for the Arduino Portenta Machine Control. A CAN-to-MQTT gateway is to be realized here. This gateway will process CAN messages according to CAN 2.0B and forward them as MQTT messages to a server connected via LAN. This software is only a short demonstration and the source code is not even close to what one would classify as suitable for production.

Why a CAN-to-MQTT Gateway?

If you want to record and evaluate data on a CAN bus, you can do this comfortably from a distance. The data can then be written to a database or examined for anomalies using other methods (e.g., with AI). Due to the transport via MQTT, several participants can evaluate the CAN data or send messages to the CAN bus. And CAN is not only used in industry or cars; a CAN bus can also be present in the domestic model railroad. [8]

CAN, MQTT and WebSocket Libraries

To send or receive CAN messages, the integrated CAN controller is used, which also handles Controller Area Network Flexible Data-Rate (CAN-FD) on the hardware side. Unfortunately, the CAN controller is somewhat slowed down here by the Mbed OS, which acts as an intermediate layer, and is currently limited to CAN 2.0B.

The PubSubClient [9] version 2.8 is used for the MQTT connection. For the WebSocket connection, EthernetWebServer from Khoi Hoang [10] is used. This also provides a web server with which web pages can be delivered. In terms of functionality, it is very similar to that of the ESP32, so that applications can be easily migrated from an ESP32 or ESP8266, at least theoretically.

The web server itself is included in the project and also returns a static page with the error 404 (page not found). As for reading from the Arduino Portenta's flash file system, there are a few minor incompatibilities between the API of the EthernetWebServer library and the API for

Figure 8: Overview STM32H747XI. (Source: STMicro, https://bit.ly/3xoRTDF)

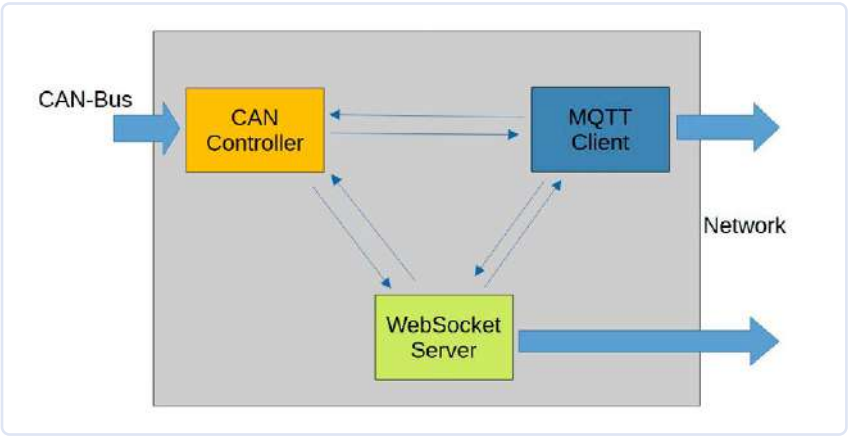


Figure 9: Rough concept of the software.

Figure 10: Configuration of CAN and MQTT server.

accessing the Arduino Portenta's file system. With a little more time and effort, however, this can be fixed by adding a small class that then provides file access suitable for the web server. The software's concept can be seen in **Figure 9**.

The software was already written with the idea of a functional web server and provides the CAN messages not only via MQTT, but also via WebSocket. This would then also allow visualization and interaction of the CAN bus via web browser.

Software Structure

For the gateway, the three components are an MQTT client, a WebSocket server, and the handling of the CAN data. Since neither WebSocket nor MQTT prescribe how the data is exchanged, JSON is used here.

The configuration of the software is done here completely in the source code, due to the project's short development time, as well as minor complications with the web server. The appropriate bit rate must be set for CAN and the IPv4 address in the network must be entered for the MQTT server. The part of the software for the configuration can be seen in **Figure 10**.

CAN messages received by the CAN controller are forwarded by the software via MQTT and WebSocket. If JSON data with CAN messages is successfully decoded via MQTT, it is forwarded to the CAN controller and to the WebSocket server. For WebSocket data containing CAN messages, these are forwarded to the MQTT server and the CAN controller.

The WebSocket server was intended to be able to display the CAN data via a web interface or to send data. This would only require a terminal device with a browser. However, the server was left in the software because other systems can connect directly to the Arduino Portenta Machine Control to exchange data via WebSocket. No MQTT server would then be necessary here.

JSON in MQTT Messages and WebSocket Connections

JavaScript Object Notation (JSON) [11] is used for data exchange. The structure of the JSON is quite simple and contains the data of the CAN message as well as the interface (CAN, MQTT, or WebSocket) through which the message was received. An example of such a JSON String looks as follows:

```
{
    "src": "can_if0",
    "type": "can",
    "extid": false,
    "canid": 2047,
    "length": 8,
    "data": [
        0,
        0,
        255,
        255,
        0,
        0,
        0,
        128
    ]
}
```

For the MQTT connection, the topic "/gateway/can/gw0/can_in/" is used for CAN messages received by the CAN controller. The CAN message is published here as a JSON message so that all those subscribed to this MQTT topic receive a copy of the message.

Messages received via WebSocket are published to the "/gateway/can/gw0/ws_in/" topic of the MQTT server. This makes it possible to distinguish which messages were sent via MQTT and which were sent via WebSocket to the CAN interface.

The topic "/gateway/can/gw0/can_out/" is subscribed on the MQTT server. If a message is now published there, the software attempts to decode this as a JSON message and to generate a suitable CAN message from it. If this is successful, the message is passed on to the CAN interface and the WebSocket server.

What Happens Next?

The software here is only intended as an idea and is far from complete. It is meant to give hints on how your own projects could be designed and show how you can quickly write your own applications with existing libraries and some knowledge about the Arduino framework. The software can be downloaded from the Elektor Labs site. [12]

The Arduino Portenta Machine Control costs €300. As you make your project budget, keep in mind that, while the case is functionally designed, it lacks a Wi-Fi antenna, so plan accordingly. Also, make a note to get an antenna setup with 3 dB or less gain to stay within the regulations.

As in most cases, software can be a challenge. But getting started with the Arduino framework is very easy and you can reuse alot of your knowledge. With the Arduino IDE 2.x, the editor has been improved a lot and getting all software parts for the Arduino Portenta Machine Control is just a matter of minutes. At some point libraries need some user input, so if you find bugs, please report them. This will help ensure that everyone has a better experience at the end.

As for the Arduino Portenta Machine Control, the impression is: A lot of potential. Its Arduino Portenta H7 is a very solid base in terms of hardware, and the price of about €120 makes it interesting for many projects. The option to realize an HMI is very inviting. Also, having RAM and Flash connected to the STM32H7 on the Arduino Portenta H7 allows it to hold more complex data and even run larger AI models on the MCU. With Wi-Fi, Bluetooth, and Ethernet, as well as hardware for the common bus systems, hardly any wishes remain unfulfilled. The Arduino Portenta H7 can not only serve as a basis for industrial control applications, but would certainly also be well-suited as the heart of other applications in terms of hardware.

220448-02

About the Author

Mathias Claussen is a senior engineer and technical editor at Elektor. Please feel free to contact Mathias at mathias.claussen@elektor.com. You can read many of his articles at www.elektormagazine.com/claussen. You also can watch Mathias on the monthly Elektor Lab Talk livestream on YouTube (www.elektormagazine.com/elt), where you can ask him your questions live!

WEB LINKS

[1] Inside a Siemens S7-1200: https://sec-consult.com/blog/detail/reverse-engineering-architecture-pinout-plc/
[2] Wikipedia: QSPI: https://en.wikipedia.org/wiki/Serial_Peripheral_Interface#Quad_SPI
[3] PlatformIO: https://platformio.org/
[4] ARM Mbed OS: https://os.mbed.com/mbed-os/
[5] OpenAMP GitHub Repository: https://github.com/OpenAMP/open-amp
[6] Wikipedia: IEC 61131-3: https://en.wikipedia.org/wiki/IEC_61131-3
[7] M. Claussen, "NXP EdgeLock SE050E Secure Element," Elektor, June 2022: https://bit.ly/nxp-edgelock-SE050E
[8] Märklin CS2 CAN protocol: https://www.maerklin.de/fileadmin/media/produkte/CS2_can-protokoll_1-0.pdf
[9] pubsubclient GitHub Repository: https://github.com/knolleary/pubsubclient
[10] EthernetWebServer GitHub Repository: https://github.com/khoih-prog/EthernetWebServer
[11] JSON: https://www.json.org/json-en.html
[12] Project on Elektor Labs: https://www.elektormagazine.com/labs/can-to-mqtt-gateway-with-arduino

Unboxing the Elektor LCR METER with David Cuartielles

Save the date: January 26, 2023

Would you like to unbox the Elektor LCR Meter Kit with Arduino co-founder David Cuartielles? Watch the January 26, 2023 (18:00 CET) episode of *Elektor Lab Talk*, where he will join Elektor engineer Mathias Claussen and Editor Jens Nickel to discuss the LCR Meter Kit, as well as take your questions about the Arduino technology and this guest-edited edition of *Elektor*. Don't miss the livestream. Bring your questions!

220555-01

Elektor LabTalk

Watch David live on Elektor Lab Talk on January 26, 2023!

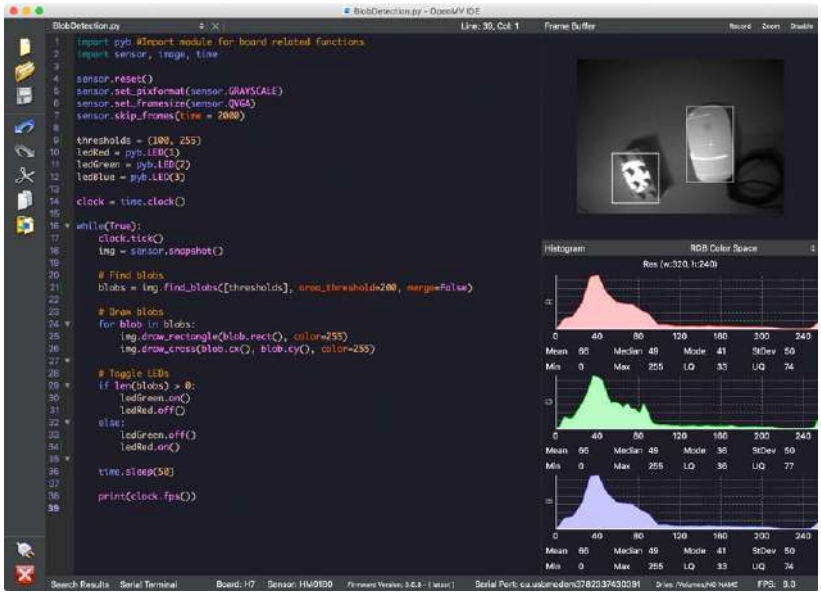
www.elektormagazine.com/labtalk-david

MicroPython Enters the World of **Arduino**

By Stuart Cording (Elektor)

MicroPython has made it to the world of Arduino, providing the first significant alternative to programming in C and C++. So, what's all the fuss, how easy is it to use, and who can benefit from programming in this, for microcontrollers, relatively new language? Elektor spoke to Sebastian Romero (Head of Content, Arduino) to find out more.

C and C++ have been the staple of Arduino software development since Arduino's inception in the early 2000s. Thanks to a predefined program structure with a setup() and loop() function, beginners to the world of embedded software development have been guided through setting up their board and executing their application in a loop. Now there is a new language available. MicroPython is a lightly stripped-back version of Python, an interpreted, general-purpose programming language that targets microcontrollers. The question is, why use an interpreted language on hardware used for real-time applications?



Blob detection running on Arduino Portenta H7 in OpenMV.

About Sebastian Romero (Head of Content @Arduino)

Sebastian Romero, head of content at Arduino, is an interaction designer, educator and creative technologist with a weakness for humans. With his team, he is responsible for crafting enthralling learning experiences to help millions of engineers, designers, artists, hobbyists and students to innovate.

"Because MicroPython's simplicity makes it well suited for beginners, educators were one of the first to ask us about it," explains Sebastian Romero, Head of Content at Arduino.

C makes interaction with microcontroller registers easier than with assembler, and object-oriented programming (OOP) in C++ makes for more concise code with fewer mistakes. However, parsing strings is challenging, and there is no native support in the language for handling today's web data formats, such as HTTP, JSON[1], or RegEx[2] (regular expression). With today's education revolving around interaction with the Internet and web services, C/C++ has been sidelined in favor of languages such as Python, which make coding such applications more straightforward.

"As a result, if you're a tutor teaching Python, you prefer to stick with Python when the topic of microcontrollers comes up," says Sebastian.

Of course, it's not just educators. Makers have had a range of MicroPython-capable boards on offer from

other sources, such as the ESP32, Raspberry Pi Pico, and pyboard, and the industry is increasingly looking to MicroPython as well. The rapid growth in machine learning (ML) is, in part, thanks to the existence of libraries available for Python. With teams of engineers competent in Python, few want to switch to C/C++ when they transfer their ML model and application to a microcontroller, preferring to stick with one development stack. The other issue is the workforce – it is increasingly challenging to find C/C++ programmers, while academia is churning out plenty of Python-competent engineers.

MicroPython Vs. Python: What's the Difference?

Python started its life back in the late 1980s, designed by Guido van Rossum[3]. Designed to be fun to use, it also aims to be explicit rather than implicit, simple, and result in readable code. In 2013, Damien George successfully launched a Kickstarter [4] campaign to deliver a version designed from the ground up for microcontrollers along with the pyboard hardware to run it. Micro Python, as it was named at the time, promised a scripting language that would "allow you to effortlessly blink LEDs, read voltages," and more. USB-enabled microcontrollers would appear as a USB flash drive onto which code could be uploaded. Alternatively, the device could appear as a serial device, offering a command line known as REPL (read, evaluate, print, loop).

Unlike C/C++ sketches which must be compiled and downloaded, MicroPython can be executed immediately after every change.

As might be expected, MicroPython requires a reasonable amount of memory to run in order to support uploaded code and its interpretation during execution. While 128 KB flash and 8 KB SRAM is enough, the feature set would be so limited as to make for a poor experience. Hence, most MicroPython boards settle for a microcontroller with at least 256 KB flash and 16 KB SRAM. This also has the side effect of selecting a device with a relatively powerful processor operating at around 50 MHz or more that offers a respectable range of peripherals.

"It constantly surprises me how much you can achieve with just 16 KB of SRAM," shares Sebastian - the quote is from Jim Mussared, an Embedded Engineer at micropython.org. "Most students, at least with introductory-level projects, don't need lots of heap. Their projects typically grow in complexity due to more code."

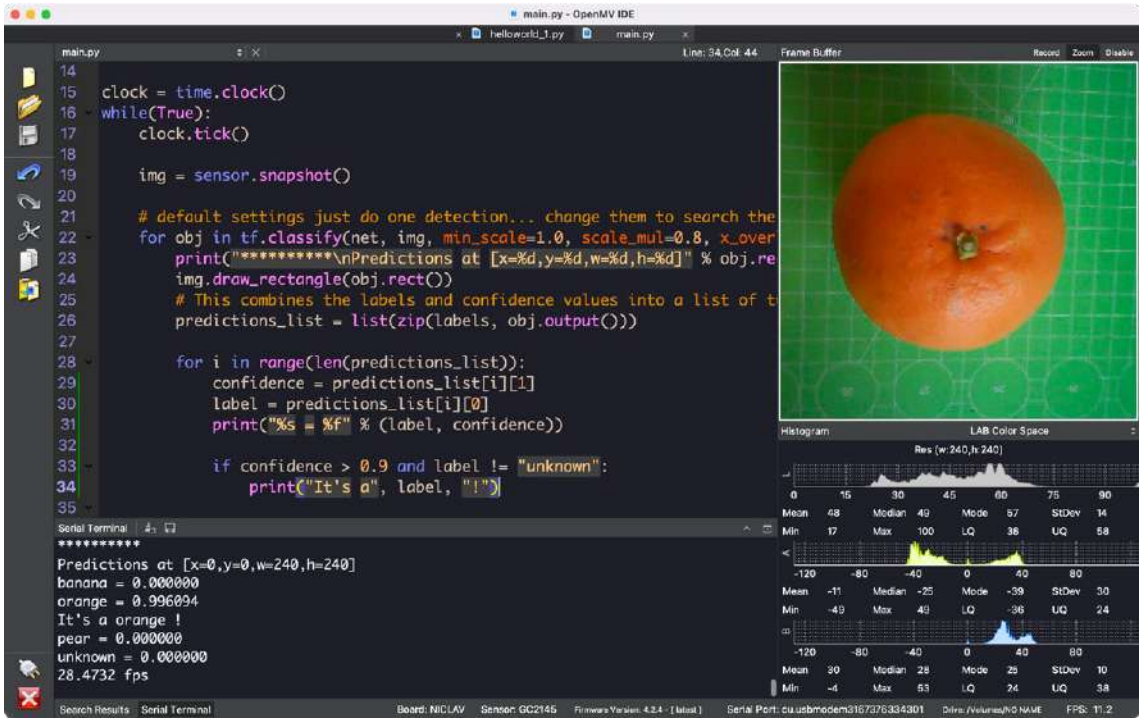
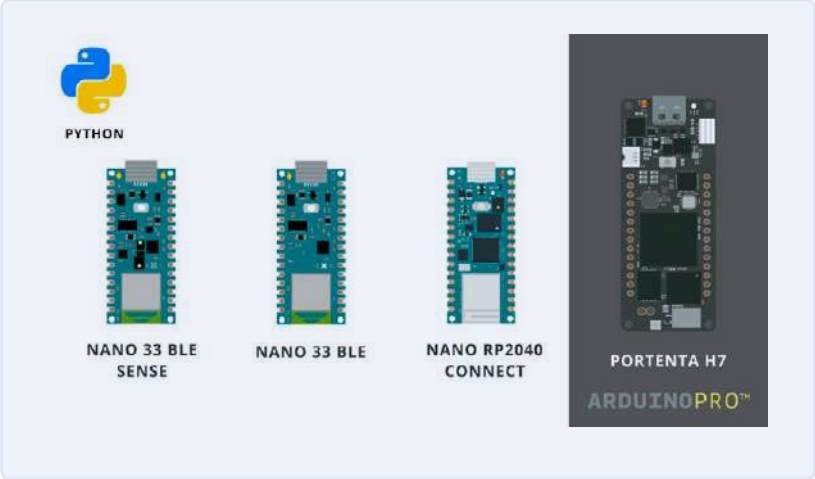


Image classification running on Arduino Nicla Vision in OpenMV.



Such MicroPython projects mainly implement state machines and evaluate sensor data.

But, the SRAM is not only used to store variables. It also stores compiled bytecode, such as imported modules, for execution by the MicroPython virtual machine (VM). That can cause issues when handling large data, such as strings, or creating and destroying many objects that leave insufficient SRAM to run the compiler. However, once the code has reached a mature state, bytecode can be precompiled and stored in flash (in the filesystem) or implemented as frozen bytecode to save even more SRAM.[5]

MicroPython is also fussier than Python when inputting code, demanding the correct use of spacing. Users also quickly learn that some default features, such as a full implementation of the standard library, are unavailable due to limited hardware capabilities.

Development Environment for Arduino MicroPython

The Arduino team selected the version of MicroPython maintained by OpenMV, as Arduino's new camera-equipped devices benefit from the Machine Vision features and the built-in support for Tensor Flow Lite provided by OpenMV [6]. As a result, users have a well-supported, mature platform and development environment. OpenMV was created to support machine vision applications on microcontrollers, which aligns well with many users' desire to create image based ML applications.

OpenMV IDE (Integrated Development Environment) provides the working area for MicroPython coders rather than the traditional Arduino IDE. It offers the core features that developers expect, such as a code development window and a serial terminal. On top, there is support for machine vision applications, such as visualization of a frame buffer and a histogram tool to visually analyse color and brightness ranges. Furthermore it has a built-in tool to upload camera pictures directly to Edge Impulse Studio to easily train a machine learning model.

But, perhaps the greatest change lies with how code is developed and deployed.

"Unlike C/C++ sketches that must be compiled and uploaded, MicroPython can be executed immediately after every change. That speeds up development significantly and brings the coding experience closer to that of Python," said Sebastian.

Another great feature is REPL, enabling short scripts to be executed or individual functions to be tested directly on the target controller.

Arduino Hardware Support for MicroPython

In total, five Arduino boards currently support MicroPython: the Nano 33 BLE and Nano 33 BLE Sense, the Nano RP2040 Connect, the Portenta H7 and Nicla Vision. Most of the boards require a firmware update to upload the MicroPython runtime into flash before getting started. As we've come to expect, not only is this process simple, but it is also well-documented[7]. Boards such as the Nano 33 have a preparation step that uses the Arduino IDE, while the others are immediately recognized by OpenMV and programmed with the necessary firmware.

The application is written as a Python script in OpenMV that is uploaded to the target board. A single click on the Play button is all that stands between the programmer and code execution.

What's Next?

What is the future for Arduino now that MicroPython is here? It is natural to worry that, with the introduction of MicroPython, the traditional C/C++ sketch may become a historical artifact. But this is neither desired nor the plan. In scenarios where real time execution is a requirement, C/C++ will still be the go to weapon.

"We are seeing growth in demand for Python support on microcontrollers, especially in industry pioneers working to develop ML applications who already use a Python stack," says Sebastian.

In fact, MicroPython support extends the available options for Arduino users rather than replacing them. Long term, C/C++ sketch developers should find suitable boards in the same form factor that they're using, that also support MicroPython.

„For many of the classic Arduino boards, a MicroPython implementation would have a very limited feature set and hence isn't a reasonable option," Sebastian adds.

Users can also continue contributing to the success of Arduino [8] with MicroPython as they have done in the past. There are 15 years of contributed C/C++ code that is still being maintained and used as drivers in combination with MicroPython. Bindings are then used to link MicroPython with this base code. For those wishing to get involved with MicroPython [10] development, OpenMV is hosted on GitHub [9], a project to which the Arduino team also contributes.

MicroPython can be seen as an addition to the current Arduino ecosystem, with its adoption driven by the success of Python as the language of preference for ML applications and interacting with cloud services. With microcontrollers increasingly hitting hundreds of megahertz and offering heaps of memory, the move to an interpreted language will be seen as an irrelevance in many cases. Of course, there are exceptions where real-time accuracy and precision are a must, and C/C++ will always be there for those who require it. For now, though, educators and students benefit significantly, allowing knowledge of Python to be transferred to microcontrollers (additionally they benefit from the simplified syntax and readability of the code). At the same time, industry developers can stick with a single language for application development.

220415-01

About the Author

Stuart Cording is an engineer and journalist with more than 25 years of experience in the electronics industry. You can find many of his recent Elektor articles at www.elektormagazine.com/cording. In addition to writing for Elektor, he hosts the monthly livestream, *Elektor Engineering Insights* (www.elektormagazine.com/eei), and he teaches Elektor Academy courses (www.elektormagazine.com/elektor-academy).

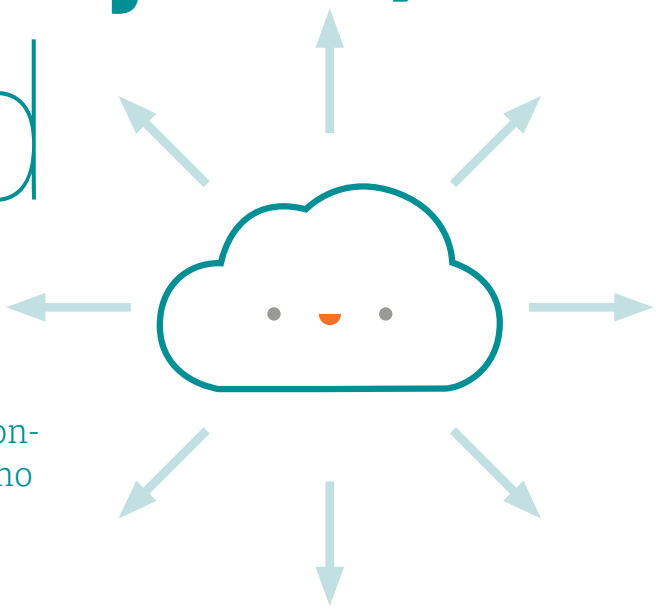
Questions or Comments?

If you have technical questions, feel free to e-mail the author at stuart.cording@elektor.com or the Elektor editorial team at editor@elektor.com.

WEB LINKS

[1] JSON, Wikipedia: https://en.wikipedia.org/wiki/JSON
[2] Regular Expression, Wikipedia: https://en.wikipedia.org/wiki/Regular_expression
[3] Python, Wikipedia: https://en.wikipedia.org/wiki/Python_(programming_language)
[4] D. George, "Micro Python: Python for microcontrollers," Kickstarter, 2016: https://www.kickstarter.com/projects/214379695/micro-python-python-for-microcontrollers
[5] "MicroPython on Microcontrollers": https://docs.micropython.org/en/latest/reference/constrained.html
[6] OpenMV: https://openmv.io/
[7] K. Soderby, "Python with Arduino Boards," Arduino, 2022: https://docs.arduino.cc/learn/programming/arduino-and-python
[8] Arduino, GitHub: https://github.com/arduino
[9] OpenMV, GitHub: https://github.com/openmv
[10] Official uPython Repo: https://github.com/micropython/micropython

Connected Projects, Simplified



By Sebastian Romero (Arduino)

Since the coining of the Internet of Things (IoT) buzzword, makers have been looking for an easy on-ramp to the connected highway of devices. Arduino Cloud has got them covered.

The rapid growth of the Internet of Things has manifested itself in billions of IoT devices that exchange data with sensors continuously, worldwide. The list of products that send their data to the Internet or can be remote-controlled through the Internet of Things is endless. Some examples are smart lamps, remote pet feeders, connected toothbrushes, thermostats, fridges, ventilation systems, smart cameras, and many more. Some devices are even deployed out in the wilderness, connected to the Internet over low-power wireless technologies such as LoRaWAN. What if you want to develop such a product yourself? Or, what if you need a solution for which there is no off-the-shelf product? Arduino has a solution for you, and it's called Arduino Cloud.

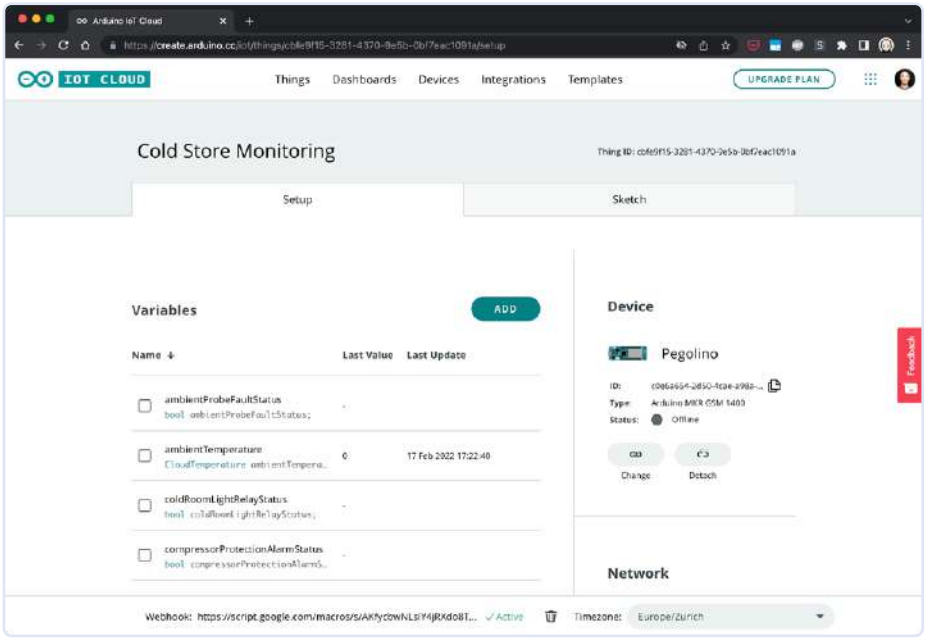


Figure 1: An example of a "thing" setup for cold store monitoring.

Core Features

Arduino Cloud consists of a variety of web services for your connected projects. In just a few minutes, you have your IoT-enabled Arduino device up and running. To do so, you can set up a data container for your project called a "thing." This will hold all data that is exchanged between the end device and the cloud. Each device needs to be associated with its corresponding thing. To store the properties of your thing, you can define variables (see **Figure 1**). Variables can be of basic types, such as integer or more advanced ones, such as "colored light." Whenever new data is available, it is synchronized automatically between the devices and the cloud. You can also choose to synchronize the variables that hold the data at regular intervals.

To visualize the data received from the end devices, and also to modify the values of the cloud variables in order to remote control the devices, you can set up dashboards. The building blocks of these dashboards are widgets. Each of these widgets can be linked to a variable from your things. It can be read-only and just display data, or

it can be a control element with which the user can interact to change the underlying variable. That way, you could, for example, turn on a fan remotely or open a locked door. Here's a list of the dashboard widgets (see **Figure 2**) that are currently available in Arduino IoT Cloud:

- Switch
- Push-button
- Slider
- Stepper
- Messenger
- Color
- Dimmed light
- Colored light
- Time Picker
- Scheduler
- Value
- Status
- Gauge
- Percentage
- LED
- Map
- Chart
- Sticky note

Dashboards created in Arduino IoT Cloud can easily be shared with others, so that others can access the things' properties in the same pleasant visual way (see **Figure 3**). Arduino Cloud also offers a cloud-based editor for your sketches (Arduino program that gets compiled into firmware). You will never have to worry about losing them or finding them in the depths of your hard drive again. Moreover, all the available libraries that are linked in the official Arduino repository are available without the need of installing them. You can work on your sketches from any computer with a web browser and upload them to your Arduino boards with the help of a little tool called Arduino Create Agent. This tool facilitates the communication between your Arduino board's serial port and your browser. Sketches created with the Arduino Cloud Editor can easily be shared with other people – and even be embedded in your website.

Home Automation

Arduino Cloud also features Alexa integration. That makes it easier than ever to implement your very own home automation setup. You could say "Alexa, lights on," for example, to turn on your Arduino-controlled light installation. Or, you could use Alexa to turn on your coffee machine, change the temperature in the living room, or change the channel on your TV, just to name a few examples. All you need to do for this to work is to install the Alexa Arduino skill [1]. Once it's installed, Alexa learns about the things that you've configured in your Arduino IoT Cloud account and will forward requests to it. You can target specific properties of your things by using the same variable names in your voice commands as you've defined in your things' setup. With Arduino's cloud infrastructure, you can create your very own Alexa-powered IoT device in just a few minutes. If you want to learn more about this, check out the tutorial, *Arduino IoT Cloud, MKR RGB Shield and Alexa integration* [2] on the Arduino Docs website.

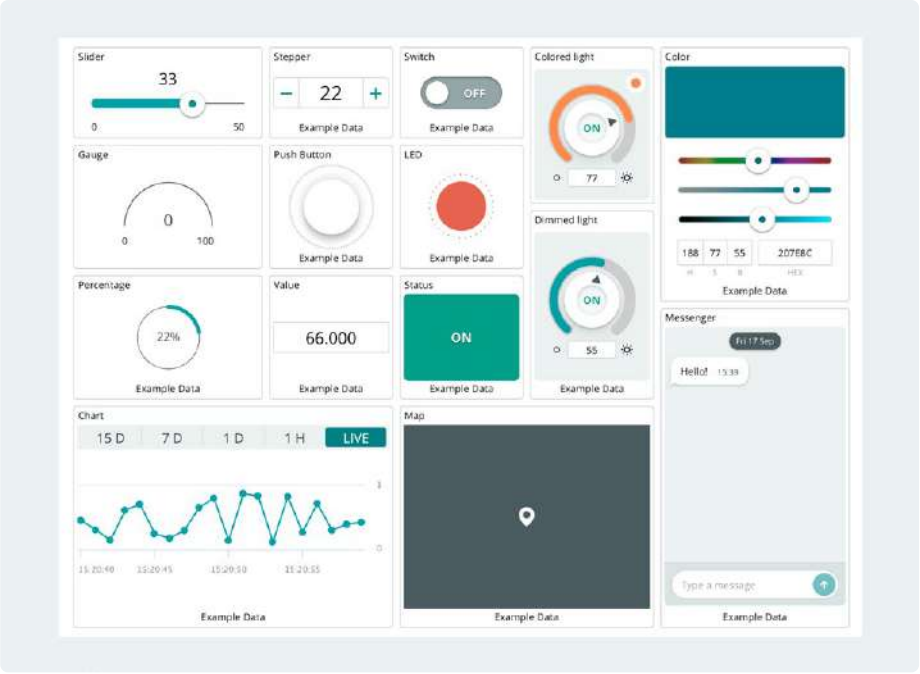


Figure 2: An overview of some of the available dashboard widgets.

Variable Synchronization

As mentioned previously, only one device can be assigned to a thing at a time. However, you may want to allow your IoT devices to communicate with one another. The way to do this with Arduino Cloud is to

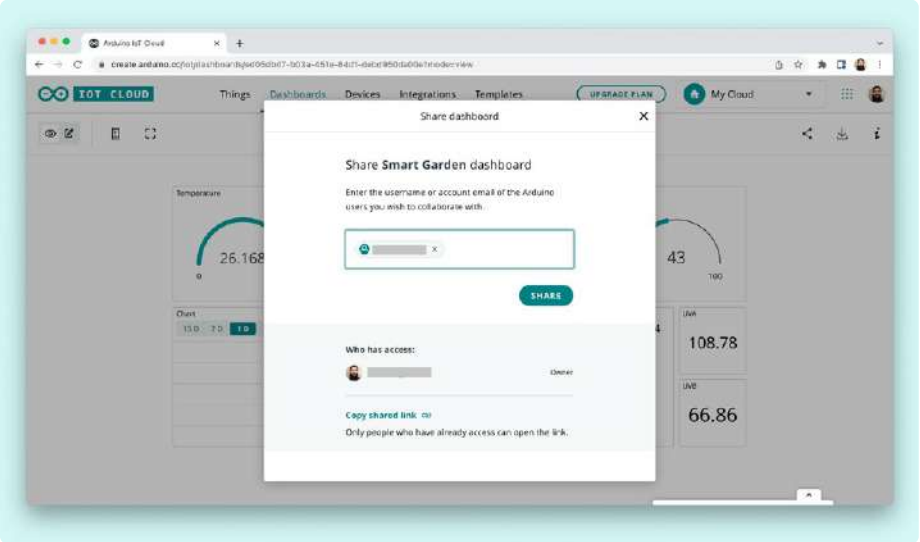


Figure 3: Dashboards can be shared with other people who have an Arduino account.

use synchronized variables. In the variable settings, you can instruct it to mirror the value of another variable (see **Figure 4**). This allows you to have a common state between multiple devices. For example, you could add a boolean variable named `enabled` to multiple things. This would allow enabling or disabling some feature on the corresponding devices based on the value of that variable. You could also set up some shared configuration parameters for your IoT devices through variables, so you can change the settings of all these devices at once without uploading new firmware.

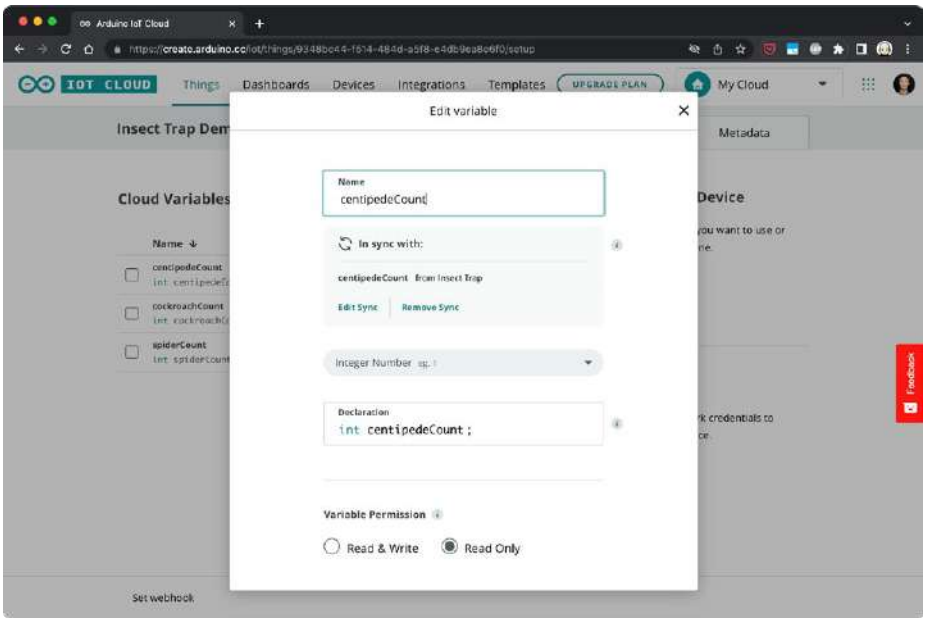


Figure 4: Variables can be synchronized with one or more other variables.

Webhooks

Arduino IoT Cloud offers the ability to interact with third-party services such as Google Apps Script [3], Zapier [4], IFTTT [5] or Google's Cloud Functions [6]. This is achieved via webhooks. Whenever new data is available in Arduino IoT Cloud, the webhook gets triggered, so the third-party service gets notified. This allows, for example, implementing a custom notification service and sending out a warning e-mail if the value of a variable exceeds a certain threshold. In a real-world scenario, this could allow you to take action if your IoT remote monitoring device notices that the temperature in your freezer cell is rising, which would indicate that the machine is malfunctioning (see **Figure 5**).

Templates

A new feature that makes the setup of your cloud projects with Arduino Cloud much more efficient is templates. There are templates for things and also for dashboards. They allow you to duplicate a thing, including its properties and any associated dashboard. For example, if you have a multitude of sensor nodes that must collect the same type of data, no longer will you have to manually set up each of these things. Instead, you can use the same template for all of them. And, since Arduino is all about sharing knowledge, you can now share the template of your IoT project with the world. They can replicate your setup with just one click and get it up and running in minutes. If you want to try

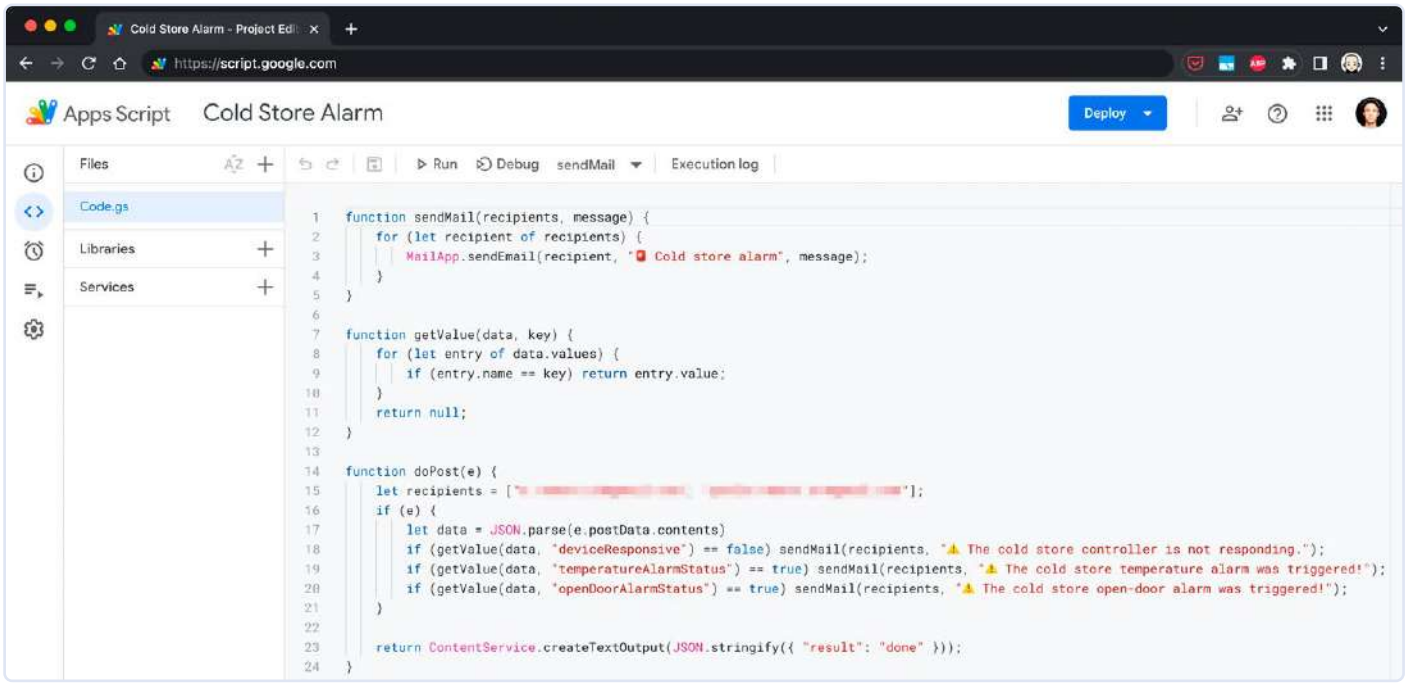


Figure 5: Google App Script example that sends a notification when there's an issue with the cold store.

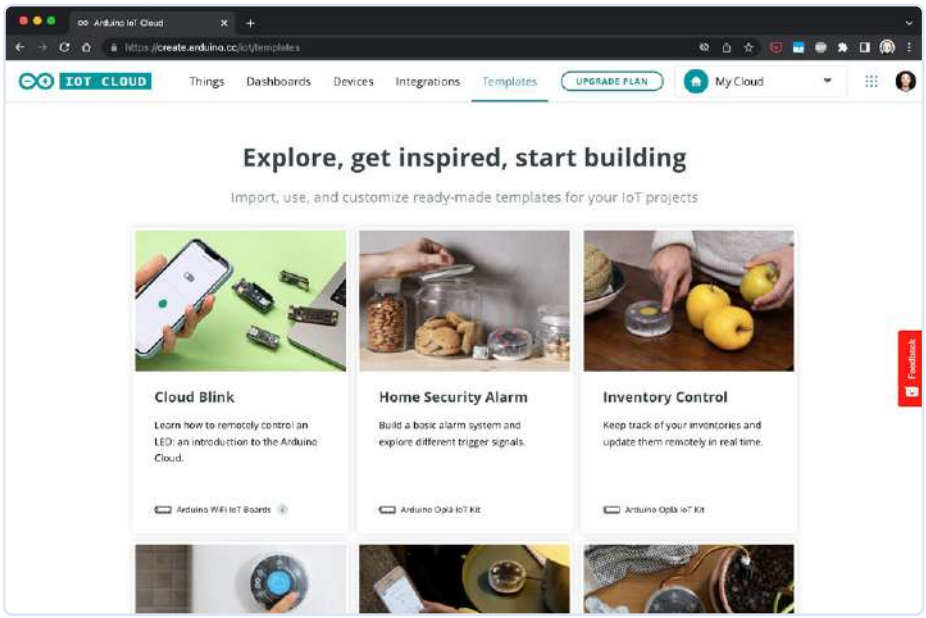


Figure 6: Some examples of the available demo projects in Arduino IoT Cloud.

this out, there is a collection of inspirational example projects that include the corresponding template, available right inside Arduino Cloud (see **Figure 6**).

Arduino Cloud CLI

If you only need to set up a few things for your own projects, you can do that easily using the Arduino IoT Cloud web interface. However, if you want to configure a multitude of IoT devices, this option is a bit restrictive. For that reason, Arduino created a command line tool [7] that makes it very easy to automate this task. You can also use it to list existing things, dashboards and devices, and modify them. You can add tags, which allows you to create groups of devices that can be managed in a unified way. You can use it to provision devices for Arduino Cloud, to assign or reassign a device to a thing, to extract properties from a thing into a template, to perform over-the-air updates of your devices (see **Figure 7**), and much more. In combination with templates, this tool enables you to scale your IoT projects with very little effort.

In case you need access to these functions inside your custom tooling setup, there is also the option of using the REST API [8] that comes with a client for JavaScript, Python, and Golang.

Arduino IoT Remote

You may not always have access to a fully-fledged computer when you're on the go, but you may still want to inspect what's going on with your IoT devices. Maybe you would like to check on the status of the cooling system in your factory, or you want to turn on your irrigation system remotely, or possibly just release some food for the dog, using your IoT pet feeder. For all these use cases and more, there is the Arduino IoT Remote mobile app, which brings all of your Arduino IoT Cloud dashboards to

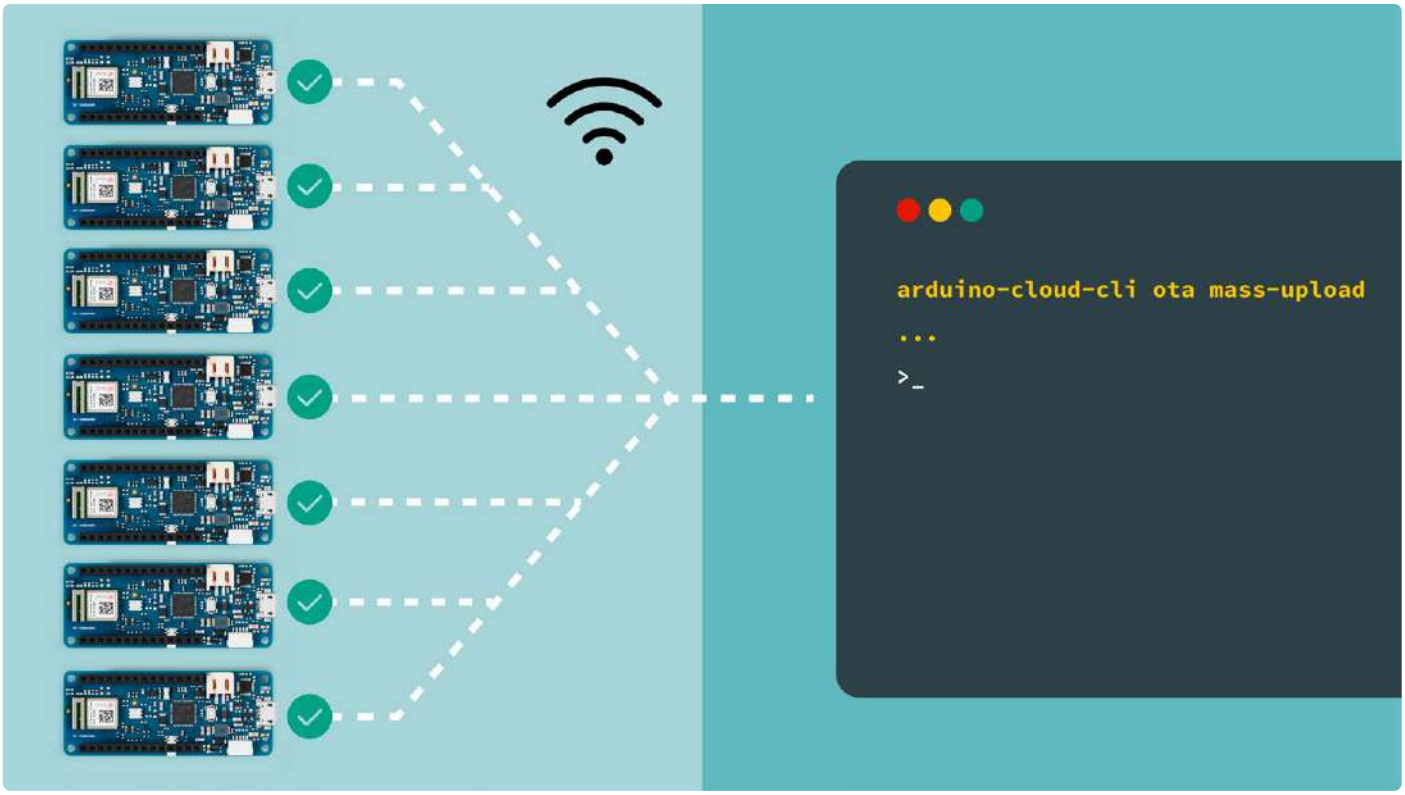


Figure 7: With the Arduino Cloud CLI, you can update your devices simultaneously, over the air.

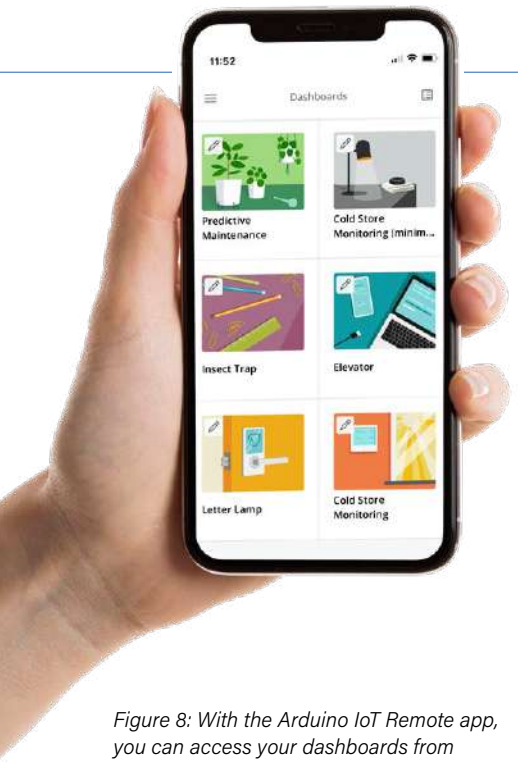


Figure 8: With the Arduino IoT Remote app, you can access your dashboards from anywhere in the world.

your fingertips (**Figure 8**). With this app, you can access, monitor, or control your IoT projects from anywhere in the world, using just your phone. The app is available for both iOS [9] and Android[10].

Things Stack Integration

Depending on where you deploy your IoT devices, you need to choose a different type of connectivity to connect your devices to Arduino Cloud. If it's a stationary setup and Wi-Fi is available, that is a convenient option. Sometimes this is not possible, so you can opt for a cellular network connection. However, the challenge is that they're quite energy-hungry. If you want to power your device with a battery, it won't last very long when using a cellular network. For these use cases, and also if no cellular network is available, there are Low Power Wide Area Networks (LPWANs), such as LoRaWAN. The Things Network [11] is one of the most popular backends to manage the data from LoRa-powered IoT devices. There is a huge community of people who share their LoRaWAN gateways with others. As a result, there is good coverage in many locations. Alternatively, you can set up your own gateway or use the network from an official provider. Getting data from your LoRaWAN devices to Arduino Cloud previously required a significant configuration effort. Now, there is an official integration between The Things Stack [12] and Arduino Cloud that makes this a breeze. When you configure your LoRaWAN-capable Arduino board in Arduino IoT Cloud, it automatically creates a new application on a dedicated Things Stack instance and adds your device to it. No manual configuration is needed. Previously, you had to think about how to marshall and unmarshall your data when transmitting it from a device over the LoRaWAN backend to another platform. If you use Arduino Cloud and the corresponding library, all of that happens automatically.

Because Arduino's Things Stack instance has an SLA of 99.9%, you will never have to worry about the uptime of your LoRaWAN solution again. Eventually, you see your IoT devices' data displayed neatly on your dashboards, no matter if it came from a LoRa device, a Wi-Fi powered one, or one connected to a cellular network.

Security

Security is a very important topic for IoT. After all, IoT devices deal with personal and sensitive data, so Arduino has put a focus on security on different levels. Arduino's connected boards are equipped with a crypto chip. This so-called secure element provides an IC-level root of trust that cannot be compromised. The secrets used to set up an encrypted connection are safely stored on a dedicated chip and cannot be extracted. Your Arduino IoT Cloud sketches and project data are stored in AES 256-bit encrypted data stores.

For application-level security, Arduino recently introduced role-based access to its cloud platform. This gives you fine-grained control over who can access what. This is especially useful in a business environment where people with different roles need to access the same data in different ways.

220569-01

Questions or Comments?

Do you have any questions or comments relating to this article? You can contact the Elektor team at editor@elektor.com.

About the Author

Sebastian Romero, head of content at Arduino, is an interaction designer, educator and creative technologist with a weakness for humans. With his team, he is responsible for crafting enthralling learning experiences to help millions of engineers, designers, artists, hobbyists and students to innovate.

WEB LINKS

[1] Arduino Alexa Skill: https://amazon.com/Arduino-LLC/dp/B07ZT2PK2H
[2] Arduino IoT Cloud, MKR RGB Shield and Alexa integration: https://elektor.link/ArduinoIotRGBAlexa
[3] Google Apps Script: https://google.com/script/start/
[4] Zapier: https://zapier.com/
[5] IFTTT Maker Webhooks: https://ifttt.com/maker_webhooks
[6] Google Cloud Functions: https://cloud.google.com/functions
[7] GitHub - Arduino Cloud CLI: https://github.com/arduino/arduino-cloud-cli
[8] Arduino IoT Cloud API: https://arduino.cc/reference/en/iot/api/
[9] Arduino IoT Cloud Remote for iOS: https://elektor.link/arduinoiot4ios
[10] Arduino IoT Cloud Remote for Android: https://elektor.link/arduinoiot4android
[11] The Things Network: https://thethingsnetwork.org/
[12] The Things Stack: https://thethingsindustries.com/stack/

Introduction to TinyML

Big Is Not Always Better

By José Bagur (Guatemala)

One of the fastest growing areas of deep learning is tiny machine learning (TinyML). TinyML is a cutting-edge field that brings machine learning models into low-power and low-cost computing devices, such as microcontrollers. This article will describe why TinyML shows us that big is not always better.

What Is TinyML?

TinyML is a subfield of machine learning (ML) focused on developing models that can be executed in real-time, low-power, and low-cost embedded devices [1]. Development of TinyML models follows the typical ML process as shown in **Figure 1**, with the difference that *inference* takes place on embedded devices rather than on traditional computational devices or cloud-based services.

Usually, TinyML models use data collected from Internet of Things (IoT) devices; this data is used to train a model in services (usually cloud-based) that extract knowledge patterns from the dataset. These knowledge patterns are then packaged into a model that considers the embedded device's computational resources, such as memory and processing power [1]. The resulting model can then be deployed into embedded devices that evaluate new sensor data in real time and *in situ*, without using external resources such as cloud-based services. The power requirements of TinyML applications are usually in the mW range; this feature enables battery-powered devices to join the ML universe (**Figure 2**).

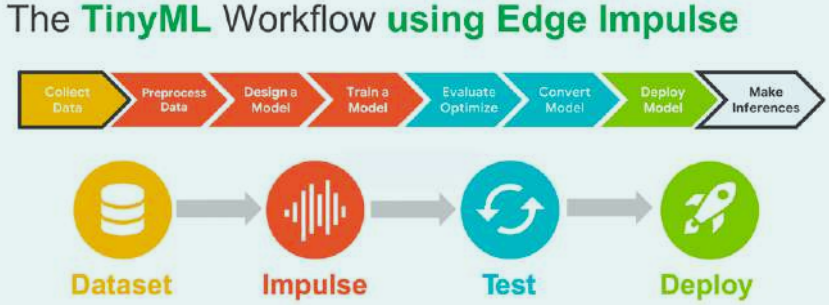


Figure 1: Typical TinyML workflow. (source: TinyMLedu)

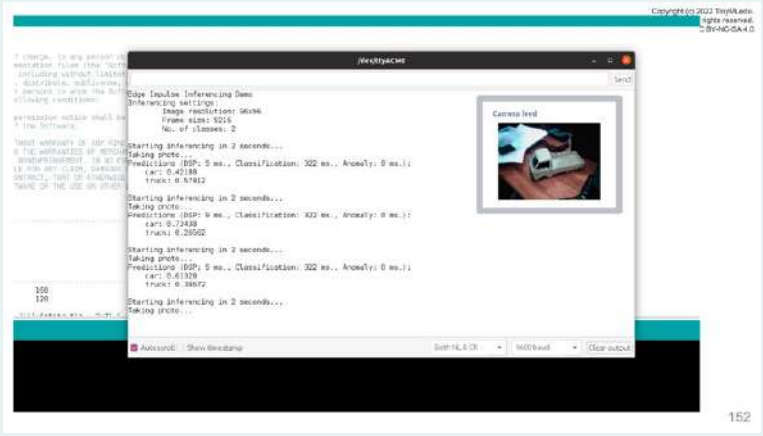


Figure 2: Inference process in a embedded system. (source: TinyMLedu)

TinyML Main Features

Now that we know what TinyML is, let's list its main features:

- **Latency**: since inference is made directly on the embedded device, latency is low in TinyML applications.
- **Power consumption**: TinyML models consider embedded devices' constraints; they can run in low-power devices such as microcontrollers, whose power requirements are usually in the mW range. This means that battery-powered devices can be used for TinyML applications.
- **Bandwidth**: Since the model runs directly in the embedded device, collected data doesn't have to be sent to an external service; this means less Internet bandwidth is used.
- **Privacy**: Data used in TinyML models is recollected and analyzed in real-time and *in-situ*; data is not sent or shared to external services at any time.

TinyML Applications and Use Cases

TinyML has the enormous potential to eliminate the bottleneck of IoT applications: data. Since inference is made locally, TinyML enables the *Internet of Thinking Things* (IoT2) era, meaning a universe of improved and new applications.

Some examples of real-world applications and use cases of TinyML are the following:

- **Power saving/optimization**: Power consumption can be drastically improved within a TinyML model by providing peak power at the lowest possible consumption. This can be translated into the electric or even mechanical world with the help of TinyML.
- **Predictive natural disaster (early stages)**: Early stages of natural disasters could be foreseen to prevent significant damages caused to the current living infrastructure using several devices with a TinyML model injected into them in a mesh network. This can be achieved by learning a broad spectrum of signals and patterns that are emitted in environments of interest.

Figure 3: The Arduino Tiny Machine Learning Kit includes a Nano 33 BLE Sense board.

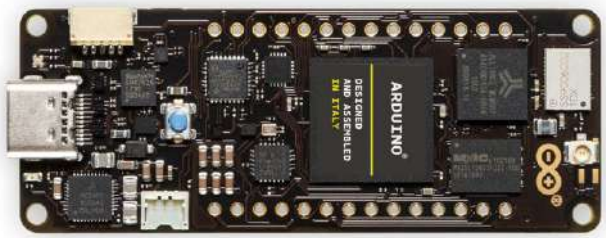


Figure 4: Arduino Portenta H7.

- **Predictive abnormal health response**: Abnormal health issues could be detected earlier to avoid spontaneous internal injury or death. For example, heart conditions could be predicted in enough time in advance to seek healthcare. The TinyML model would be realized in a wearable device that can be connected to external services for emergencies.

Arduino and TinyML

Arduino offers several hardware options and software libraries that can be used for TinyML applications. Let's talk about two great boards that can be used as a good starting point for developing TinyML applications: the Arduino Nano 33 BLE Sense and the Arduino Portenta H7.

The Arduino Nano 33 BLE Sense (**Figure 3**) uses the nRF52840 from Nordic Semiconductor; this is a 32-bit ARM Cortex-M4F microcontroller running at 64 MHz, with 1 MB of Flash and 256 KB of RAM [2]. The Nano 33 BLE Sense has several onboard sensors that can be used in many TinyML applications:

- Accelerometer, gyroscope, and magnetometer (LSM9DS1)
- Microphone (MP34DT05)
- Gesture, light, and proximity (APDS9960)
- Barometric pressure (LPS22HB)
- Temperature and humidity (HTS221)

The Arduino Portenta H7 (**Figure 4**) is a high-performance, industry-rated board designed for demanding applications. The Portenta H7 uses the STM32H747 microcontroller from STMicroelectronics; this microcontroller combines a Cortex-M7 core, running at 480 MHz, and a Cortex-M4 core, running at 240 MHz. The Portenta H7 can run high-level code along with real-time tasks simultane-

ously. For example, with the Portenta H7, we can run Arduino compiled code along with MicroPython compiled code and have both cores communicate via the Remote Procedure Call (RPC) mechanism [3].

As for software, the Nano 33 BLE Sense and the Portenta H7 can use the TensorFlow Lite software framework for developing TinyML models; the Artificial Intelligence for Embedded Systems (AIfES), developed by the Fraunhofer Institute for Microelectronic Circuits and Systems, is also a great software framework optimized for embedded systems. Edge Impulse, a Cloud-based ML service, is also gaining popularity among the community – it supports Arduino boards, the Nano 33 BLE Sense, and the Portenta H7.

Further Resources

Learning about an emerging field can be difficult, but for TinyML, there are great resources online:

- Pete Warden and Daniel Situnayake's book, *TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power*, [4] is a must-read and a good starting point for the TinyML universe.
- The Professional Certificate in TinyML [5] from Harvard University available at edX. This free, online four-course specialization dives deeper into the TinyML universe.
- The Introduction to Embedded Machine course [6] from Edge Impulse available at Coursera. This free online course also gives a broad overview of how ML works, how to train TinyML models using Edge Impulse, and how to deploy those models in microcontrollers.
- The TinyML Open Education Initiative (TinyMLedu) [7]. This initiative is an international group of academics and industry professionals working to improve global access to educational materials for the cutting-edge field of TinyML.
- The TinyML for Developing Countries (TinyML4D) [8]. This initiative is working to develop content for a network of researchers and practitioners focused on enabling innovative solutions for the unique challenges developing countries face.

The IoT2 Era

TinyML is an emerging field that studies ML models that can be deployed in small, low-cost, and low-power devices such as microcontrollers. With the versatility of hardware and software tools such as the Arduino ecosystem, and software frameworks such as Tensorflow Lite and Edge Impulse, the IoT2 era is now possible.

220573-01

About the Author

José Bagur is a lecturer and researcher at Universidad del Valle de Guatemala (UVG). He studied Mechatronics Engineering at UVG before obtaining a master's degree in IoT from the University of Salamanca. With a particular interest in space-related projects, José's research is focused on developing open-source low-cost nanosatellite hardware. José also works for Arduino as a content creator.

Questions or Comments?

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

Related Products

Looking for the main items mentioned in this article? Arduino and Elektor have you covered!

- **Arduino Tiny Machine Learning Kit (SKU 19943)**
  www.elektor.com/arduino-tiny-machine-learning-kit
- **Arduino Pro Nicla Vision (SKU 20152)**
  www.elektor.com/20152
- **Arduino Portenta H7 Development Board (SKU 19351)**
  www.elektor.com/19351

WEB LINKS

[1] M. Zennaro, "TinyML: Applied AI for Development Challenges with Machine Learning in Developing Countries.": https://sdgs.un.org/sites/default/files/2022-05/2.1.3-9-Zennaro-TinyML.pdf
[2] Nano 33 BLE Sense: https://docs.arduino.cc/hardware/nano-33-ble-sense
[3] Portenta H7: https://docs.arduino.cc/hardware/portenta-h7
[4] P. Warden & D. Situnayake, *TinyML* (O'Reilly Media, 2019): https://www.oreilly.com/library/view/tinyml/9781492052036/
[5] Tiny Machine Learning Open Education Initiative (TinyMLedu): https://tinyml.seas.harvard.edu/
[6] TinyML4D: TinyML for Developing Countries: http://tinymledu.org/4D
[7] Tiny Machine Learning (TinyML) Professional Certificate: https://www.edx.org/professional-certificate/harvardx-tiny-machine-learning
[8] Introduction to Embedded Machine Learning: https://www.coursera.org/learn/introduction-toembedded-machine-learning

awear of your air

Imagine the possibilities: what new experiences can be created when integrating advanced AI and powerful sensors in one of the most iconic outdoors jackets? Start sensing and interacting with the world around you like never before.

We put a Nicla Sense ME, the new sensory brain from Arduino, into a K-Way jacket, powered by Edge Impulse AI, to unleash a new generation of smart clothing.

The Nicla Sense ME is a tiny, low-power tool that sets a new standard for intelligent sensing solutions. With the simplicity of integration and scalability of the Arduino ecosystem, the board combines four state-of-the-art motion and environmental sensors from Bosch Sensortec.

A CHANCE TO GET 1 ARDUINO X K-WAY JACKET

Imagine a board packed with sensors, mounted on an iconic casual outdoor jacket – what would you do?

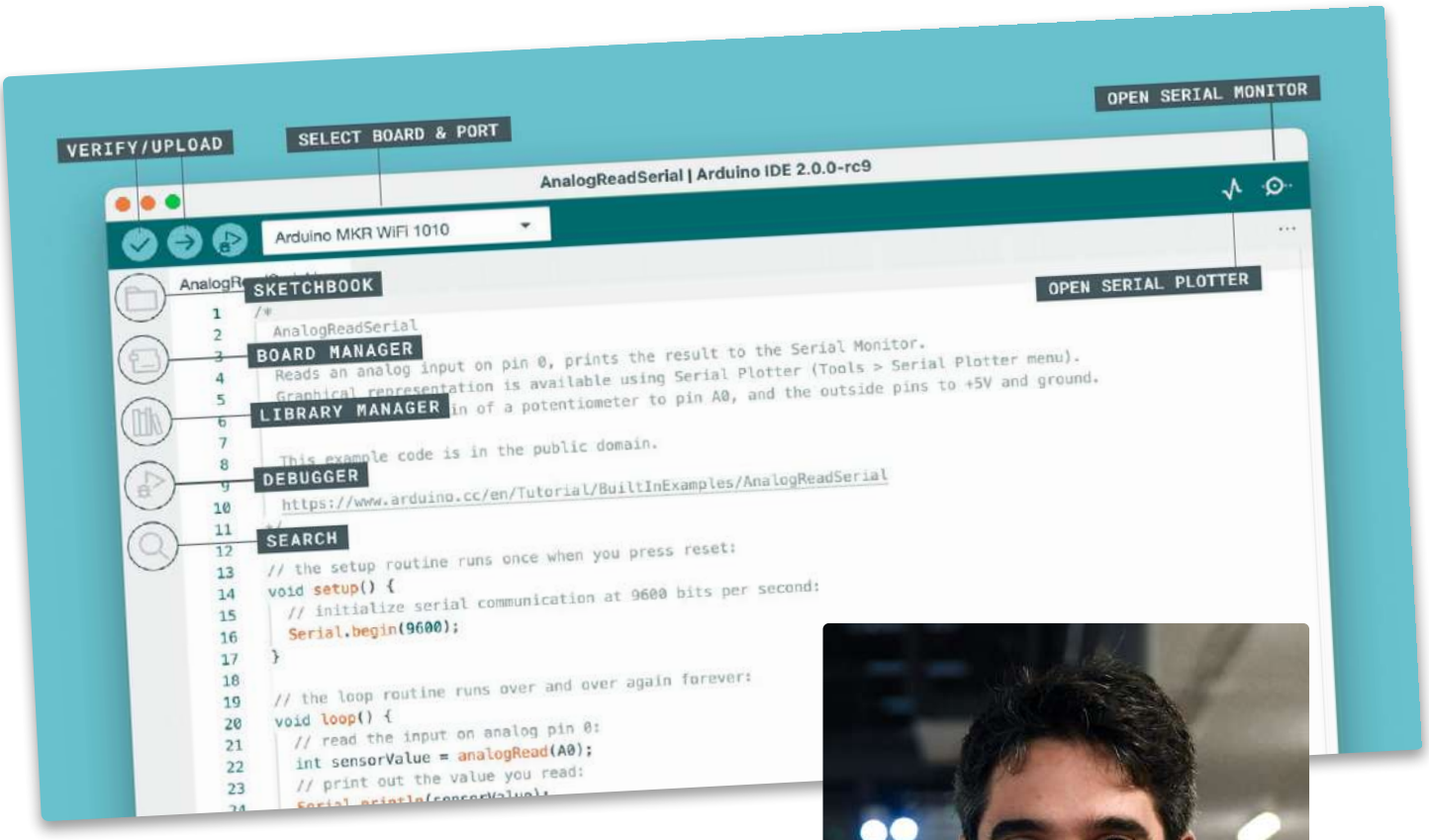
specs

Microcontroller:	64 MHz Arm® Cortex M4 (nRF52832)
Sensors:	BHI260AP - Self-learning AI smart sensor with integrated accelerometer and gyroscope.
	BMP390 - Digital pressure sensor.
	BMM150 - Geomagnetic sensor.
	BME688 - Digital low power gas, pressure, temperature & humidity sensor with AI.
Connectivity:	Bluetooth® 4.2

WIN 1

You now have a chance to get your hands on this unique jacket, as Arduino and K-Way have reserved 1 of only 100 produced (not available for general sale) for readers of the Elektor Arduino Edition.These jackets are designed to explore ingenious applications for monitoring people's personal environments, taking wearables right to the edge! So, to participate, simply pitch* your idea here: **elektormagazine.com/k-way-arduino**

(*Promotion ends 31st January 2023.)



Writing **Arduino Sketches** Just Got **Better**

Alessandro Ranellucci

By Stuart Cording (Elektor)

Arduino IDE 2.0 provides a break with the old without breaking the Arduino experience. Curious about the details? Arduino's Alessandro Ranellucci offers some insights.

For most Arduino users, their first interaction with a programming language was the Arduino integrated development environment (IDE). With its green borders and code editing window, newbies have spent hours decrypting the output messages as they have built and downloaded their sketches. Since its launch in 2005, the IDE has grown in capability, supporting more boards, providing better access to shared libraries, and adding a plotter to support graphing of data. However, as the number of professional developers has grown and makers have improved their skills, one glaring omission is the lack of debugging support.

To find out more, I recently caught up with Alessandro Ranellucci, Head of Maker Business, Open Source & Community at Arduino.

Arduino IDE 2.0 retains the clarity of its forebear but offers a host of new features suited to professional developers and experienced makers.

A Complete Rewrite

Back in 2018, the Arduino team made an important step that would change not only the capability of the IDE, it would refresh the entire codebase. IDE 1 was written in Java as a monolithic block of code (**Figure 1**). This made it challenging to get support from enthusiastic community supporters. Furthermore, Java was becoming increasingly challenging to support as a desktop application on operating systems and app stores, something which led to much time spent on resolving compatibility issues.

"We started by refactoring the entire toolchain, creating a command-line interface (CLI) called Arduino CLI," Alessandro explains. "Written in Golang and TypeScript, it exposes all the functionality of the old IDE as well as all the new features."[1]

Thanks to its modular approach, it is easier to pair with the developer's preferred IDE, which made Arduino's new IDE possible too. Furthermore, the CLI is something that professional developers strongly welcome, enabling them to easily integrate the Arduino environment with other popular components used in software development, such as continuous integration/continuous deployment (CI/CD) tools.

Refreshing the IDE

With progress made on Arduino CLI, it was time to explore how best to implement the visual interface. While beginners were satisfied with the original editor, those with more experience missed some of the little extras provided as standard in the IDEs from the big semiconductor and development tool vendors. As code grows, it can become challenging to navigate, so the ability to toggle comments was earmarked as a desired feature.

Most IDEs also implemented auto-indentation, keeping code legible, and auto-closing brackets, which avoids common code entry mistakes. But above all, support for debugging was a clear community desire. For experienced embedded software developers, the ability to explore variables' content and follow code execution is considered a standard IDE capability. By comparison, IDE 1's slow edit-build-download process to debug a single edit was frustrating for many.

"We selected Theia with Electron as the software stack for the IDE," says Alessandro, "as they are open-source projects that enable us to provide flexible, multi-language cloud and desktop IDE support." This would ensure that the more than 60 language translations could be supported. In 2021, the first beta of IDE 2.0 was shared with the community.[2]

The Power to Debug

Some Arduino boards, such as those featuring Arm Cortex-M-based microcontrollers (MCU), already have the hardware required to support debugging. Often labeled as an Embedded Debug (EDBG) chip, it communicates with the target MCU's debug interface but also provides the virtual COM serial interface over USB. Others have a header for third-party debuggers, such as J-Link.[3] Both are typically supported by another open-source software project, GDB (GNU Project Debugger).[4]

"Debugging was a clear requirement, especially by the professional developer community," recalls Alessandro. This also aligns with the launch of the Arduino Pro [5], a new range of hardware supported by secure connectivity to offer an all-in-one IoT platform. However, some things remain unchanged. Developers still start with the traditional .ino sketch, although, unlike other IDEs, such files are recognized as executable code.

Figure 1: The well-known Arduino IDE 1.0, based on Java.

A Tour of IDE 2.0

The new Arduino IDE [6] retains the instantly recognizable color scheme and familiar layout (**Figure 2**). However, upon closer inspection, a new "Start Debugging" button at the top and panel along the left-hand side have appeared. This is where developers get access to the internals of the MCU and an overview of what's happening in their code. Otherwise, the menu is more or less the same but with a few extras in some drop-downs.

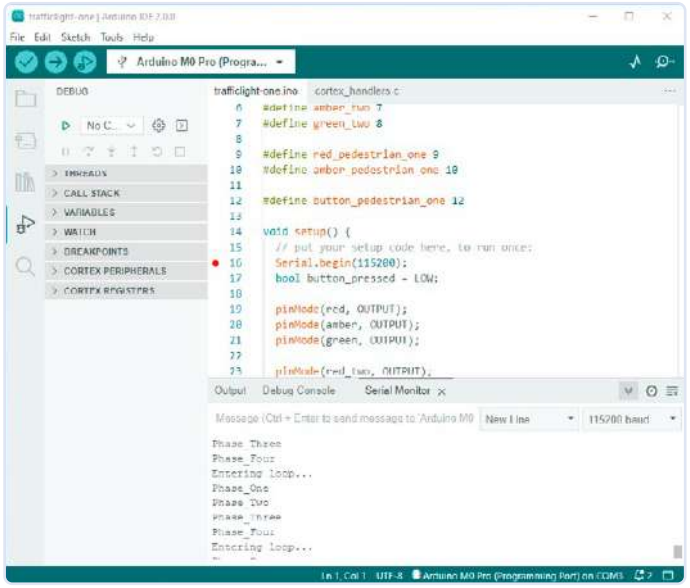


Figure 2: Familiar, but different. Arduino IDE 2.0 retains the clarity of its forebear but offers a host of new features suited to professional developers and experienced makers.

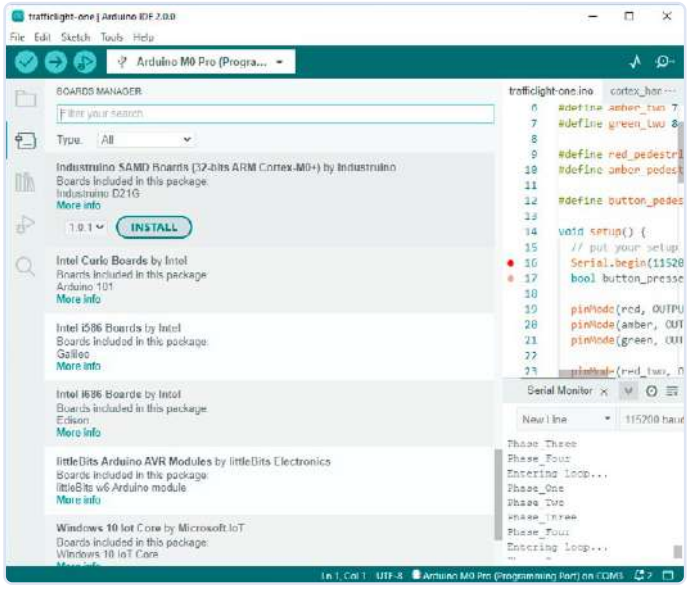


Figure 3: The board manager is familiar, but its location has moved.

A narrow panel provides access to some IDE components using icons on the left-hand side. The board and library manager are two examples, with these utilities embedded into a window to the left of the text editor (**Figure 3** and **Figure 4**). Despite the visual changes, users will be happy to know that the overall functionality is essentially the same. This window is also where the debug capability can be selected and viewed. The serial monitor now opens within the main IDE window as a tabbed output, alongside where compilation and download messages appear, rather than in separate windows. For data visualization, the serial plotter function is still available, opening in its own window (**Figure 5**).

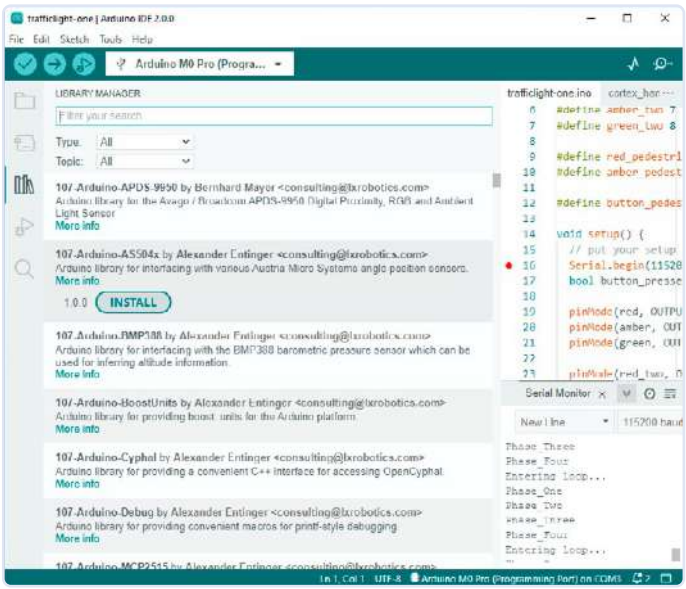


Figure 4: The library manager retains the same functionality but is docked alongside the editor.

Figure 5: Data visualization is supported by an updated Serial Plotter window.

The editor has received some upgrades too. "Some major improvements include code completion, with the environment offering potential variable names when typing," explains Alessandro. "We've also added code navigation, enabling developers to quickly jump to the line of code where a function is defined." To help those developing IoT applications, there is also a firmware updater for Wi-Fi boards and SSL root certificates.

One of the biggest challenges with today's embedded software development tools is keeping abreast of the library versions in use. The other is being able to build the same project on a different computer, something that typically goes badly wrong as file paths either don't exist or are mangled when switching between operating systems. Arduino have resolved this by adding a metafile that stores the version of all libraries and board packages in use. Developers can move workstation, share their project, and even make projects available on GitHub and still be able to compile it. What hasn't yet been optimized is the build time, meaning all source code still gets compiled even though only one line of code may have changed. Alessandro tells me that "the Arduino CLI can support recompiling only changed files," so, hopefully, the IDE will benefit at some point too.

Where to Next for Arduino IDE?

The first official release of Arduino IDE 2.0 was announced in September 2022 after several release candidates that allowed the community to test its features. However, like IDE 1.0, the project will never finish — it just continues to morph and develop through feedback from users.

"Thanks to the modular approach, it will now be easier for the community to contribute to the IDE's development," Alessandro informs me.

In fact, this is how development is working today. Arduino only has six full-time internal developers on the project. Hundreds of contributors have ably supported them, tens of whom have been active in shaping new features and testing. As the project is hosted on GitHub, anyone can contribute with code development, reporting issues, or suggesting improvements.

As an open-source veteran, I ask Alessandro what the biggest challenge is in running a project where anyone can contribute. "Keeping everyone focused on the mission of Arduino," he replies. "As a technology enabler, we must remember that Arduino IDE goes beyond the Arduino boards — we enable an entire industry."

220520-01

About the Author

Stuart Cording is an engineer and journalist with more than 25 years of experience in the electronics industry. Find many of his recent Elektor articles at www.elektormagazine.com/cording. In addition to writing for Elektor, he hosts the monthly livestream, Elektor Engineering Insights (www.elektormagazine.com/eei), and he teaches Elektor Academy courses (www.elektormagazine.com/elektor-academy).

Questions or Comments?

If you have technical questions, feel free to e-mail the author at stuart.cording@elektor.com or the Elektor editorial team at editor@elektor.com.

About Alessandro Ranellucci

(Head of Maker Business, Open Source & Community)

Alessandro Ranellucci, Head of Maker Business, Open Source & Community, joined Arduino in 2020 to lead the products and strategy for makers, as well as the open source ecosystem. He previously worked as head of open source and was a member of the digital transformation task force for the Italian government. He's the curator of Maker Faire Rome and main author of Slic3r, a popular open-source tool for 3D printers.

WEB LINKS

[1] The Go Project: https://go.dev
[2] Arduino, "Announcing the Arduino IDE 2.0 (beta)," March 1, 2021: https://blog.arduino.cc/2021/03/01/announcing-the-arduino-ide-2-0-beta
[3] Segger, J-Link Debug Probes: https://segger.com/products/debug-probes/j-link
[4] Free Software Foundation, "GDB: The GNU Project Debugger": https://sourceware.org/gdb
[5] Arduino Pro: https://arduino.cc/pro
[6] Arduino Downloads: https://arduino.cc/en/software

Get to Know Arduino

Why is Arduino so successful? What's on Massimo Banzi's workbench? What inspires David Cuartielles? Where did the name Arduino come from? Our friends at Arduino address all of these questions and more.

Fabio Violante

In June 2022, Arduino announced the completion of a $32 million Series B funding round. Fabio, can you explain how the funding has helped Arduino to date? What are your plans for 2023?
C. J. Abate (United States)

Thanks. It was a very successful raise, and we are aligned with our investors in terms of direction and what it will take to get there. The level of funding received from the investors reflects what we were aiming to raise to enable us to deliver on our strategic goals to capitalize on the new business opportunities evolving as a new generation of engineers enters the workforce. The funding had two main objectives:
Firstly, technology — to expand and refine the Arduino Platform through R&D investments. This means not only Hardware and IDE, but also firmware and cloud services for our audiences and community. These are being developed to support IoT and AI use cases, especially for enterprise users. We are delivering more powerful hardware and several cloud features to safely manage fleets of devices with new interaction paradigms. We will also invest in professional-grade open-source content and libraries that are the basis for lowering the barrier to entry for many users. Makers and educators are already benefiting from these investments. Much more is in the making and will be released in the near future. Secondly, market expansion — to build a solid reputation in the enterprise by leveraging on the thousands of companies that have already embraced Arduino, not only for prototyping, but also explaining the strength of the platform and gathering feedback for further developments. — Fabio Violante

What has been the most difficult challenge associated with running Arduino during the COVID-19 pandemic?
Raoul Morreau (The Netherlands)

Naming the most difficult challenge Arduino faced during the COVID-19 pandemic is a challenge in itself, as it was such a rapidly changing set of circumstances, especially in Italy, where our R&D team is based, and which was particularly hard hit at the beginning of the pandemic. As Arduino boards were used in multiple medical and sanitation applications including the design and production of open-source ventilators as well as smart hand sanitizer dispensers, we met the necessary criteria to continue running our operations in Italy despite the lock-down. As the pandemic continued, our biggest challenge was probably the same as that faced by many companies – keeping employees who were based all around the world motivated and engaged during a time in which they were facing their own personal challenges from the health of themselves and their families to keeping upbeat during some very restrictive lockdown rules. The lockdown made it more difficult to prototype new products, resulting in several delays. Subsequent increases in shipping costs and component shortages with the associated supply chain disruptions have further challenged the R&D pipeline. Overall though, I was very proud of the way the Arduino team responded to the challenges over the two years, coming up with many ways to remain positive and engage with the global Arduino community. Massimo hosted the live Bar Arduino show to share the great things people could do with Arduino around the home whilst they were locked down, and our Arduino Education team designed a completely new product — The Student Kit — within two months, to enable students to study Arduino individually in a remote learning environment. This was no easy challenge, because the Arduino Education programs were originally focused on shared learning in a classroom setting. It proved a great success though, being adopted by schools and parents around the globe. — Fabio Violante

Why did you name the company Arduino? — Udo Bormann (Germany)

We were about to register our project on an ancient open-source repository called BerliOS, in Germany. Back then, it was the time when the most-used programming language in academia was Java. We had worked for many days helping out our students with their interaction design projects and had been drinking a lot of coffee. Being in Italy, we wanted to give our own project a coffee-related name such as mocaccino, marrocchino, or something along those lines. But everything was taken, because it was somehow Java related. Sitting in front of a computer with an open window to BerliOS, it just came to us: why not name the project after the coffee place in the village's main square, Arduino? It was also the name of the first Italian king, and it was not taken. The rest is history. — Massimo Banzi

What does Arduino look for in new business partners? — Margriet Debeij (The Netherlands)

In terms of supplier partners, the final user experience is the driving force behind development and therefore the choice of suppliers' components that make up the boards. That is, we select the vendors/partnerships based upon what is most appropriate to create the innovative tools we are producing for the customer. To improve our service to professional customers, we have recently launched a Systems Integrators program, where we carefully select whom we work with, requiring a strong commitment to customer service throughout the partnership. We look for partners who can bring to the table additional expertise, solutions and strategies that can work in synergy with ours, contributing to IoT innovation and growth for all parties involved. – David Cuartielles

In addition to his work at Arduino, David Cuartielles teaches at Malmö University in Sweden. How does he juggle both roles? Is there a synergy between the two? — Beatriz Sousa (The Netherlands)

David Cuartielles

As a matter of fact, my work at Malmö University is what inspired me to work with something like Arduino. It is not a matter of juggling roles, but more one activity which inspires the other. Education inspires board and content design, and vice versa. At a practical level, Arduino and Malmö University are very close, just 10 min. away by bike. I use Arduino as my personal office and run down to the School of Arts whenever I have a class or meeting. At the same time, I really love teaching, and I learn a lot from the students and alumni. I apply this knowledge to my work at Arduino. — David Cuartielles

The Arduino UNO and IDE filled a gap by providing a simple and affordable microcontroller development platform for hobbyists and makers and managed to become a market leader in this domain. The "Arduino Pro" product line tries to penetrate an existing industrial market occupied by many well-established competitors, products, and tools. How do you plan to make the difference? — Clemens Valens (France)

A fundamental aim of the Arduino Pro proposition is to transfer the productivity and creativity that makers and students have enjoyed with Arduino into the business world. We are actively helping companies transform their business models with IoT, providing robust and understandable hardware and SaaS platforms. The difference being that we will support the full project development (Hardware, Software, and Cloud) and the frictionless transition from prototyping to production. — Keith Jackson

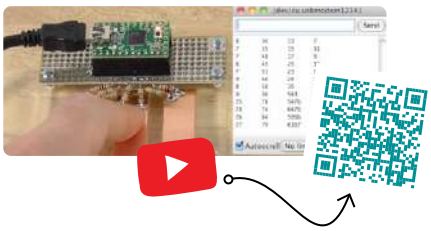
Massimo Banzi

What's on Massimo's personal electronics workbench? Is he working on any special projects? — C. J. Abate (United States)

On my workbench, there is a mess all the time. I'm mostly working on a couple of projects: building linear actuators with stepper motors to assemble different kind of "machines." I taught a class about it last July, and I'm figuring out what the next step is. The other big ongoing project is restoring old computers. I've been restoring some Apple II computers and other devices, such as French Minitel terminals. So, the desk is a mixture of stepper motors and old computer parts. — Massimo Banzi

Thousands of Arduino-based projects are posted online. Do David and Massimo have any favorites? Can they point to a few projects that really stand out? — Jens Nickel (Germany)

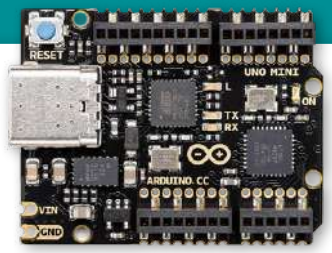
Watch the video "CapacitiveSensor Arduino Library!"



There are plenty of very complex projects that I could mention; however, I have a fondness for projects that help others develop their own ideas. There are some libraries that have lived very long within Arduino and that keep on helping others make great things. One example is the CapacitiveSensor library by Paul Bagder and Paul Stoffregen. This piece of code allows making touch controls for projects, saving you from having to use buttons. There are so many great projects based on that. On the other hand, if I had to choose a board-based project, I would fall hard for a bubble-blowing machine made by secondary school students in Catalunya. They used a servo motor to trigger the current on a fan that would blow the bubbles by mechanically shorting two cables because they lacked a relay. — David Cuartielles

The success of Arduino is due in large part to its users, who've shared tons of open-source libraries and applications for the UNO and similar boards. The "Arduino Pro" product line is targeted at industrial users who will probably be much more reluctant to share their code. Aren't you afraid that this will severely limit your chances at success? — Clemens Valens (France)

Arduino has always been open-source and always will be, as it's at the heart of Arduino. Open-source is not only fundamental to innovation and creativity that originates within the Arduino-lover community, but it also provides a key benefit to professional developers by enabling faster adoption of the technology and reducing the lock-in risks. It's only a limitation when looked at through the eyes of traditional businesses. There are thousands — if not millions — companies looking to transform their businesses. Many enterprises today use open-source software in their critical applications, starting from Linux OS and expanding into more application software such as databases, message brokers, etc. On the same line, there is a distinction to be made between Arduino cores and enabling technologies versus proprietary application code developed by our customers. We already provide tools to manage hybrid scenarios, combining open- and closed-source software with confidence. — Keith Jackson



Arduino owes much of its popularity to the Arduino UNO board. Are there any considerations for a Rev. 4, perhaps with a faster processor and Wi-Fi!? — Muhammed Söküt (Elektor, Germany)

This time last year, we launched a special miniature version of the UNO board, the UNO Mini LE, which was incredibly popular, selling out the short production run, proving the everlasting popularity of the UNO concept. A couple of months ago, we followed this up with the Make Your UNO Kit (or MY UNO in short), which is effectively three products in one, enabling beginners to learn to solder, solder their own UNO from scratch, and build an audio synth. We are always looking to build upon the success of our most loved products, so we are continually assessing how we can update them with faster processors, and important features requested by our customers. As such, we have a dedicated team working on this product. — David Cuartielles

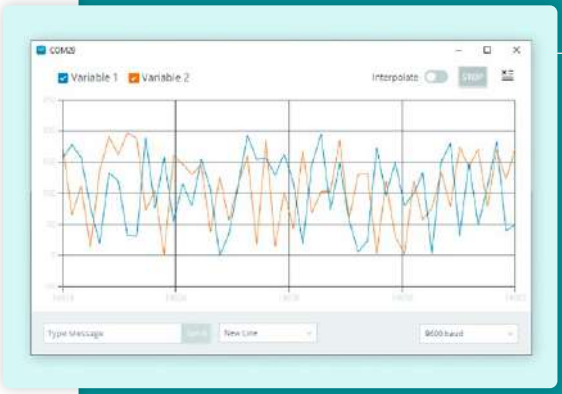
Thousands of companies around the world are using Arduino as an innovation platform. Tell us about one or two exceptional "pro" applications. — C. J. Abate (United States)

As you rightly say, thousands of companies are using Arduino around the world, so picking out just two is an interesting challenge.

Firstly, I'd have to say the Fluid Eye oil monitoring solution by Fluid Intelligence as this one was one of the use cases that first highlighted the potential of the professional market to us. Fluid Eye is an intelligent maintenance solution. It foresees fluid performance changes on heavy industrial machinery so early that end customers have time to make cost-effective maintenance decisions and prevent downtimes. The IoT solution has a front-end analysis and data transmission unit that is connected to the Fluid Cloud – using machine vision and machine learning algorithms to do the needed analysis and provide monthly and annual reporting. Different versions of Arduino MKR boards are used to read, operate and send the data from edge to cloud, with Arduino proving to be both easy to use and well-performing in these harsh environments.

Secondly, I'd go for the system developed by Arol with Arduino Pro. Arol were looking to add smart capabilities and data acquisition to their current industrial equipment used in high-speed capping machines and packaging systems during the bottling process, which handles 50,000 bottles and 150+ euro pallets per hour. The predictive maintenance system with the Nicla Sense ME monitors temperature and vibrations in specific critical areas of the machinery, providing both diagnostic and prognostic data to intercept errors or anomalies before they become faulty and develops an understanding of when the machines require maintenance. — Keith Jackson

Arduino IDE Serial Plotter is a very useful tool, especially in real-time data plotting applications. The problem with this tool is that it is not configurable (e.g., the X-axis is fixed at 500 points). Do you have any plans to make this tool configurable? Professor Dogan Ibrahim (United Kingdom)



Thanks for the feedback. This kind of insight from our users is always appreciated as it helps us to focus developments on what's meaningful to the widest user base. Currently, the Serial Plotter has been designed to have a fixed frame, but it's not 500 points; it's 5000 milliseconds (5 seconds) and whatever you can plot in that window will be plotted. We are assessing the possibility of adding a zoom in/out and area selection to zoom features for the future. — David Cuartielles

I have been using Arduino Cloud for over six months, and, as a maker, I enjoy its simplicity of use despite advanced features. As an educator, I also appreciate its potential as a tool for learning and teaching. What is your vision for Arduino Cloud? What can we expect to see in the next couple of years in the Arduino Cloud in terms of new capabilities, functions, support for non-Arduino hardware, and security? — Peter Dalmaris (Australia)

The Arduino Cloud was designed with the maker audience in mind, but, as it has been growing and evolving over the time, it has incorporated many features, such as RBAC or mass scale programming and deployment, that make it a good fit for industrial and professional environments, as well as at home and in the classroom. In the next few years, new capabilities, such as event processing and notifications, integration with third-party platforms and additional hardware platforms support, will change the face of Arduino Cloud while keeping the essence of ease of use of every Arduino product. — Keith Jackson

I am interested in what Arduino is doing for the younger generation. Tell us about the Arduino Education program. What is next on the agenda for Arduino Education? — Alina Neacsu (Germany)

Arduino Education is about preparing students for their future. Arduino is a facilitator in education, but the key message is to provide future career skills, future technologies and solving future problems. We are actively working on defining industry skill sets and bringing them into the learning process, to help students to build their competencies step by step with a clear goal.

Meanwhile, Arduino Education is moving forward with providing cutting-edge technologies: The Explore IoT kit is an example of how we can provide a unique IoT learning solution for different age groups, building and experiencing IoT projects. All the learning is embedded with strong, sustainable development goals in mind, because we would like to show the students through our tools, to understand the future challenges we are facing and how to make a better world through technology.

In order to facilitate our commitment, we are also expanding from hardware to digital services. Other than flexible, easily-to-use and affordable boards, carriers and components, we are working hard to bring our IoT cloud experience into the next level for education. With features such as classroom management and resources, together with advanced IoT and data dashboard functions, Arduino cloud has become an edifying tool for schools. Furthermore, we are improving the Science Journal app to improve the experience of experiments and scientific journaling, which has helped millions of students in science education. All the digital services form a strong foundation to our hardware development, and provide a unique and advanced experience between physical and digital learning.

In order to help schools' transformation into an innovative learning environment, Arduino Education started the Inspiration program. An Arduino Education Inspiration Lab is a dedicated space, whether that's in a school, university, business, or other institution, that provides innovative, exciting STEAM learning opportunities and certifications. In an Inspiration Lab, you can hold specialized STEAM courses for students, teacher training and professional development, and one-off workshops or projects. You could even open the lab to the public so that they, too, can learn about Arduino Education solutions, extend their STEAM knowledge – and have fun!

Even more than that, an Inspiration Lab promotes the UN's fourth sustainable development goal of inclusive and equitable education and lifelong learning opportunities for all, and supports career-based and professional skills development. We are dedicated to bring Arduino Education Inspiration Labs around the global and further carry our mission of teach today for tomorrow. — Yu Hu

220514-01

Getting **Started** with the **Portenta X8**

Manage Software Securely with Containers

Figure 1: The Arduino Portenta X8.

By Benjamin Dannegård (Arduino)

The Arduino Portenta X8 is a powerful, industrial-grade System-on-Module (SoM) running a Yocto Linux distribution from Foundries.io. Thanks to the Arduino Cloud integration, it is easy to create a FoundriesFactory, compose Docker containers and upload them securely to your board(s).

The groundbreaking Arduino Portenta X8 (**Figure 1**) is a powerful, industrial-grade System-on-Module (SoM) preloaded with Linux OS, making it a plug-and-play solution capable of running device-independent software, thanks to its modular container architecture. Onboard Wi-Fi and Bluetooth Low Energy (BLE) connectivity allow OS and application updates to be carried out remotely, always keeping the Linux kernel environment at top performance levels. The X8's combination of microprocessor and microcontroller offers unprecedented flexibility to run Linux apps and perform real-time tasks simultaneously and securely. Arduino's extensive available libraries, together with a container-based Linux distribution, enable IT professionals, system integrators and consulting firms to build and boost a wide variety of solutions for industrial contexts. It also lends itself to building automation and smart agriculture applications. Some applications could be:

- Connected edge computer for manufacturing
- Autonomous Guided Vehicles (AGV)
- Interactive full-HD secure kiosks and digital signage
- Office and home control systems
- Navigation and control for smart agriculture
- Behavioral analytics for offices and factories

Core Concepts

The Arduino Portenta X8 uses an embedded Linux environment, meaning that it needs a base distribution – a mechanism to update it and some applications that can run on the board. As the base, the X8 uses a Linux distribution built with the Yocto Project with applications that are installed and packaged as confined containers (**Figure 2**). The Yocto Project was created specifically for embedded systems to ensure that it never lacks the flexibility needed to accomplish target footprint sizes and functionality tweaks.

In collaboration with Foundries.io [1], we have developed an easy solution for connecting the Arduino Portenta X8 to their FoundriesFactory distribution. By connecting the Portenta X8 to a computer, we can immediately access the setup web page (**Figure 3**). This allows for a smooth and fast installation process. Here, the board's Wi-Fi can be set up, and then it can easily be linked to a FoundriesFactory of your choosing (see **Figure 4**). With the help of Foundries.io's integration with Arduino Cloud, you can easily create your FoundriesFactory right from the Arduino Cloud page.

The FoundriesFactory page allows you to add members, making it easy to keep track of the members of a team that need to have access to the Portenta X8s linked to the Factory. It is also possible to set up teams for better management. On the page, you can also find a list of all devices linked to the Factory, along with their names and the versions of the containers that are currently uploaded to the boards. There is also a page that lists

all the different container versions uploaded to the Factory. All of this allows for an easy overview and management of the people in the team, as well as the devices and targets. This already enables easy fleet management for one's devices, but the Foundries.io fleet management tool, Waves, allows for even more control. For a better understanding of how this tool works with the Portenta X8, have a look at the *FoundriesFactory Waves Fleet Management* tutorial for the X8 [2].

Foundries.io basically created their generic distribution based on Yocto with minimal software installed. By default, it implements top-level cybersecurity features such as OP-TEE, a system that allows for securing the OS, optionally supporting a secure element, and OSTREE, a system to checksum every single file and directory in the system. This makes their solution ideal for professional applications. A custom OTA system-update mechanism is based on a client running on a target and a robust cloud server. Foundries.io connected docker-compose as a way to deploy a software solution to a target. This is like having an app store for a particular device – with the difference that we're not installing an app but a container that may contain a whole distribution or a minimal distribution running only our app or our set of apps. In addition to that, they developed the cloud side as well. This means you can use FoundriesFactory, a cloud DevSecOps subscription service, to build, test, deploy, and maintain secure, updatable IoT and Edge products. This provides a unique ID and automatic builds of the base system and containers for this system in one place.

Containers

As mentioned briefly before, containers are an important part of the X8's functionality. A container is a standard unit of software that packages up code and all its dependencies, so the application runs quickly and reliably from one device to another. A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application. With the integrated use of Docker on the Portenta X8, it is easy to download already-existing containers or create your own highly-customizable container. Using a site such as Docker Hub [3],

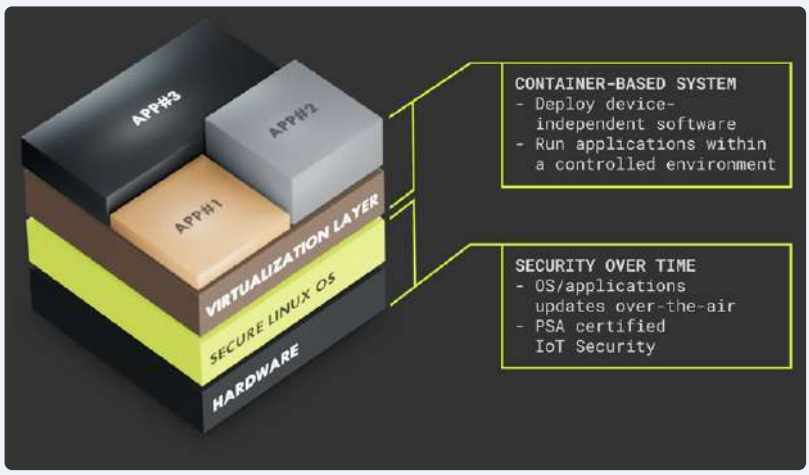


Figure 2: System infographic.

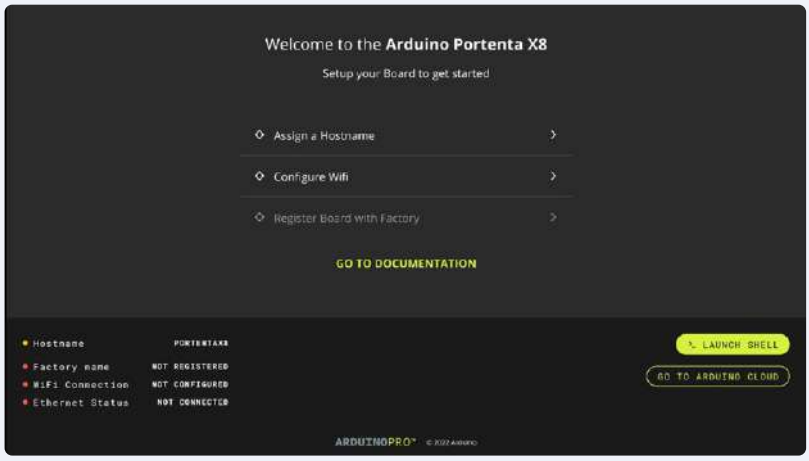


Figure 3: Portenta X8 setup web page.

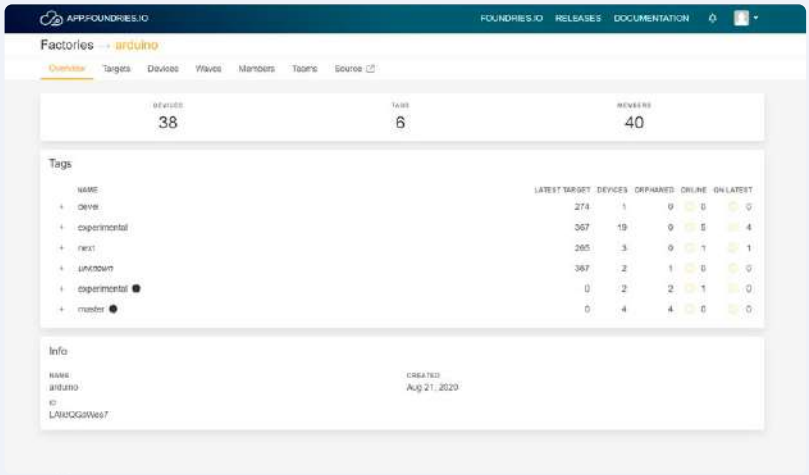


Figure 4: FoundriesFactory overview page. (Source: Foundries.io)

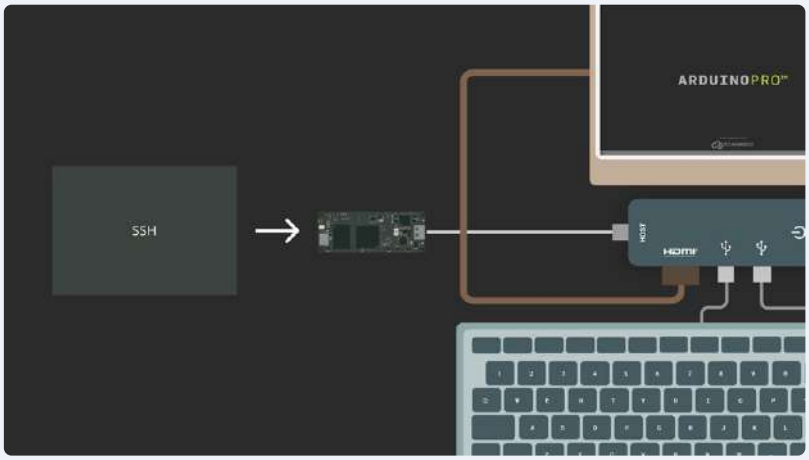


Figure 5: Portenta X8 connection illustration.

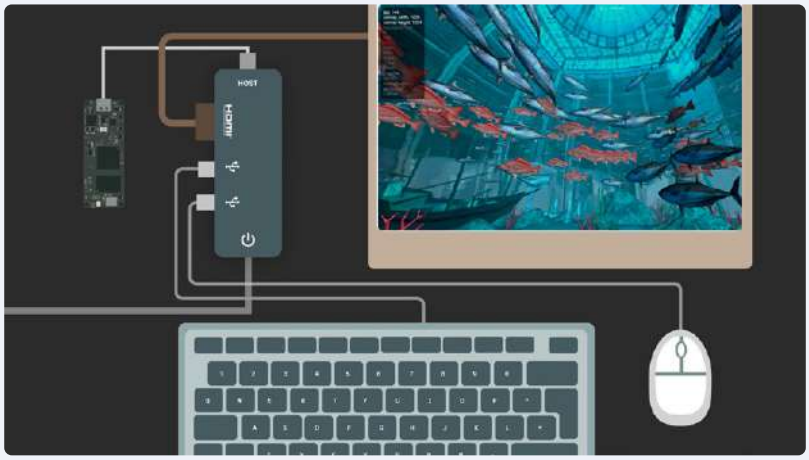


Figure 6: Demo running on Portenta X8.

which contains a plethora of ready-to-go containers, allows us to apply the Portenta X8 to work in many different fields. When creating a container from an image, that image must contain the container's entire file system. This means that it must contain everything needed to run an application, all dependencies, configurations, scripts, binaries, etc. The image also contains other configurations for the container, such as environment variables, a default command to run, and other metadata. However, a container is a single app, whereas you want to use multiple container applications, so, with the X8, we use docker-compose, which helps us manage these. A docker-compose app's file structure might look something like this:

- `docker-build.conf` contains the minimal 'unit test' command to be executed on the container to prove it's working.
- `docker-compose.yml` defines the app name through the Factory, its permissions and settings for the involved containers.
- `Dockerfile` is a file that builds a Docker image. The image contains all the dependencies the Python application requires.
- `requirements.txt` contains the Python dependencies that your app will copy and download.
- `src` directory will contain the source code of the app you want to run in the container or a startup script.

Thanks to Foundries.io this procedure is made easier, since there is a container build script running on the CI build server associated with the FoundriesFactory, which is triggered by a push to the Factory repository. This lets us create a new docker-compose application, add it to the repository and commit or push it. After a moment, the Factory's page will show that the container build script has been triggered and it is now building the new docker-compose app. All of this helps us strive towards an embedded Linux platform running a custom Yocto distribution that is secure and can survive severe cybersecurity scrutiny, and where the user can do whatever they have in mind by deploying applications and custom code using containers.

RPC Feature

The container infrastructure provided by Arduino allows us to run real-time processing on the Arduino side while running a full-fledged operating system on the Linux side. Thanks to the X8's architecture, the microcontroller can take care of certain peripheral handling and exchange the required data between the microcontroller and a Python application (it is also possible to use any other programming or scripting language), which runs on the Linux side. The communication mechanism that is being used to accomplish this is referred to as RPC (Remote Procedure Call). Using the docker infrastructure, it is possible to create a python application that uses the RPC mechanism to exchange data between the microcontroller and the iMX8, which runs the Linux operating system. If you are interested in delving deeper into this feature, take a look at the *Data Exchange Between Python on Linux and an Arduino Sketch* tutorial [4].

Portenta X8 in Action

Let's take a look at what the process of using the Portenta X8 with FoundriesFactory and containers looks like with a quick demo. The Arduino Portenta X8's NXP i.MX 8M Mini Processor can be used for 3D rendering. This will allow us to display 3D content on a screen or video output. The device uses OpenGL to process the 3D-related calculations. In this demo, we will render web content from the internet using WebGL and display it

on a screen by means of a USB hub that has an HDMI output. There are plenty of ways to communicate with your board, be it wireless or tethered, and then use ADB or SSH to communicate with the X8 via a command-line interface. As the Portenta X8 is a Linux device, you can use normal Linux commands to create files, change directories, etc.

After going through the process, the latest firmware for the X8 will be installed – this will make the rest of the process much easier. Now that the board is linked to a FoundriesFactory, we can easily download the necessary container to the Portenta X8 and run it with only a few lines using SSH or ADB. Using FoundriesFactory, we can easily upload a target containing this example to a linked device. If a local method is preferred, then it is just as simple to download the repository, push it to the device using the ssh or adb shell, and install and run it with a few commands. Once it is all downloaded and running, the Portenta X8 can be connected to a screen using a USB hub (**Figure 5**). The Aquarium 3D sample from WebGL will now play (**Figure 6**). A mouse can also be connected to the USB hub to interact with the demo on the screen. This allows for testing the X8's 3D rendering capabilities with WebGL, making it possible to test different parameters or access other relevant information that might be needed. Another option is to modify the container using a Linux text editor to change the URL to show something else on the screen.

If you would like to replicate this process or examine it in more detail, check out our tutorial, *Output WebGL Content on a Screen* [5]. On the Arduino docs site, you can also find more tutorials that show the possibilities of the X8 and its use of docker containers. If you would like to see how you can run a database container on the Portenta X8 that hosts a WordPress site, then head over to our *Running Wordpress and Database Containers on the Portenta X8* tutorial [6]. Maybe you are more interested in seeing how the Portenta X8 can provide a solution for MQTT data logging? Then, you should check out the *Data logging with MQTT, Node-RED, InfluxDB and Grafana* tutorial [7]. If you want to extend the capabilities of the X8 even more, plug it into an Arduino Portenta Max Carrier board. Then, for example, you could build a multi-protocol gateway. To do so, have a look at our *Build a Multi-Protocol Gateway with Portenta X8 & Max Carrier* tutorial [8].

Now that you have gotten an idea of what the Arduino Portenta X8 is capable of, what its core concepts are and what kinds of applications it can be used for, it is up to you to put it into action and get the most out of it.

220546-01

About the Author

Benjamin Dannegård is an Interaction Designer from Malmö, Sweden who has an interest in all things tech and nerdy. He has been with Arduino for two years as a maker and Pro content creator.

Related Products

Looking for the main items mentioned in this article? Arduino and Elektor have you covered!

- **Arduino Portenta X8**
  www.elektormagazine.com/arduino-portenta-x8
- **Arduino Portenta Max Carrier**
  www.elektormagazine.com/arduino-portenta-max-carrier
- **Arduino Portenta H7**
  www.elektormagazine.com/arduino-portenta-h7

WEB LINKS

[1] Foundries.io: https://foundries.io
[2] Tutorial: 'Using FoundriesFactory Waves Fleet Management' for the X8: https://docs.arduino.cc/tutorials/portenta-x8/waves-fleet-managment
[3] Docker Hub Container Image Library: https://hub.docker.com
[4] Tutorial: 'Data Exchange Between Python on Linux and an Arduino Sketch': https://docs.arduino.cc/tutorials/portenta-x8/python-arduino-data-exchange
[5] Tutorial: 'Output WebGL Content on a Screen': https://docs.arduino.cc/tutorials/portenta-x8/display-output-webgl
[6] Tutorial: 'Running Wordpress and Database Containers on the Portenta X8': https://docs.arduino.cc/tutorials/portenta-x8/wordpress-webserver
[7] Tutorial: 'Data Logging with MQTT, Node-RED, InfluxDB and Grafana': https://docs.arduino.cc/tutorials/portenta-x8/datalogging-iot
[8] Tutorial: 'Build Multi-Protocol Gateway with Portenta X8 and Max Carrier': https://docs.arduino.cc/tutorials/portenta-x8/multi-protocol-gateway

Build, Deploy, and Maintain Scalable, Secure Applications

With Arduino Portenta X8 Featuring NXP's i.MX 8M Mini Applications Processor and EdgeLock® SE050 Secure Element

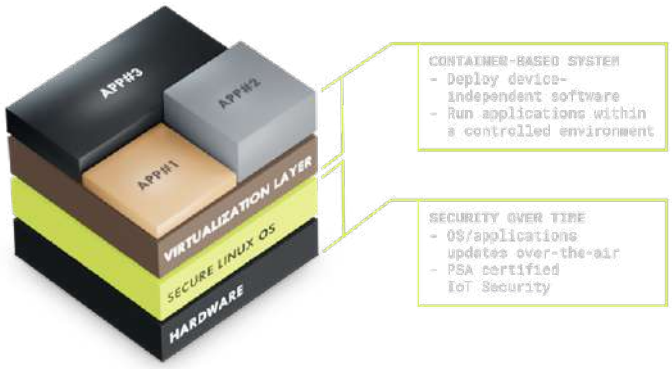
Contributed by NXP Semiconductors

Bringing an IoT device to the market involves significant design and development effort – with scalability issues, security challenges, and device limitations around every corner. Adding intelligence makes it even more complicated. This makes the selection of the right development hardware and software critical to getting secure edge products to market faster. This article introduces the Arduino Portenta X8 platform, an industrial-grade, secure SOM based on NXP's i.MX 8M Mini applications processor and an onboard EdgeLock® SE050 hardware secure element. This PSA-certified platform is also Arm® SystemReady IR for assured security.

Arduino Portenta X8 is a powerful, industrial-grade system on a module with Linux® OS preloaded onboard, capable of running device-independent software because of its modular container architecture. It offers two approaches: flexibility of usage of Linux combined with real-time applications through the Arduino environment. Onboard Wi-Fi/Bluetooth® Low Energy connectivity allows remote OS/application updates, always keeping the Linux kernel environment at top performance levels.

State-of-the-Art Security

The container-based system integrates different layers of security starting from the hardware layer which includes NXP's Secure Element. It utilizes the cloud-based DevOps platform from Foundries.io [1] to reinvent the way embedded Linux solutions are built, tested, deployed and maintained. The Portenta X8 includes the customizable open-source Linux microPlatform OS, built using best industry practices for end-to-end security, incremental OTA updates and fleet management.

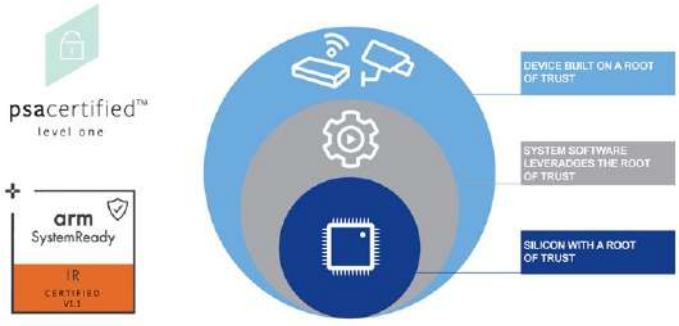


Portenta X8 Container and Security.

The virtualization layer allows users to deploy device-independent software running within a controlled environment. They can create their own containers using Docker and download premade images from Docker Hub or other public registries available to build a tailored application. If the developer wants to enter the embedded world, they can do so easily by building their application, running it on a container, putting it on the board and testing it out of the box. This provides a wide range of opportunities by mixing the Linux capabilities and the Arduino standard experience.

Portenta X8 achieved PSA Certification and the NXP EdgeLock SE050 hardware secure element provides key generation, accelerated crypto operations and secure storage. X8 also achieved Arm® SystemReady [2] certification and integrated Parsec services, making it one of the first Cassini Products or Cloud Native Edge devices available to developers in the market. It seamlessly runs Fedora IoT, Fedora Server, Debian and Linux microPlatform. Enabling the migration of cloud-na-

tive workloads from the Cloud to the edge, the Portenta X8 contributes to a cloud-native developer experience across Arm's diverse and secure IoT ecosystem.



Platform Security Architecture.

EdgeLock SE050 – A Trust Anchor for IoT

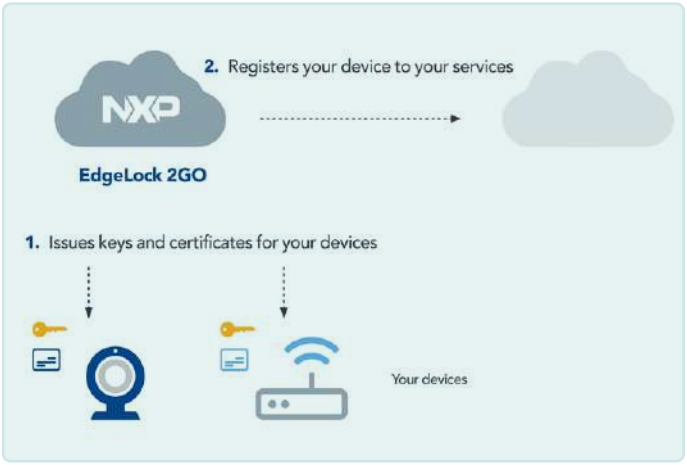
NXP's EdgeLock SE050 [3] is a discrete and tamper-resistant security hardware for protecting the identity of a device, including cryptographic keys and certificates. It's a standalone embedded secure element that is attached to the main processor over the I2C interface. The EdgeLock SE050 is certified Common Criteria EAL 6+ for the hardware and operating system. This ready-to-use secure element for IoT devices provides a root of trust at the IC level and delivers real end-to-end security – from edge to cloud – without the need to implement security code nor handle critical keys and credentials.

Silicon-based Root of Trust: EdgeLock® SE050 Secure Element.

Delivered as a ready-to-use solution, EdgeLock SE050 comes with multiple pre-implemented cryptographic algorithms and protocols and a complete product support package that simplifies design-in and reduces time to market. In addition to libraries for different MCUs and MPUs, the support package also offers integration with the many common OSs including Linux, RTOS and Android.

IoT device designers are facing two major challenges when implementing device onboarding to the cloud: provisioning of the device identity and managing device identities once released to the field. The provisioning of the device refers to the installation of keys and certificates. Managing device identities refers to the update, addition or revocation of keys and certificates throughout the device lifecycle.

To help designers solve these challenges, NXP provides the EdgeLock 2GO [4] managed service. The platform is a purpose-built hardware and service combination that establishes a silicon-based root of trust. EdgeLock 2GO issues the identities required for IoT devices and installs the credentials securely into the EdgeLock SE050 hardware. It also automatically registers the IoT device directly to the cloud service.



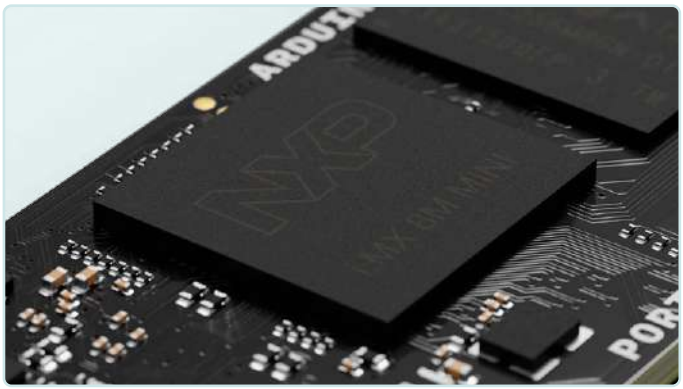
NXP Manages Device Credentials.

This flexible service supports multiple types of credentials and applies different configurations depending on the project. Credentials can be renewed or added to devices released in the field. With the commissioning of EdgeLock SE050 and EdgeLock 2GO, users get an end-to-end solution that is simple, secure and flexible.

As IoT continues to expand, so do the risks. NXP's EdgeLock combination, with its hardware-based security and service for credential management, gives device manufacturers a safer way to do business. With NXP EdgeLock supporting the deployment of a device, it reduces time-to-market and lowers the day-to-day costs of operating an IoT deployment while having the confidence of knowing devices are protected by high-level security.

Unleash the Power: Providing More Speed and Improved Efficiency

The i.MX 8M Mini [5] SoC is NXP's first embedded multicore applications processor built using advanced 14LPC FinFET process technology, providing more speed and improved power efficiency. The i.MX 8M Mini family of applications processors brings together high-performance computing, power efficiency, and embedded security needed to drive the fast-growing edge node computing, streaming multimedia, and machine learning applications.



The i.MX 8M Mini SoC is offered in single, dual and quadcore variants using Arm® Cortex®-A53 operating at up to 1.8 gigahertz per core. Delivered in advanced low-power process, the core complex is optimized for fanless operation, low thermal system cost and long battery life. The Cortex-A cores can be powered off while the Cortex-M4 subsystem performs low-power, real-time system monitoring. The DRAM controller supports 32-bit/16-bit LPDDR4, DDR4, and DDR3L memory, providing great system design flexibility.

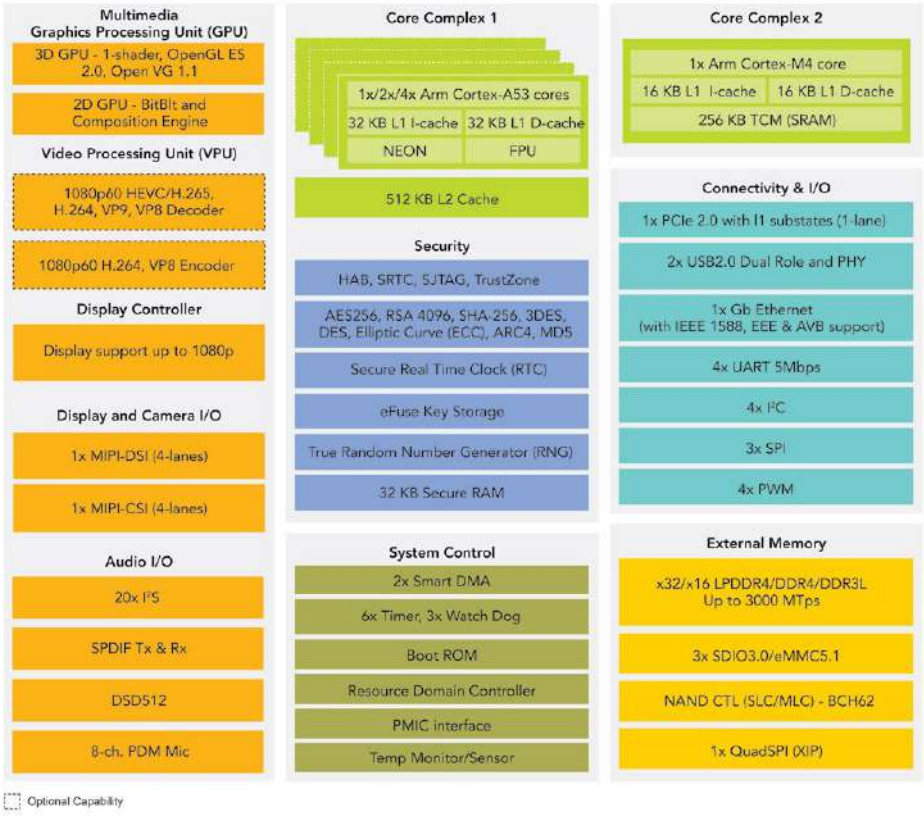
i.MX 8M Mini core options are optimized for ultra-low-power, even sub-Watt in specific applications, but offer the breadth of processing power necessary for consumer, audio, industrial, machine learning training and inferencing across a range of cloud providers. The i.MX 8M Mini SoC also packs-in hardware 1080p video acceleration to enable two-way video applications, 2D and 3D graphics to provide a rich visual HMI experience, and advanced audio capabilities to enable audio-rich applications. An extensive selection of high-speed interfaces enables broader system connectivity and targets industrial-level qualification.

Application Examples Include:

- **Industrial Automation**
  - The Portenta X8 can then act as a multi-protocol gateway, sending data to the Cloud or ERP system via Wi-Fi, LoRa, NB/IoT, LTE Cat.M1.
  - The availability of Linux containers like ROS within the Arduino environment makes the Portenta X8 a great fit for autonomous guided vehicles.
- **Building Automation**
  - Interacting with environmentally smart sensors, Portenta X8 allows the implementation of real-time ML and image processing on the edge.
  - Smart kiosks usually leverage several components (e.g. card readers, cameras, microphones), requiring a diverse selection of I/Os. When combined with a Max Carrier, the Portenta X8 ensures Wi-Fi connectivity and allows administrators to remotely monitor machine usage.
  - The Portenta X8 can simultaneously control HVAC systems, switch on/off smart appliances, autonomously adjust lighting and control accesses on the edge.

Start developing today with the industrial-grade, secure Portenta X8 SOM [6] with outstanding computational density.

220576-01



i.MX 8M Mini Applications Processor Block Diagram

WEB LINKS

[1] Foundries.io: https://foundries.io/
[2] Arm SystemReady: https://www.arm.com/architecture/system-architectures/systemready-certification-program
[3] EdgeLock SE050: https://bit.ly/EdgeLockSE050
[4] EdgeLock 2GO: https://bit.ly/EdgeLock2GO
[5] i.MX 8M Mini: https://bit.ly/iMX8MMini
[6] Portenta X8 SOM: https://www.arduino.cc/pro/hardware/product/portenta-x8

How I Automated My Home

Arduino CEO Fabio Violante Shares Solutions

Fabio Violante

By Keith Jackson (Arduino)

Many CEOs can talk the talk, but how many can walk the walk when it comes to actually using the products and services of the businesses they run? In Arduino CEO Fabio Violante's case, walking the walk is what he does — he takes work home with him, having developed multiple Arduino-based solutions for automating his family home.

A lot of heads of industry do their corporate jobs on weekdays, and then head home, where they won't see another product of theirs until they're back at the office. Not so in the case of Arduino's CEO, who "dogfoods" his own products at home.

Like many people during state-imposed pandemic lockdowns, Fabio began a whole host of DIY projects to improve things around the house. With all that time at home, he could finally put his hands and mind into creating the ideal Arduino Cloud-based solution for controlling the home.

Looking at Fabio's smartphone, you can quickly see what devices he's connected up around the home, as using the Arduino IoT Remote App enables Fabio to monitor, control, and automate these appliances from anywhere in the world – or at least from Milan to Rapallo, where Fabio has his home.

So, what drove Fabio to create these solutions, and how did he go about it? Like many things in life, it was those frustrating small annoyances inherent in the way appliances operate that inspired him to give it a go.

Resource: Arduino Home Automation Ideas

It can be daunting building a home automation system from scratch. But it really doesn't need to be. It will be a lot more fun than you expect, and home automation can become incredibly addictive – just ask Fabio! Here are a bunch of ideas that can easily and effectively make a difference to your home and lifestyle, including more info on the detailed projects . Enjoy! https://arduino.to/hub

Intelligent Fan Coil (HVAC)

Problem:
Recessed fan coil was not able to push (hot/cold) air far enough.

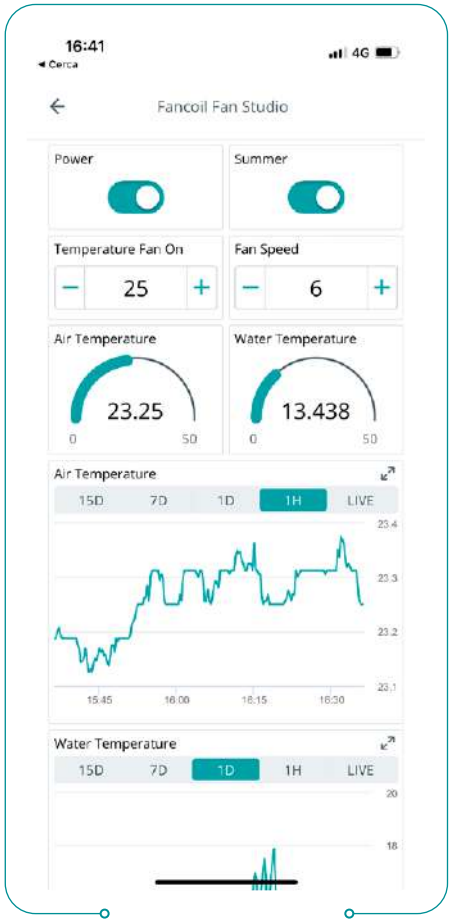
Solution:
Add two supplementary silent, brushless fans for ventilation, but only activate them when the coolant fluid is at the correct temperature (different for summer and winter) and only when needed, based on the differential air temperature.

Hardware:

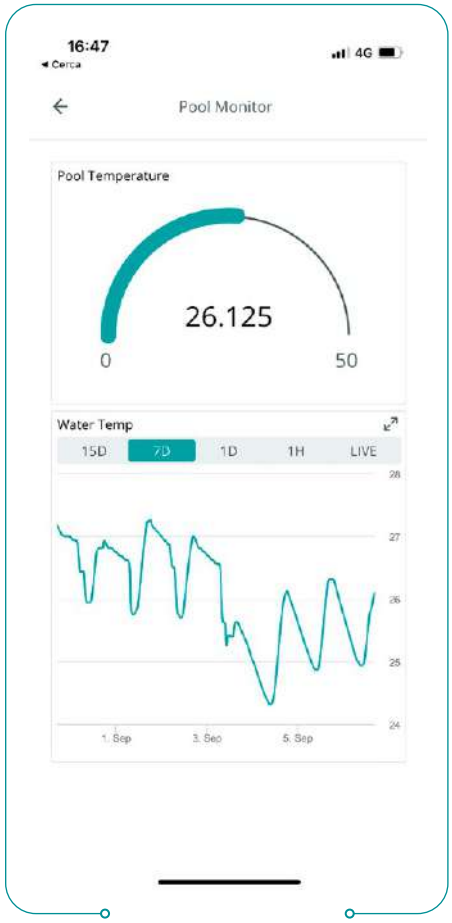
- 2 × fans
- Arduino Nano 33 IoT
- Screw terminals
- 12 V power supply
- 2 × 1-wire temperature sensors

Software:

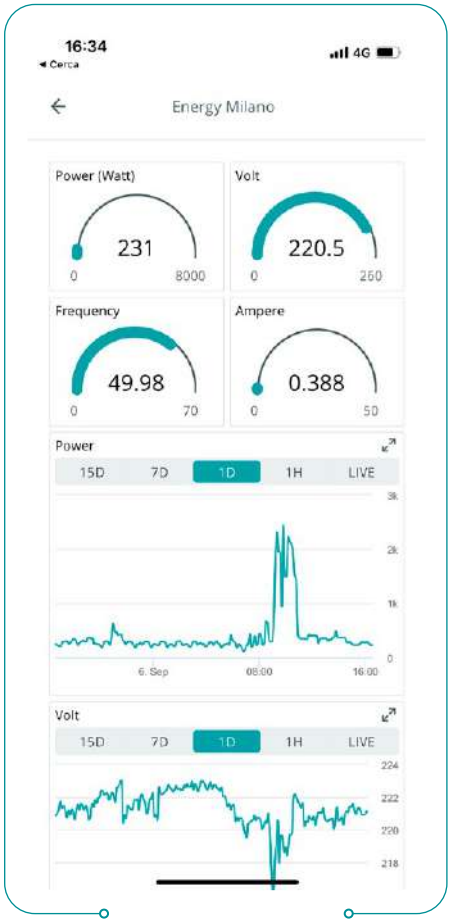
- Arduino Cloud with mobile dashboard, Alexa integration for manual operations or settings



Fan control



Pool monitor



Energy monitoring

Swimming Pool Temperature Monitor

Problem:
The family (especially the children) need to know if the outdoor pool is warm enough to swim in.

Solution:
Measure the pool temperature digitally using a temperature sensor connected to Arduino Cloud and distribute the information to the dashboard, to Alexa, and to the app on the family's phones.

Hardware:
- Arduino Nano 33 IoT
- screw terminal
- 1-wire temperature sensor
- 220 VAC-to-12 VAC isolated transformer
- 12 VAC DC rectifier (required by pool regulations)

Software:
- Arduino Cloud with mobile dashboard, Alexa integration

Living Room Environmental Monitor

Problem:
Measure the environmental parameters of the living room. We need to keep the humidity in the living room at the right level for the beloved piano.

Solution:
Arduino-based all-in-one device that abides by my wife's rule of anything to be displayed in the living room: does not look like a bomb.

Hardware:
- Arduino Oplà IoT Kit

Software:
- Arduino Cloud with mobile dashboard, Alexa integration

Outdoor Plant-Watering

Problem:
Measure the soil moisture of multiple outdoor plants and increase the amount of scheduled watering based on their needs. Create a single point of control to keep the garden plants healthy and watered when we're not around.

Solution:
Distributed Arduino solution for sensing, and centralized water solenoid control, working alongside the existing home automation system.

Hardware:
- Arduino MKR WiFi 1010
- 1-wire temp sensor
- 2 × generic soil moisture sensors
- Arduino Nano 33 IoT
- screw terminal adapter
- generic 8-relay board

Software:
- Arduino Cloud with mobile dashboard, Alexa integration, shared variables

Energy Meter

Problem:
Monitor and record energy consumption around the home.

Solution:
Use two energy meters (one for the day living area, one for nighttime) to gain a better understanding of the energy consumption around the home, to work out how to optimize it and save money in the future. Use Arduino and a professional metering solution.

Hardware:
- Arduino MKR WiFi 1010
- Arduino MKR RS485 shield
- 2-unit DIN rail enclosure
- 5 V DIN rail power supply
- Finder Energy Meter (1-phase 230 V 40 A, W17, 5 mm, DIN rail, RS-485 Modbus, NFC, IR, MID certificate) [model 7M.24.8.230.0210]

Software:
- Arduino Cloud with mobile dashboard, Alexa integration

Comprehensive Irrigation Solution

Having successfully automated his home, Fabio has now started on a megaproject (not to be confused with the Arduino Mega!) to manage the irrigation at his father's orange farm.

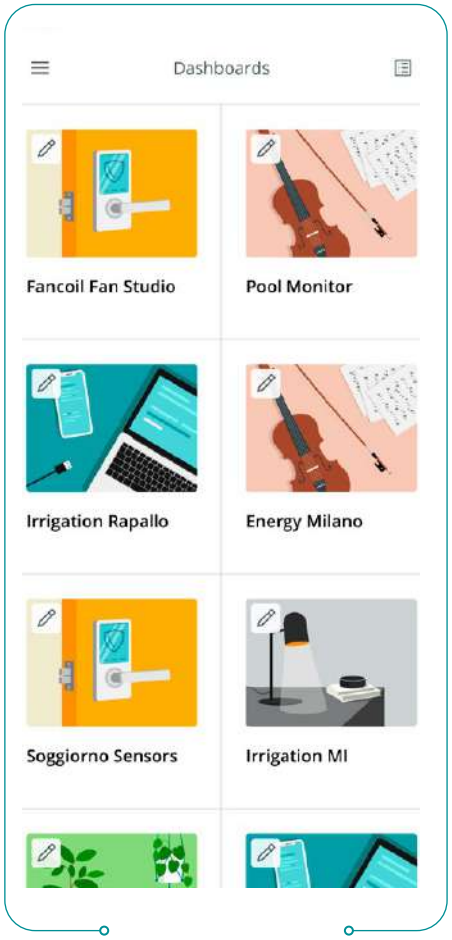
Problem:
Monitor moisture and manage the watering of several hectares of orange trees, using professional 380 V well pumps, sprinklers, distributed valves, and multiple soil types. Consider connectivity options, given the remote location.

Solution:
Arduino Pro integrated with the existing motor control system and manual fallback. Include remote connectivity via LoRa with cellular as a connectivity fallback, Wify energy metering.

Preliminary Hardware:
- Portenta Machine Control
- Portenta Max Carrier and Portenta H7
- 6 × Arduino Edge Control
- 5 × 5 W solar panels and 10 Ah lead-acid batteries
- 15s Watermark soil moisture sensors
- N × 1" Latching valves

Software:
- Arduino Cloud with mobile dashboard, Alexa integration

220548-01



Check out what is on Fabio's smartphone.

Questions or Comments?

Do you have any questions about these solutions or ideas of your own? Please reach out to Elektor at editor@elektor.com.

Arduino CEO Fabio Violante

Prior to Arduino, Fabio served as CTO of BMC Software, following BMC's acquisition by Neptuny, a startup he had co-founded. Fabio is also co-founder and board member of Moviri group. He holds an MSc and a PhD in Computer Engineering from Politecnico di Milano (Italy), and he was an adjunct professor at the same university. Outside of electronics, he has a passion for music (especially the piano) and fine wine.

Related Products

Looking for the main items mentioned in this article? Arduino and Elektor have you covered!

- **Arduino Nano 33 IoT with Headers (19937)**
  www.elektor.com/arduino-nano-33-iot-with-headers
- **Arduino Oplà IoT Starter Kit (19942)**
  www.elektor.com/arduino-opla-iot-starter-kit
- **Arduino MKR WiFi 1010 (19935)**
  www.elektor.com/arduino-mkr-wifi-1010
- **Arduino Portenta H7**
  www.elektormagazine.com/arduino-portenta-h7

ALTAIR 8800 SIMULATOR

Hardware Simulation of a Vintage Computer

Figure 1: My Altair 8800 simulator.

By David Hansel (United States)

Ever wanted to play with the computer that sparked the PC age but don't have the cash to buy a true vintage machine? You can build your own!

What the heck is an Altair 8800? If you don't remember (because you're not that old), having a look what Wikipedia tells us: "The Altair 8800 is a microcomputer designed in 1974 by MITS and based on the Intel 8080 CPU ... The Altair is widely recognized as the spark that ignited the microcomputer revolution as the first commercially-successful personal computer." [1] So, historically, it is a truly important computer system.

If you would like to build a system that looks and behaves like the now-48-year-old Altair 8800 but don't like the idea of searching for all the different parts, you can even buy an Altair-Duino kit [2], which comes with all components, a PCB, a case and a pre-programmed Arduino Due. In the Arduino Project Hub article [3] you can find links to nicely-made builds from other fans.

The Backstory

For a long time, I thought it would be cool to have an Altair 8800 computer to play around with. But, working old Altair systems are rare and therefore expensive, easily costing $2000 to $3000 if they are available at all. There are other options, such as at altairclone.com, but even this one still costs $600, which is too much for me to spend on a computer that — while very cool — will be of limited use. Thankfully, Mike Douglas, the creator of this Altair 8800 Clone, has made all the old documents and software he hunted down and used when creating the clone available to the community. Thanks to Mike's work, there's a wealth of information about the inner workings of the Altair and its most popular peripherals, readily available.

At one point, I looked at the Arduino Mega 2560's specs and wondered whether it would have enough I/O pins to simply connect the Altair's front-panel LEDs and switches to it and write my own emulator software. It turned out that the Arduino Mega has exactly the right number of I/O pins. So, I just had to create my own Altair simulator.

Using the Arduino Mega as the basis of the simulator worked well and was easy to set up, but the emulation only runs at about 25% of the speed of the Altair and can only provide 6 KB of emulated memory (although that would have been a lot in the 1970s). Permanent memory capacity (for storing programs/data created in the simulator) is also limited, as the Mega's EEPROM only holds 4 KB.

The Arduino Due has enough memory to support a full 64 KB of emulated RAM and is much faster than the Mega. In addition, the Due can store data in flash memory at runtime. This makes it possible to use any part of the 512 KB flash memory not used by the simulator itself as permanent memory. With the Due, I was able to create an Altair 8800 simulator that runs at about the original speed, provides 64 KB of emulated RAM, includes a lot of Altair software, and can still provide 32 KB of semi-permanent memory for loading and storing programs and data in the emulator.

Goals, Examples, and More

It was important to me to get as close as possible to the "real" Altair 8800 feeling when working with the simulator. This includes that the lights on the front panel reflect the original behavior as much as possible. One criterion for this was that it should be possible to play the "kill-the-bit" game on the front panel. **Figure 1** and the YouTube video "Arduino Altair 8800 Simulator - Entering and Playing Kill-the-Bit" [4] demonstrate that I succeeded.

It turned out that the simulation is so faithful to the original that even the original Altair 8800 music demo [5] works – for comparison, here's my version [6]. This kind of music production required an AM radio to pick up the electromagnetic interference generated by the Altair's circuits.

In 1977, Processor Technology released a small expansion board (just a few capacitors and resistors) with accompanying software that turned the Altair into a respectable (for the time) music system. The same additions can be made to the simulator (see documentation) so that it plays the tunes created for the music system at the time. This YouTube video [7] shows my Clone playing *the William Tell Overture*.

Another historically important expansion for the Altair was the Cromemco Dazzler graphics card. With the help of a software or hardware extension, the simulator can also emulate this card [8], as shown in **Figure 2**. Another software/hardware extension [9] allows emulation of the VDM-1 graphics card (**Figure 3**) from Processor Technology.

Because I don't own an original Altair, I had to get all the necessary information from documents and videos (see acknowledgements in [3]). There may be a few minor differences, but all in all I think it reproduces the original behavior quite well (see the **Highlights** frame). One well-known (and intentional) difference is the HLDA status light: In the original, it signals that the CPU has confirmed that it has been stopped by an external device. This feature is never used in the simulator, so in there it signals that a file (serial/tape record/playback) is currently open.

When using the Arduino Due, please note that all data captured or stored in the simulator will be erased when you load a new version of the sketch into the Due. This is because the saved data is stored in flash memory, which is erased when a new sketch is uploaded (the Due does not have an EEPROM for permanent storage). When an SD card is connected to the Due, the saved data is stored on the SD card. In this case, the data will be preserved when a new sketch is uploaded.

The original Altair documentation (easily found via Google) contains all the information needed to operate the front panel switches. However, the simulator contains additional functions and integrated software. This is explained in the Documentation.pdf [10] file from the source code repository.

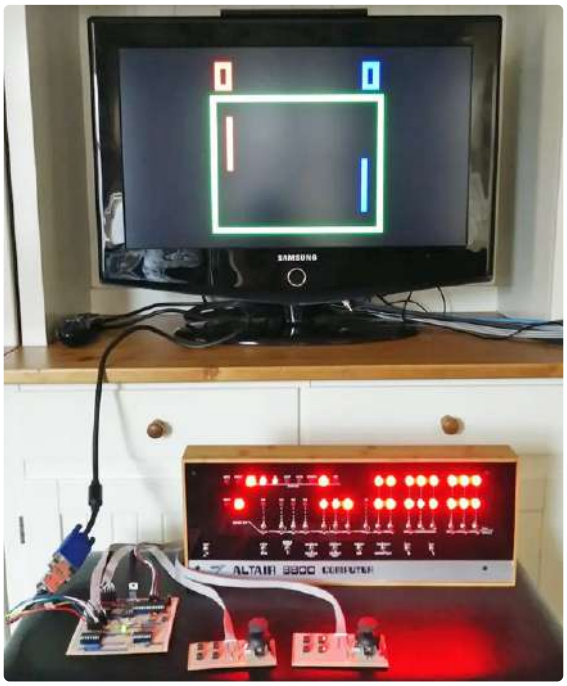


Figure 2: Cromemco Dazzler graphics card emulation in action.

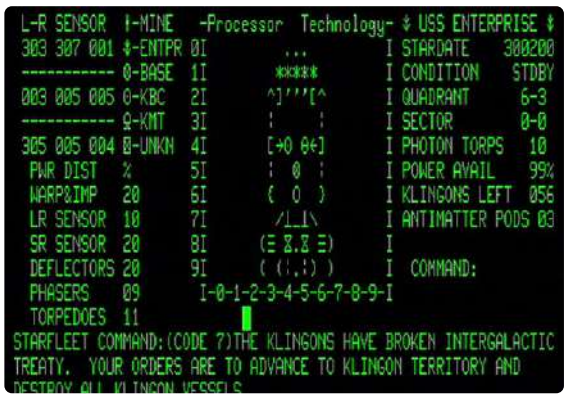


Figure 3: Screen of the VDM-1 graphics card.

Build Instructions

My goal for the project was to use as few additional components as possible. Both the Arduino Mega and the Due have enough I/O pins to connect all the front-panel elements directly. There are only additional transistors and resistors needed to drive the 36 LEDs (if they were connected directly to the Arduino's output pins and too many were turned on at once, the total current would exceed the Arduino's limits).

A complete schematic would be redundant (36 identical LED driver circuits, wiring for 32 switches) and thus not very helpful. Consequently, the circuit consists of sub-circuits (such as LED drivers, **Figure 4**), and tables (see [3]) indicate which elements are connected to which Arduino pins. A Fritzing file [3] shows the layout of the LED driver components on stripboard (**Figure 5**).

To create the front panel, I started with a high-quality scan of the Altair front panel [11] and had it printed on cardboard at a copy store. For the back (to hold the switches and LEDs in place), I used a sheet of 22-gauge metal and drilled the holes for the LEDs. The LED driver circuits are soldered to strip boards that connect directly to the LEDs, which in turn are held in place by the metal plate. The front panel is held upright by a simple wooden box. The box is not as deep as the original Altair (because it only needs to hold the front panel and the Arduino). **Figure 6** shows what it looks like inside.

To wire the front panel ON/OFF switch, I simply attached a power socket (the same as on the Arduino itself) to the box, connected it to the front panel switch, and from there to a power plug that connects to the Arduino.

When using the Arduino Due, emulation of up to 16 of the 88-DCDD disk drives can be enabled by connecting an SD card to the Due's SPI port. The last page of the schematic document [3] shows the required wiring in detail.

Before uploading the sketch to the Arduino Due, make sure that the optimization setting of the Arduino compiler is set to "Performance." By default, it is set to "size" (no idea why, since the Due has 512 KB of flash memory). To do this, load the file into a text editor and change any occurrence of -Os to -O3.

```
c:\Users\[user]\AppData\Local\Arduino15\packages\arduino\
hardware\sam\1.6.9\platform.txt
```

You can skip this step, but then the simulator will run significantly slower.

The simulator software can also run on a pure Arduino (Mega or Due) without any front-panel controls connected. This allows it to run some of the included programs (the ones that mainly use the serial terminal and not the front panel elements). To do this, edit the source `config.h` file and set `#define STANDALONE 1` (instead of `0`). See the "Debugging Capabilities" section of the documentation [3] to learn how to operate the virtual front-panel elements in this configuration. Note that this is not the intended use of the simulator. PC-based software emulators are more intuitive if you don't want to build the front panel hardware yourself. The `config.h` source file contains several switches that enable or disable simulator functionalities. The default settings work fine, but here you can optimize your simulator.

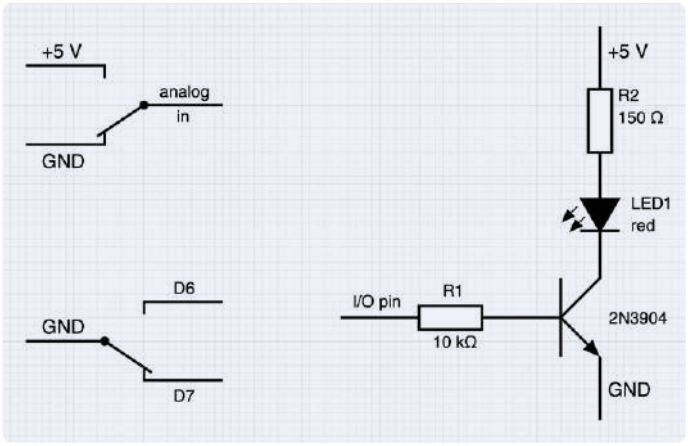


Figure 4: Sub-circuits for switches and LEDs.

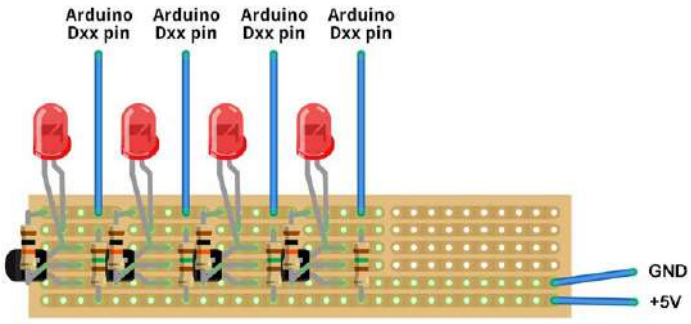


Figure 5: Parts on stripboard (Fritzing).

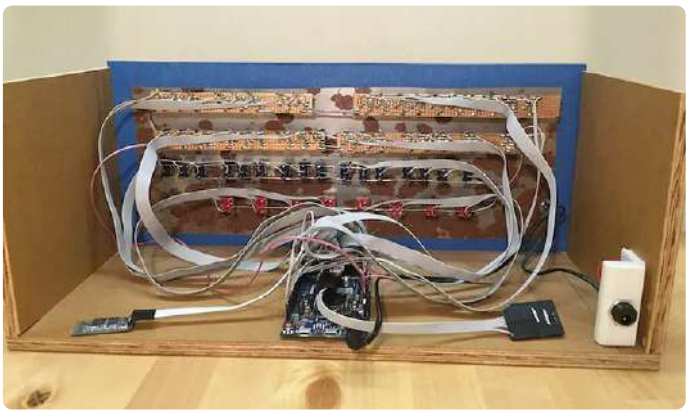


Figure 6: Inside view of the Arduino Altair clone.

Highlights

- Exactly reproduces the behavior of the Altair's front-panel elements.
- About as fast as the original Altair 8800 using an Arduino Due (25 % using an Arduino Mega).
- Size of emulated RAM = 64 KB (Due) or 6 KB (Mega).
- Some Altair programs are included and can be easily loaded onto the emulator: Pong, Altair 4K BASIC (the first Microsoft product), Altair Extended BASIC, MITS Programming System II (Due only), Altair Timesharing BASIC (allows multiple users to use BASIC simultaneously).
- BASIC and assembly example programs are included and can be easily loaded into BASIC/Assembler.
- Emulates an 88-SIO, 88-2SIO, and 88-ACR (audio cassette recorder interface) board. Any simulated serial device can be mapped to Arduino serial ports. By default, the two most common (88-SIO and 88-2SIO port 1) are mapped to the Arduino's main 115,200 baud serial port (8N1), which can be accessed via the USB cable. A serial-to-Bluetooth dongle connected to the RX/TX serial pins is recommended for other devices to serve as a terminal via Bluetooth.
- On the Arduino Due, both the main serial port (USB) and the Serial1 port (pins 18/19) can be used simultaneously.
- Data sent to each serial device (including the ACR tape) can be recorded to and played back from up to 256 files, which are stored in the Arduino's local memory (EEPROM or FLASH).
- The cassette interface supports the use of the CSAVE/CLOAD commands in Extended BASIC (support is automatic with no user interaction required). Ideal for developing your own BASIC programs!
- Emulates a Cromemco Dazzler graphics card (requires additional hardware/software [9]).
- Emulates a Processor Technology VDM1 video terminal card (requires additional hardware/software [10]).
- Emulates up to 16 88-DCDD disk drives (4 in default configuration). Drive emulation is optional, but requires the connection of an SD card to the Arduino's SPI header. (Due only.)
- Emulates an 88-HDSK hard disk controller with up to 4 hard disks attached (1 in the default configuration) and 4 platters per unit.
- Emulates an 88-RTC VI board with real-time clock and vector-interrupt processing. Enables the use of Altair Timesharing BASIC.
- 256-byte memory pages can be saved to permanent memory and reloaded into RAM. This is an easy way to save programs entered using the front-panel switches.
- Many settings can be easily changed via the integrated configuration editor.

Let's Share!

If you find this project interesting and want more information, then visit the Google Group, created by Chris Davis, for discussion of Altair-Duino-related questions [12]. Feel free to contribute to the discussion!

If anybody else wants to share their creation, let me know, and I'll post it on the Simulator website.

220406-01

About the Author

David Hansel is a maker and developer. He discovered Arduino in 2012 and has been creating projects with it (and other microcontrollers) ever since. He gathers projects that he thinks could be useful to others on his GitHub profile: https://github.com/dhansel.

Questions or Comments?

If you have technical questions, feel free to e-mail the author at david@hansels.net or the Elektor editorial team at editor@elektor.com.

WEB LINKS

[1] Wikipedia about the Altair 8800: https://en.wikipedia.org/wiki/Altair_8800
[2] Altair-Duino kit: https://altairduino.com/
[3] Arduino Altair 8800 Simulator on Arduino project hub website: https://tinyurl.com/yxkta8nz
[4] Arduino Altair 8800 Simulator entering and playing kill-the-bit: https://youtu.be/prdvkMP3FAA
[5] Altair 8800 music demo: https://youtu.be/1FDigtF0dRQ
[6] Music demo with my simulator: https://youtu.be/q45ENdbz8EU
[7] Altair Simulator playing the William Tell Overture: https://youtu.be/nqy8v41q5as
[8] Dazzler Display for Altair Simulator: https://tinyurl.com/y8f2wzsr
[9] VDM-1 Simulator: https://github.com/dhansel/VDM1
[10] Documentation, PDF: https://tinyurl.com/yvxtxhbn
[11] Altair front panel: https://vintage-computer.com/images/altairfrontpanelscan.jpg
[12] Google Group about Altairduino.com: https://groups.google.com/g/altair-duino

Related Products

Looking for the main items mentioned in this article? Arduino and Elektor have you covered!

- **Arduino Due with Headers**
  www.elektormagazine.com/arduino-due

MS-DOS on the Portenta H7

Run Old-School Software on Contemporary Hardware

By Clemens Valens (Elektor)

The Arduino Portenta boards provide the computing power required to emulate other computer platforms and virtualize hardware. This article explains how to install and run MS-DOS on the Portenta H7.

Many machines and production lines in factories all over the world that are spitting out the parts for the products we use today were built and installed several decades ago. At the time they ran modern software on state-of-the-art computer platforms. Today, however, many of these have become obsolete. As long as a machine keeps on working, modernizing it can wait until tomorrow but what to do when a board fails and spare parts are hard to find or, even worse, no longer available?

Virtualize It!

This is where the Arduino Portenta boards come in. As they pack a generous amount of computing power in a tiny module, they can be used to virtualize another computer system. Virtualizing means running a program that emulates another computer platform in such a way that the emulated platform can run its software as if it was the real thing. From the outside there is no difference.

A virtual machine can run on any computer, solving the spare-parts problem, and it executes existing programs, limiting custom software development costs. Another advantage is the possibility of adding modern features like USB and HDMI. This allows for instance replacing clunky and power-hungry CRT monitors by modern flat displays and hard-to-find serial keyboards by cheap USB models.

MS-DOS

One popular operating system from the past is Microsoft's MS-DOS, intended for x86-compatible computers. Even though development stopped somewhere in 2000, it is still widely used today. For this reason, the Arduino team has decided to implement a virtual x86 machine for the Portenta H7 that can execute MS-DOS. With it, the Portenta H7 can replace an old computer on a machine so it can keep on producing stuff for another 40 years.

The x86 emulator for the Portenta H7 is still a work in progress, but already you can give it a try. Here is how.

Prerequisites

Note that for the MS-DOS emulator to work a USB-C hub with HDMI port and (laptop) charging capabilities is required (i.e., it must have a power input). The Portenta outputs the video over its USB-C port, and it is powered from the same port. Therefore, a hub capable of providing power to the Portenta is practical. This option is often advertised as it having charging capabilities. Not all USB-C hubs have this, so choose carefully (**Figure 1**).

Figure 1: Use a USB-C hub with laptop charging capabilities.

▶

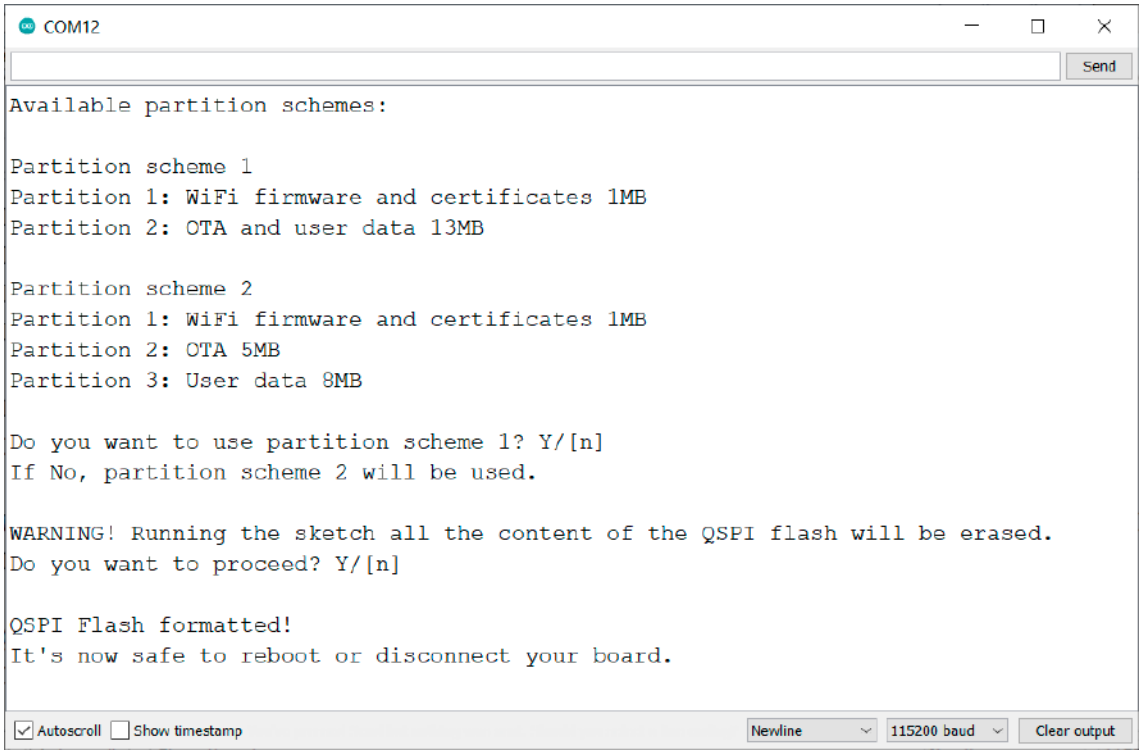


Figure 2: Format the QSPI flash memory with partition scheme 1.

Of course, it is also possible to power the Portenta H7 from a separate power supply instead but then it must power the hub and everything that is connected to it too.

Also, to play with MS-DOS on the Portenta you need a USB keyboard. This must be a standard Low Speed type (USB 1.0), not Full Speed or faster. This requirement is probably fulfilled most easily by the oldest and cheapest USB keyboard you can find. I tried to be clever by converting an old PS/2 keyboard into a USB keyboard with an Arduino Leonardo, but this doesn't work as the Leonardo turns it into a Full Speed USB keyboard.

Step by Step

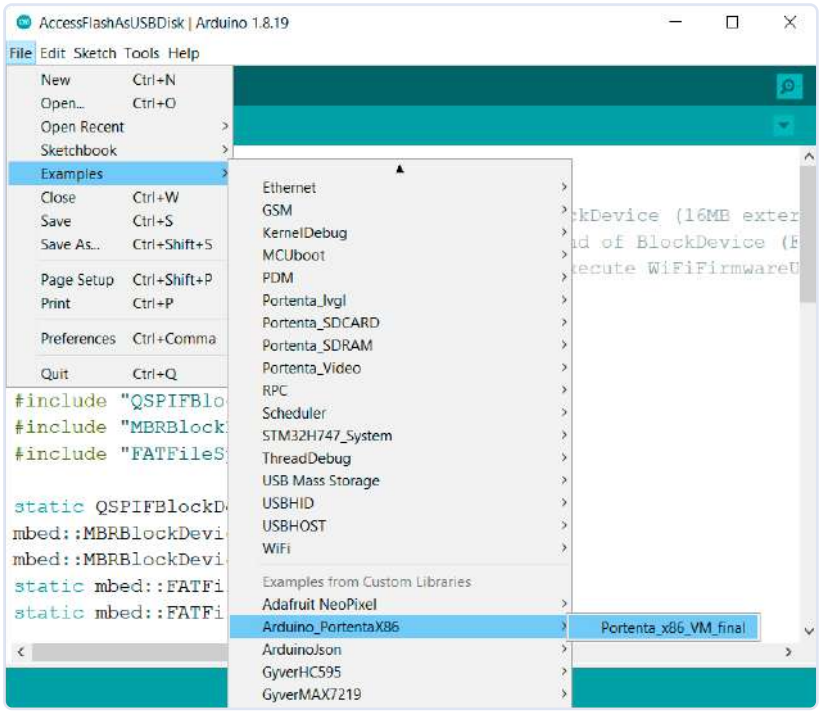
1. Open the Arduino IDE version 2.x or 1.8.x. I used 1.8.19.
2. Using the Boards Manager install the Portenta Boards Package *Arduino Mbed OS Portenta Board* (v3.3.0 at the time of writing).
3. As board, select the *Arduino Portenta H7 (M7 core)*.
4. Select the correct port.
5. Using the Library Manager install the library *Arduino_PortentaX86*. If you can't find it there, download it from [1] and install it as a library in the *libraries* folder of your *sketchbook* folder.

Now you must run a few sketches to prepare the Portenta's flash memory:

6. Run *Examples → STM32H747_System → QSPIFormat*' and make sure to use '*Partition scheme 1*' (**Figure 2**).
7. Run '*Examples → USB Mass Storage → AccessFlashAsUSBDisk*'.

Step 7 turns the Portenta in a USB flash drive on which you must copy the MS-DOS disk image that is included in the Arduino_PortentaX86 library. This library, based on the Faux86 MS-DOS emulator, is in the '*libraries*' folder of your '*sketchbook*' folder. You can find the location of the latter from the IDE's preferences (yes, it is a bit involved).

Figure 3: Upload the virtual machine (VM) sketch `Portenta_x86_VM_final`.



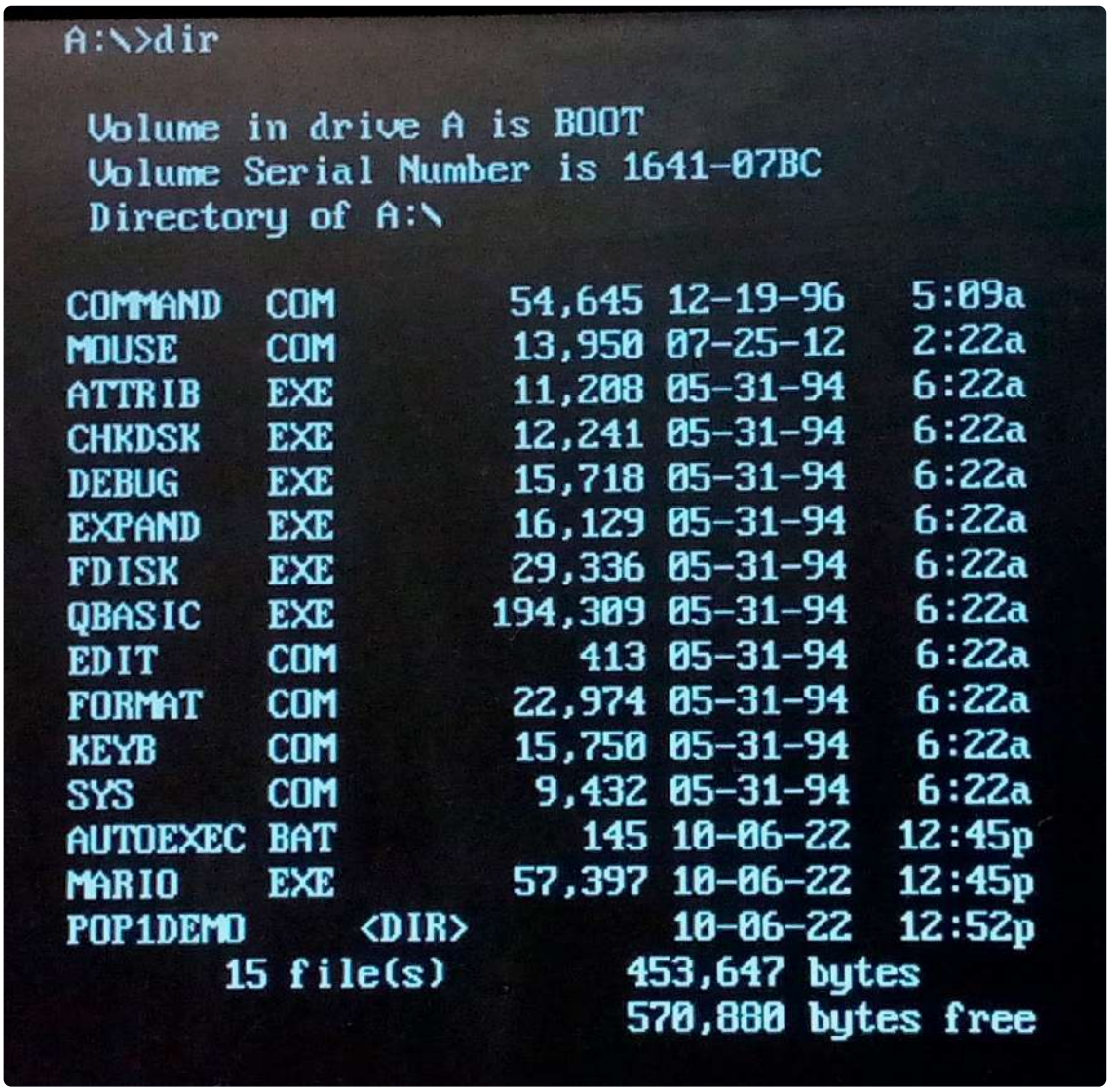


Figure 4: The contents of the A: drive at the MS-DOS prompt.

Copy the DOS image into the largest partition. Check the disk properties for the partition's size in case you are unsure.

8. Run '*Examples → Arduino_PortentaX86 → Portenta_x86_VM_final* (**Figure 3**).

The x86 virtual machine is now ready and loaded with MS-DOS. Make sure to connect the USB keyboard and monitor to the USB-C hub before connecting the Portenta to it as hot-plugging is not supported (even though I found it to work most of the time).

Figure 5: The MS-DOS disk image includes a playable time-limited version of the classic nineties game Prince of Persia.

9. Switch on the USB-C hub.

If all is well, you should now see the emulated BIOS execute the memory test and when done the MS-DOS prompt should appear (**Figure 4**):

```
A:\>
```

At the time of writing this article there wasn't a whole lot you could do with the emulator, but MS-DOS's Edit worked (with mouse support), and I could also write and run a simple program in QBasic. A playable time-limited demo version of the game Prince of Persia from 1990 is included in the disk image (in the folder POP1DEMO), showing off colorful VGA graphics (**Figure 5**).

Notes

The Portenta H7 has two cores, an ARM Cortex-M4 and an ARM Cortex-M7. It is the M7 that emulates the x86 processor while the M4 core takes care of all the USB stuff. If the monitor shows a yellow square instead of an MS-DOS prompt, then the M4 core has encountered an Mbed OS error. Power cycling the Portenta usually fixes this.

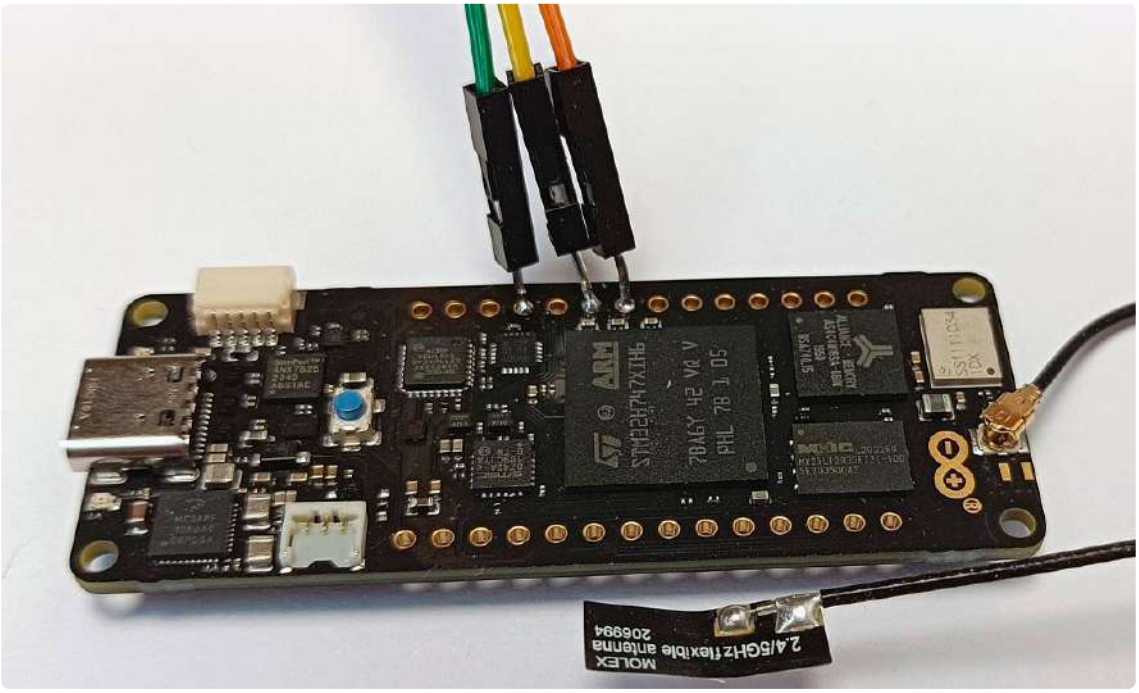


Figure 6: Rerouting debug information to the serial port on the MKR extension headers.

The virtual machine sketch loaded in Step 8 outputs status and debug information on serial port 'Serial3', which is available on high-density connector J2. This port is accessible if you have a Portenta break-out board, but if you don't, like me, it is possible to reroute the serial data to another port. There is one available on the MKR extension headers, on pins D13 (RXD) and D14 (TXD), see **Figure 6**. Change line 3 of the sketch `Portenta_x86_VM_final` to achieve this:

```
// UART mySerial(PG_14, PG_9);
UART mySerial(PA_9, PA_10);
```

With a Serial-to-USB converter you can now easily capture the data with a program like Tera Term or RealTerm (**Figure 7**).

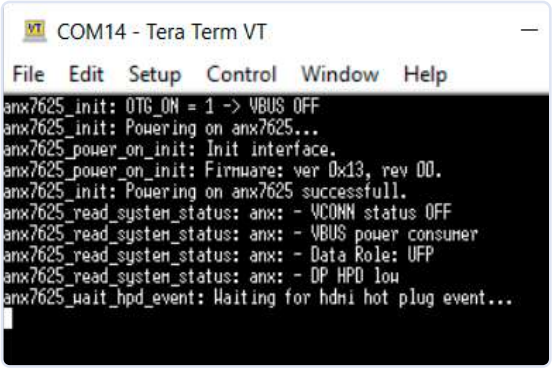


Figure 7: The virtual machine sends boot, debug and status information on a serial port.

Depending on your USB keyboard, the key mapping may (will?) be wrong. A QWERTY keyboard is expected but other types will work too. Once you have figured out where the keys are on your keyboard, typing becomes a bit easier.

220453-01

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.

Related Products

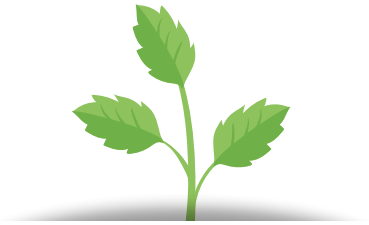
- **Arduino Portenta H7 Development Board (SKU 19351)**
  www.elektor.com/19351
- **FTDI Serial TTL RS232 USB Cable (SKU 20173)**
  www.elektor.com/20173

WEB LINKS

[1] Arduino_PortentaX86 library: http://github.com/arduino-libraries/Arduino_PortentaX86

Grow It Yourself

A Digitally Controlled, Single-Box Solution for Indoor Farming



By Dmitrii Albot (Moldova)

How about a small box that contains a "digital farm"? Read this article if you want to know how to build a small container with sensors and a microcontroller to grow plants under digital control.

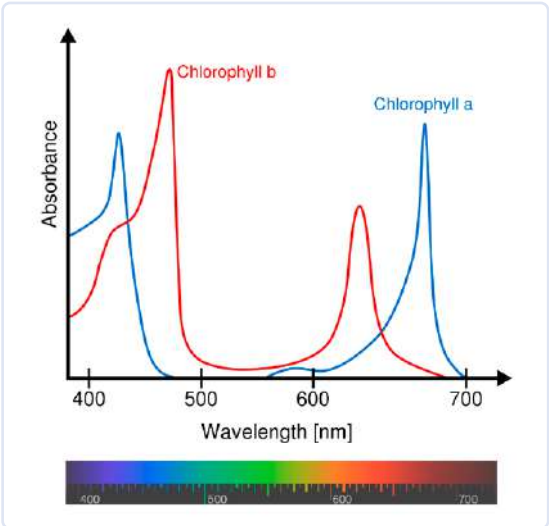


Figure 1: Chlorophyll a and b absorbtion spectra. (Source: Daniele Pugliesi. CC BY-SA 3.0 [9])

I was very inspired by MIT's Open Agriculture Initiative Director Caleb Harper's TED talk on digital farming, *This computer will grow your food in the future* [1]. The most important question he addressed in his talk was, "What if we could grow delicious, nutrient-rich food indoors, anywhere in the world?" And so, my idea was born!

Goals & Considerations

What I was trying to construct was something like a box or incubator that would create ideal climate conditions for growing plants and provide just the amount of light and nutrients needed. It should include a sunlight simulator, an irrigation system, and a climate control system in a sleek and modern enclosure.

Since chlorophyll in plants responds primarily to only two bands of light around 450 and 650 nm (**Figure 1**),

the lighting system should be a combination of red and blue LEDs to provide the perfect mix to support both vegetative and flowering growth.

I wanted to use a new watering method called Aeroponics or Fogponics [2], which uses an ultrasonic atomizer as a fog machine to water the plants through fertilizer-infused mist. The system should automatically dose the right amount of nutrients to the plants exactly when they are needed.

Of course, my system also needs pH and TDS sensors (Total Dissolved Solids). These help to achieve a balanced pH in the water reservoir, which is best for the plants, and help with nutrient dosage, respectively. Further, the system should be equipped with a connection for automatic water change, which can then be easily controlled at the touch of a button.

The system should have an air-control system that allows precise control of temperature and humidity inside the system to within 1 °C and 1%, respectively. Therefore, a temperature/humidity sensor such as DHT22 or DHT11 and a fan with appropriate control are required.

Last, but not least, the system should be controlled and monitored via a mobile app. This would be the first time in my life that I'd tried to develop such an app! The app should provide real-time information on pH, temperature, humidity, nutrients, etc., and a graphical representation over time to generate statistics and share growth progress via social media. I would also like to implement smart alerts that let me know when the system needs my intervention. In addition to displaying measurements, the system should also offer the ability to adjust its parameters.

As you can imagine, I was not the first to come up with such an idea, but good results are often achieved by improving what already exists. There are several similar projects on the market, but, despite all their advantages, they have some disadvantages: for example, they need too much space, they are too small, too expensive, or they are designed for only one plant, and so on. In any case, I wanted to develop a new, advanced open-source system that has few disadvantages and implements only optimized features. This sounds very ambitious, and it is. Also, a note: The project is very extensive and would go far beyond the scope of an Elektor article. Therefore, please refer to the project webpage at [3], where you can find a lot of details and background information as well as detailed assembly instructions.

Experiments

Dealing with plants is very time-consuming! They usually need weeks to months to grow - you must be ready for that! Even if the basic concept is clear, it still needs a few experiments in advance to see if the whole thing can succeed as planned and what adjustments to reality are required. First, I wanted to find out if growing plants with nutrient-enriched fog and LED lighting is better than a conventional system with irrigated soil and natural sunlight. In theory, this should be the case, but you shouldn't just assume anything until you try it!

In my experiments, I have thus divided plants into several groups:

- Soil + Sunlight: This group is grown in natural conditions, planted in soil, and placed on the window sill.
- Fog + Sunlight: This group is planted in a container with nutrient-rich fog and placed on the same window sill.

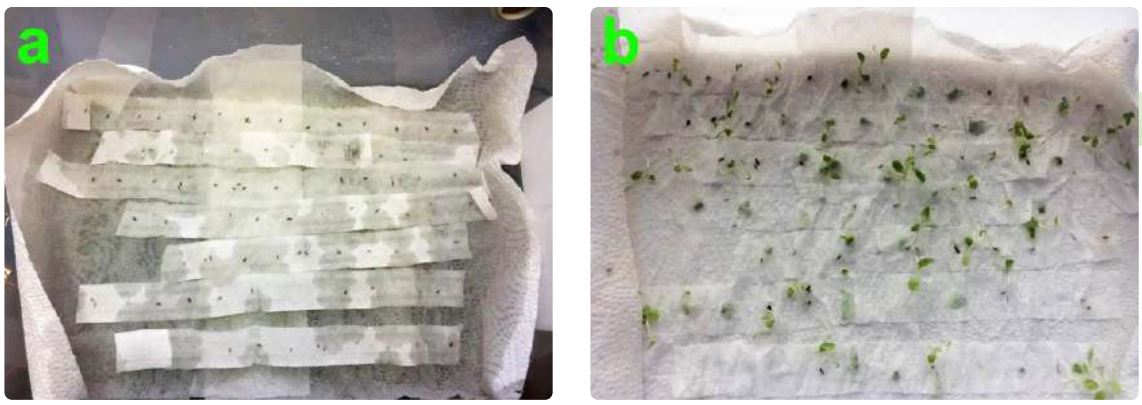


Figure 2: The first seeds germinate.

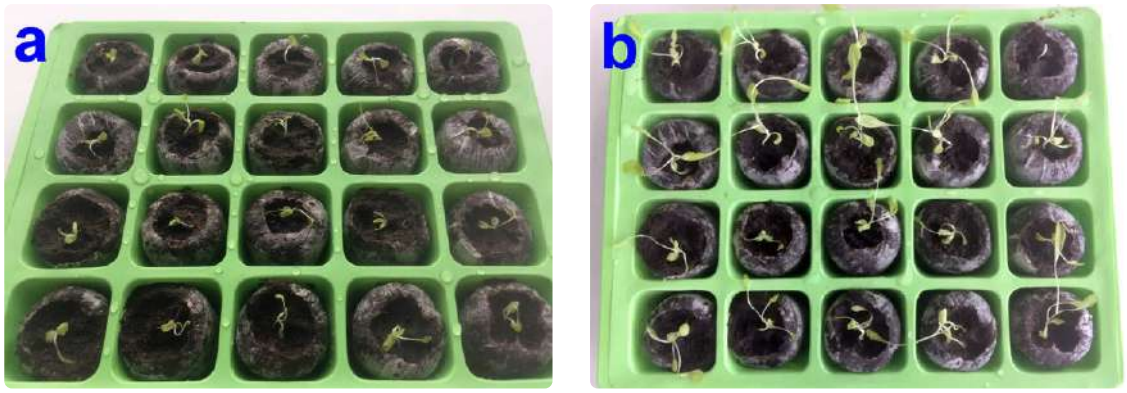


Figure 3: Seedlings transplanted into another substrate before (a) and after (b) a week.

- Soil + LED light: This group is in soil, but with artificial LED light instead of sunlight.
- Fog + LED light: This system is supposed to be the best because it combines the two main features of the proposed system.

In the plant department of the local branch of German supermarket chain Kaufland, I bought the seeds of mixed lettuce as my "guinea pigs." At this point, have some patience with me as this was the first time in my life that I planted something. Before that, I did some research and learned that you must let the seeds germinate first. So, I got a plastic container for the seeds and covered it with a paper towel. At this stage, seeds need almost 100% humidity, so I sprayed the paper towels with water and covered the whole thing with a plastic bag to prevent the water from evaporating (**Figure 2a**). I left the seeds for 10 days, and when I opened the container, I was really surprised: Almost all the seeds had germinated (**Figure 2b**)! I was as happy as a child.

Next, I had to choose the best plantlets, the ones that had the thickest stem and just looked bigger and better. A YouTube instructional recommended transplanting the small seedlings into another substrate until so-called "second leaves" form. On Amazon, I ordered coconut coir pellets. They have similar properties to soil, and what's especially cool is that they increase in size by a factor of 6 when you water them. I also ordered a box with separators specially for planting. I got it for only €2, and this organized things way better. I placed the coconut pellets inside the box's separators, watered them until they were fully hydrated, placed the little plants in the middle, and covered the box with a transparent piece of plastic, which came with the box kit. My experimental greenhouse looked like **Figure 3a**. I closed the box and left it for one week. Again, when I opened it, I was surprised: They had actually grown. **Figure 3b** proves that after one week, the difference is noticeable! Now I could place them in soil and clay balls and start experimenting with fog!

Figure 4: 3D-printed mesh trays (a) and ready-assembled test system (b).

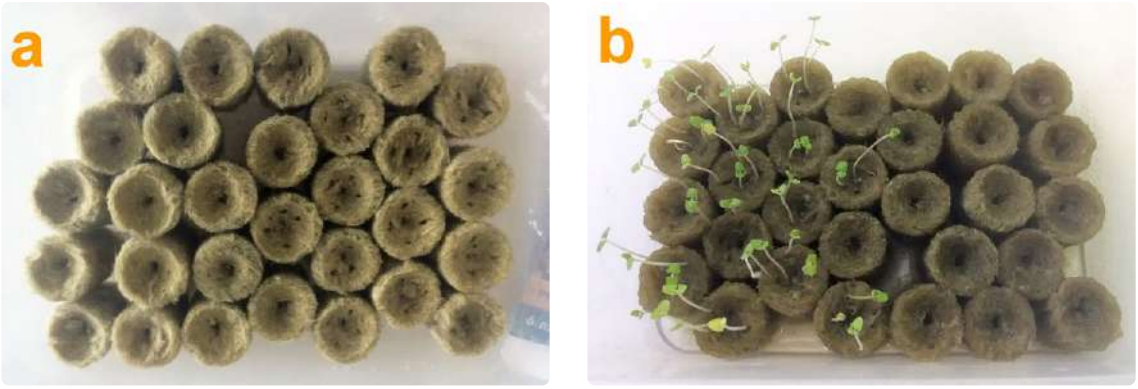


Figure 5: Next trial with new substrate (a). Only one type of seed germinated (b).

During a computer-aided design course, I designed mesh trays in which to insert the plants (**Figure 4a**). I also bought a plastic container of the appropriate size and sketched out the holes to be drilled. In **Figure 4b** and in my YouTube video [4] you can see my assembled test system, with the plants in the coconut pellets, and the clay balls and soil.

To be honest, the results could be better. Only a few plants survived my misting system. I suspect this was due to the growth medium or the fog, which was not enough to keep the clay balls and soil moist. Even worse, I couldn't do anything during the Easter vacations because the lab was closed. As a result, all of my little plants died!

After this setback, I decided to test another system using rockwool as an alternative growing medium (**Figure 5a**). After a week, I checked my box (**Figure 5b**) and found that one type of seed was successful, but another not! To speed up the process, I continued with the seeds that had germinated. In the meantime, I set up another fogging system and tried to optimize the operating time to see if I could get seeds to germinate using fogging **only**! I continue to experiment with different parameters and conditions to see how far I can get.

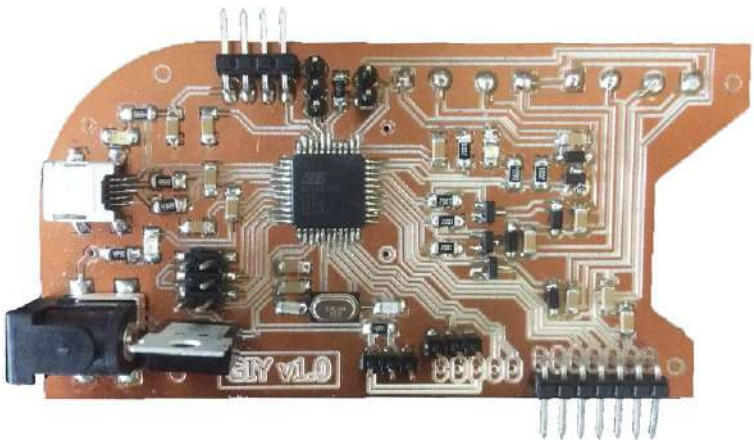
Electronic Design

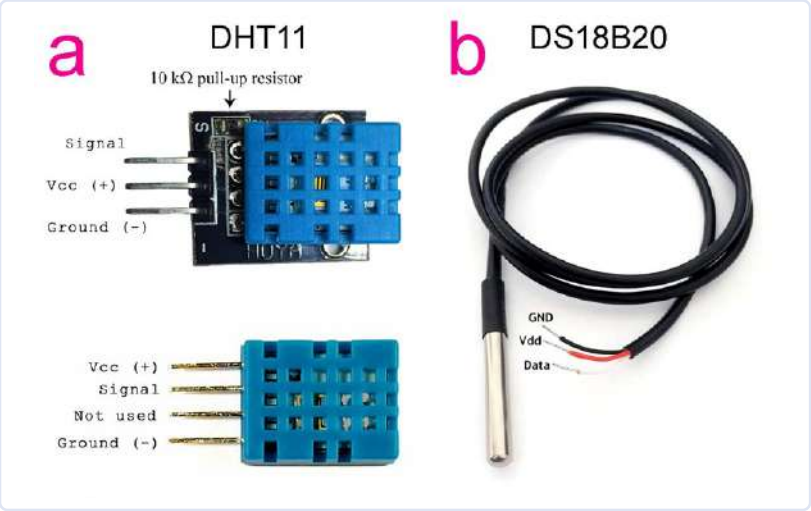
Short story: when choosing the board, I wanted to choose something between *satshakit* [5] by Daniele Ingrassia and *FabLeo* [6] by Jonathan Grinham. The former is 100% compatible with the Arduino IDE and its libraries, fabable and an improved, open-source version of Fabkit. It is not only cheaper, but also faster (16 MHz) and easier to solder. The Arduino IDE recognizes this board as an Arduino Uno.

On the other hand, *FabLeo* has very similar features, as well as hardware USB. Since I already had experience with the ATmega328P, I wanted to try something new and decided to use the FabLeo.

For the board layout, I used EAGLE. The free version is sufficient for this project and the design files can be downloaded from [3]. The fab network also maintains the continuously-updated fab.lbr library [7], which I used. The details of the circuit, as well as the making and soldering of the board can also be found at [3]. A special feature is that I needed a (switchable) step-up converter from 12 V to 24 V to supply power to the ultrasonic fog machine. On the underside of the board, I placed a Wi-Fi board I made during a network and communication week of a course I was attending. The resulting already-populated board can be seen in **Figure 6**.

Figure 6: My prototype's board, already soldered.





▲

Figure 7: The sensors: DHT11 (a) and DS18B20 (b).

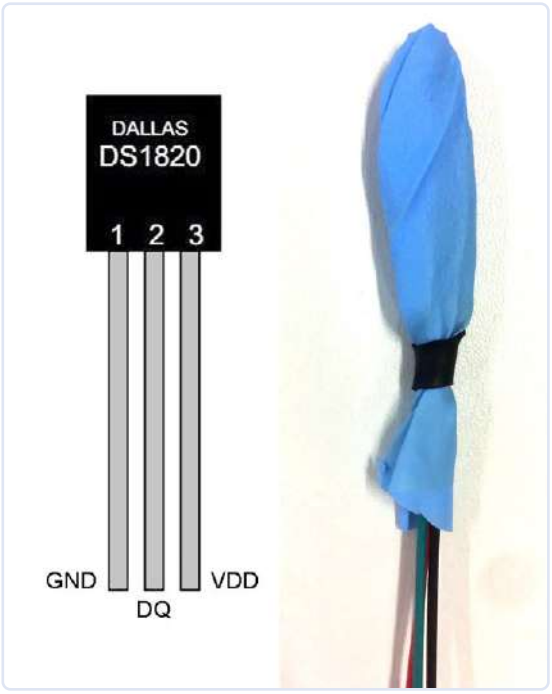


Figure 8: DIY isolation of a DS1820 sensor with one a finger of a rubber glove.

▶

Figure 9: The LCD used has a resolution of 128×64 pixels and is connected to the board using only 3 pins.

▶

Sensors

To program my board, I used the Arduino IDE. I connected the Arduino board to the USB hub and uploaded the code [3]. The first sensor I connected was the DHT11 (**Figure 7a**) to measure temperature and humidity. These sensors are very simple and slow, but great for beginners who want to do some simple data logging. The DHT11 combines a capacitive humidity sensor and a thermistor. Inside is also a very simple chip that performs analog-to-digital conversion and outputs the temperature and humidity data digitally – easy for any microcontroller to read. The DS18B20 (**Figure 7b**) is a digital 1-wire temperature sensor from Maxim IC with 9- to 12-bit accuracy (range: -55 to 125°C ±0.5°). For the use of this sensor, corresponding functions are available in the Arduino world.

The strange thing is that after I successfully programmed the sensors, the board stopped working while I was connecting all the sensors together. It took me quite a while to figure out why this happened. As a quick-and-dirty solution, I decided to use a "naked" DS18B20 sensor (**Figure 8**), which I isolated in true DIY fashion with one finger of a rubber glove.

The next sensor is an old-fashioned LDR – a rather passive electronic component. Its resistance reaches about 1 MΩ in darkness (≈0.1 lx) but drops to about 1 kΩ (≈100 lx, depending on the model). The LDR was driven as a voltage divider with a resistor and read out via an ADC input of the microcontroller.

The next task was to measure the water level. I want to have a sensor that will sound an alarm when the water tank is empty and it's time to refill the water. I simply built

my own water sensor. The basic principle of the water sensor is to measure electrical conductivity between two electrodes, which is the same for a soil moisture sensor. I figured out how I could calibrate the sensor to meet my requirements. After some experimentation, I got it right.

LCD Connection

The system needs a display. My LCD used a lot of pins - too many for my board. So, I had to find a way to connect the 128×64 screen in a different way. I was able to connect my display using only 3 digital pins, leaving the rest of the pins available for other purposes. A second advantage is that I need fewer wires to connect all parts, which avoids a rat's nest of wires. **Figure 9** shows the display I used.

pH Sensor

This is the most difficult sensor I had to deal with. It was difficult to find information about the "pH Sensor v1.1" (**Figure 10**). So, I decided to explore it myself. The probe works like a (tiny) battery when placed in an aqueous liquid. Depending on the pH, it gives off a positive or negative voltage of a few millivolts. Therefore, I had to use an opamp to amplify this weak signal to a manageable range of 0…5 V for an Arduino board. The calibration process of this sensor is described in detail in [3].

Figure 10: The "pH Sensor v1.1" is complicated to calibrate. (Source: [10])

MOSFETs

The things that need power in my system are an RGB LED strip and the ultrasonic atomizer. To connect them, I used MOSFETs controlled by digital pins. **Figure 11** shows the part of the circuit with the MOSFET outputs.

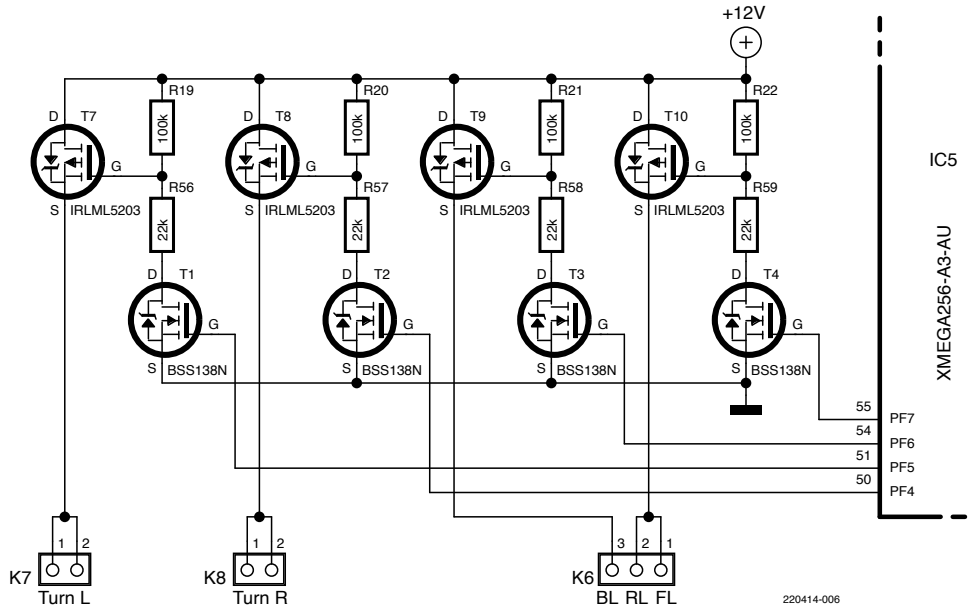


Figure 11: This part of the circuit (made with EAGLE) shows the MOSFET output stages.

Figure 12: Complete case and water container construction.

Figure 13: Hole-alignment plan for the plants.

Outer Appearance

For my final system, I used a lot of 3D design and 3D printing techniques. I had to use two different printers and played a lot with the settings until I got the results I wanted. The challenge was that I wanted it to look really good. Aesthetics was a very important criterion, as well as functionality! I also wanted the system to be able to be assembled, which made it even more challenging. I also wanted to incorporate the skills I had learned, such as 3D printing, CNC milling, laser cutting, etc.

First, I made a simple sketch on paper and then dove into Autodesk Fusion 360. **Figure 12** shows the complete container I constructed as a result of all this effort. My YouTube video [8] shows how this was 3D-printed. It has several features! First, there is enough space for plants, and I drilled a hole in the middle for the ultrasonic nebulizer because I wanted to balance the nebulizer with the tank. The thing is that the water level should be 2 cm above the nebulizer, so we don't level all the water with the height of the nebulizer (waste)! Rather, if I place the nebulizer below the level of the tank, I would have a water zero exactly at the point where it should be: zero for the tank = 2 cm above the nebulizer. In this way, all of the water is consumed!

There was much more to do. I had to 3D-print the mesh trays for the plants and laser-cut a mount for them (**Figure 13**). All in all, a lot of acrylic sheets need to be cut and drilled. More details, especially how I managed to vacuum seal the water container with a plastic plate, can be found at [3]. There are more links to YouTube videos on that page.

Go Grow It!

Figure 14 shows the LED strips I used. The circuit board is in the "electronics" section (**Figure 15**). To be as clear as possible: **Figure 16** gives an impression of how the small plants are housed in their mesh cups, which are in a matching cover plate. Now, please applaud: **Figure 17** shows the complete system in action. Isn't it beautiful?

Figure 14: The RGB LED strips used.

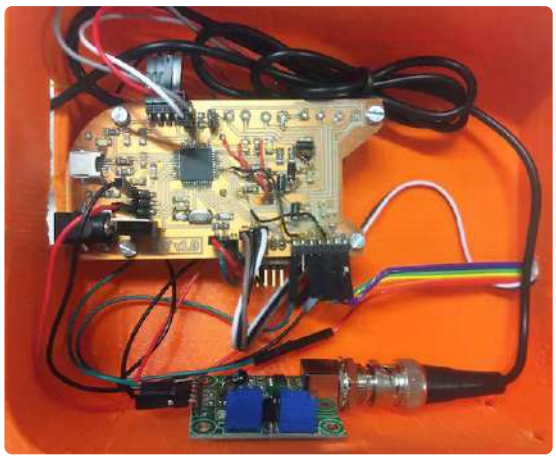


Figure 15: The board and a sensor are located in the "electronics" section.

If you're inspired by this project and want to build your own system or an optimized version of it, you'll find a wealth of information on the project webpage [3] mentioned several times earlier. Before you start, keep in mind that this is not a project that can be completed in one weekend!

220414-01

Figure 16: The little plants in their mesh cups, viewed from the side.

Figure 17: The complete system in its full splendor.

About the Author

Dmitrii Albot is a Fab Academy graduate and former FabLab Coordinator in Jordan. Currently, he is the founder of cityfarm (www.cityfarm.md) and has the mission to bring the art, science, and business of AgriTech into the cities.

Questions or Comments?

If you have technical questions, feel free to e-mail the author at albot.dumitru@hsrw.org or the Elektor editorial team at editor@elektor.com.

Related Products

Looking for the main products mentioned in this article? Arduino and Elektor have you covered!

- **ESP-12F – ESP8266-based Wi-Fi Module (SKU 17781)**
  www.elektor.com/17781
- **Elektor 37-in-1 Sensor Kit (SKU 16843)**
  www.elektor.com/16843

WEB LINKS

[1] TED Talk: This computer will grow your food in the future: https://youtu.be/KJlrd3U1Kxk
[2] Fogponics: https://en.wikipedia.org/wiki/Fogponics
[3] Project website at create.arduino.cc: https://elektor.link/arduinogiy
[4] YouTube video of my test system: https://youtu.be/LF93Xjd8avk
[5] satshakit @ GitHub: https://github.com/satshas/satshakit
[6] FabLeo board data: https://elektor.link/fableoboard
[7] fab.lbr library download: https://elektor.link/fablbr
[8] 3D-printing the boxes (YouTube): https://youtu.be/938Yz_WegH8
[9] Attribution-ShareAlike 3.0 license: https://creativecommons.org/licenses/by-sa/3.0/deed.en
[10] Photo of pH sensor v1.1: https://elektor.link/phsensor11pic

Save the Planet
With Home Automation?

MQTT on the Arduino Nano RP2040 Connect

By Clemens Valens (Elektor Lab)

You can automate your home with the right components and a little ingenuity, and this Arduino Nano RP2040 Connect-based project is an excellent place to begin. An added bonus is that you just might be helping to save the planet.

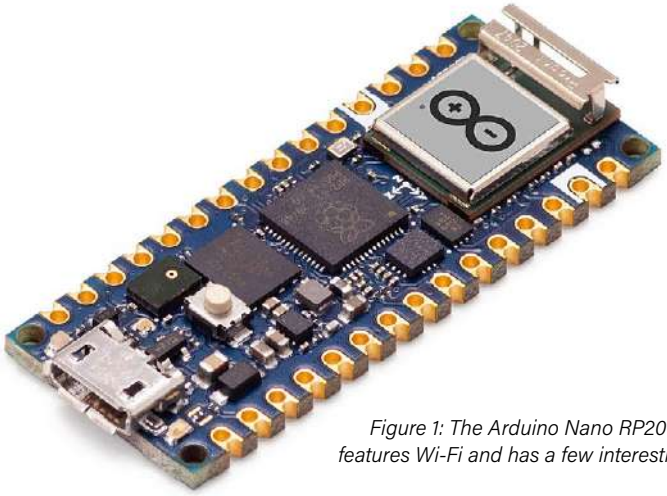


Figure 1: The Arduino Nano RP2040 Connect features Wi-Fi and has a few interesting sensors.

Today, the environment appears to be the number one concern of many people. Our planet is big, and there are many polluters over which you have no control. But there is one place where you can make a difference, and that is your home. Automating your home can make it more energy efficient and thus can help a bit in preserving the planet. Here is a way to get started.

Home automation is generally applied to the following domains:

- Climate control
- Lighting
- Energy management
- Access control
- Water allocation

The first three all concern saving energy and they do this by controlling the temperature and humidity in the building with heaters, coolers, ventilators, and blinds, and by cutting the power to lights, devices, and machines when these appliances are not needed. Water control is useful too, unless it is about sprinkling the lawn, which is always a waste.

We Need Sensors

Controlling parameters, such as the temperature in a room or the current consumption of a machine, requires sensors and actuators. In this article, I will present a sort of universal device that reads sensors and transmits the captured data to a home automation controller using MQTT. The actuators that can switch things such as motors, pumps, relays, and lights on and off are not discussed here.

For those not familiar with home automation, a controller is the heart of a home automation system that provides the glue between all the devices in the system, including you and other users (if you accept being considered a living device). MQTT is a data exchange protocol that has become quite popular on the IoT and in automation.

The Arduino Nano RP2040 Connect

The sensor device described here is based on an Arduino Nano RP2040 Connect board (**Figure 1**) that features a 3-axis gyroscope, a 3-axis accelerometer, and a microphone. Now, these may seem strange sensors to use in a home automation system, but they can be useful. (Also, it differs from the usual temperature-humidity approach of your typical IoT sensor project.)

As an example, when mounted on a door or window, the gyro can detect its movements, expected and unexpected entries and exits, or provide a window-left-open alarm. Accelerometers have many applications in vibration sensing (is the freezer motor running continuously?) and a microphone can detect sounds where there shouldn't be any (running tap) or notice a change in background noise or frequency (motor out of control?). But it doesn't really matter if you have no use for such sensors, the software framework is easily adapted to other sensors.

The Arduino Nano RP2040 Connect has Wi-Fi connectivity thanks to its NINA wireless module (a disguised ESP32) and we will use it to connect to the home's Wi-Fi network. The home automation controller is Home Assistant (HA) running on a Raspberry Pi 3B+ board in my case [1], but any other controller capable of handling the MQTT protocol can

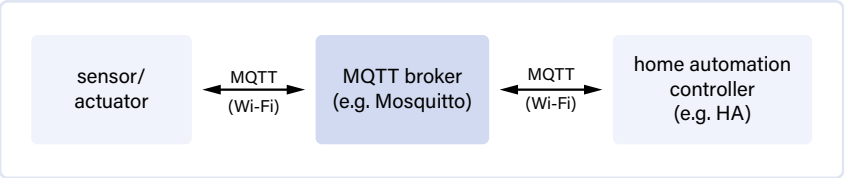


Figure 2: A high-level overview of the MQTT-based home automation system presented in this article.

be used too (which is about 99% of them). The controller also connects to the Wi-Fi network. **Figure 2** shows a high-level overview of the system.

Preliminaries & Requirements

Because saving the planet is an urgent matter, we will not lose ourselves here in all sorts of technical details about the Arduino board — we'll get straight to business.

If you don't have a home automation controller up and running already, start by installing and configuring one. This is easier said than done. For more details, see e.g. [1]. Remember that the controller needs MQTT capabilities, for which it may require an add-on. If, like me, you use HA as a controller, you can install the widely-used Mosquitto (with two 't's) 'integration' [2]. This is a so-called MQTT broker, a device that receives and dispatches MQTT messages, for example between our Arduino board and HA.

Preparing the Arduino IDE

When you have a working home automation controller with MQTT capabilities, you can move on to setting up the Arduino development environment:

1. Install the Arduino IDE. There are many versions; I used 1.8.19.
2. Using the Arduino IDE's Boards Manager (*Tools → Board → Boards Manager...*) install the '*Arduino Mbed OS Nano Boards*' board package (I used v3.2; see **Figure 3**).
3. In the IDE, select the Arduino Nano RP2040 Connect board – see **Figure 4** (*Tools → Board → Arduino Mbed OS Nano Boards → Arduino Nano RP2040 Connect*).
4. Using the IDE's Library Manager (*Tools → Manage Libraries...*), install the following libraries (the versions I used are in parentheses):
   - *WiFiNINA* (v1.8.13)
   - *ArduinoMqttClient* (v0.1.6)
   - *Arduino_LSM6DSOX* (v1.1.0)
5. Download my sketch from [3] and unpack it into the IDE's sketchbook folder. Note that the sketch consists of two files, one of which is named `arduino_secrets.h`. Enter your network credentials in this file.

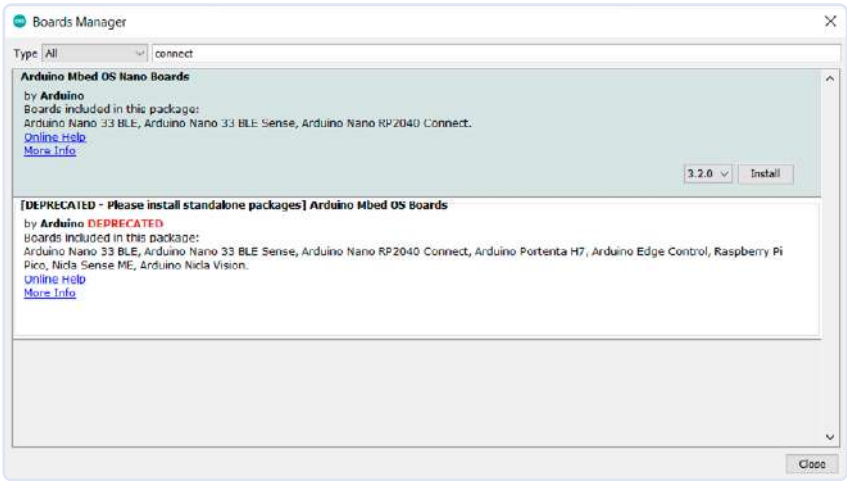


Figure 3: Use the IDE's Boards Manager to install support for the Arduino Nano RP2040 Connect.

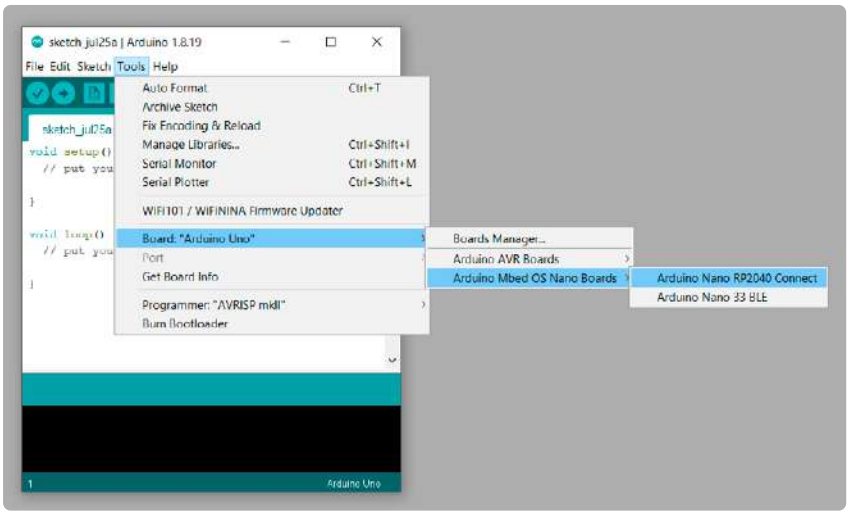


Figure 4: Before trying to upload anything to the board, make sure to select the right board (and its serial port!).

Program Configuration

Before you can compile my sketch, you must collect some information from your MQTT broker. In HA, the Mosquitto broker requires you to define a user and a password; without it, no MQTT traffic will get through. Enter the MQTT credentials in the `arduino_secrets.h` file as `SECRET_MQTT_USER` and `SECRET_MQTT_PASSWORD`. Also enter your Wi-Fi network's SSID and passphrase into this file.

In the sketch's main file, on line 37 at the time of writing, enter the MQTT broker's IP address. In HA, this is simply HA's IP address, which you can find at *Settings→System→Network* (that's where it was in HA Core v2022.8.6 with HA OS v8.4):

```
const char broker[] = 'xxx.xxx.xxx.xxx';
```

If, for some reason, you changed the default MQTT port, you will have to change the next line as well:

```
int port = 1883;
```

With the sketch configured correctly, you can compile it and upload the executable to

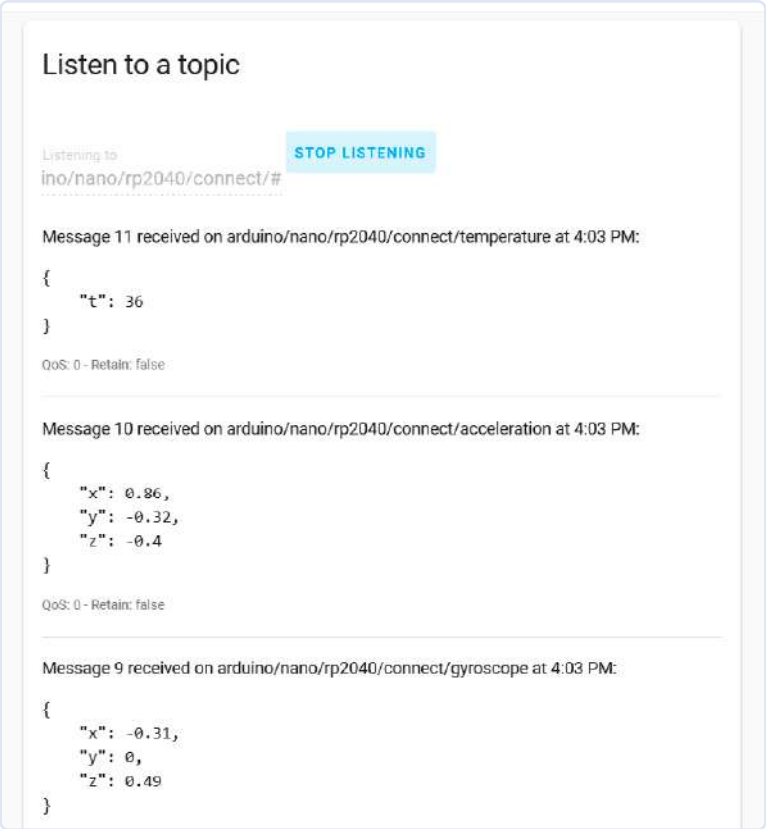


Figure 5: Receiving MQTT messages with Mosquitto in Home Assistant.

the Arduino Nano RP2040 Connect board. There shouldn't be any warnings or errors, even with the compiler set to 'verbose output' and all warnings activated (Arduino IDE *File → Preferences*).

Try It Out

Supposing that all went well, and you didn't make any mistakes when you entered the passwords, etc., the board should now connect to the Wi-Fi network and start sending MQTT messages at a rate of 0.1 Hz (i.e. every 10 seconds). The board's RGB LED flashes every time a message is sent. If the LED does not flash or remains on, something is wrong. Red indicates no network connection and blue a sensor problem.

The home automation controller should receive the MQTT messages. In HA with the Mosquitto add-on, click 'Configure' on the Mosquitto broker integration card, and scroll down to 'Listen to a topic.' Enter `arduino/nano/rp2040/connect/#` and click 'Start listening.' Messages should appear below it at a rate of one every ten seconds (**Figure 5**).

My example sketch sends gyro and acceleration data every ten seconds, and also the temperature (Ha! You didn't expect that one, did you? The LSM6DSOX IMU chip has a built-in temperature sensor, that's why). The temperature value will be higher than the ambient temperature most of the time as the sensor is heated by the board's wireless module.

When the microphone hears a loud sound, it will send a message. Clap your hands to try it out. (Don't forget that there may be up to ten seconds of latency.)

If you enter in 'Publish a packet'
`arduino/nano/rp2040/connect/incoming/xxx`
with `xxx` (the 'topic') replaced by a word or number, and then click *Publish*, you should see it appear in the Arduino IDE's Serial Monitor. You can optionally add a payload (anything will do).

Where to Go from Here?

We have now reached the point from where you will have to continue on your own. There are several options:

a. Add data and signal processing routines to the sensor software to make it send messages only if certain events are detected.
b. Add more or other sensors.
c. Add automation rules to the home automation controller to make it respond to received messages in a useful way.
d. All of the above.
e. Don't do home automation (see inset).

To help you get started with options 'a' and 'b,' I will now briefly explain the workings of the sketch. For option 'c,' please refer to the home automation controller's documentation.

Program Internals

The program starts by defining some things such as network details, but also the MQTT topics that will be used. All the messages sent by our device start with

```
arduino/nano/rp2040/connect/
```

Also, the device only listens to messages that start with this prefix. It is a long prefix, but great for debugging.

The prefix is followed by a so-called topic which can be anything. The only thing important here is that the MQTT broker or destination must listen to the same topic, otherwise the messages will simply be ignored. It is good practice to use meaningful topics. There can be as many topics as you like, and they can have an (almost) unlimited length.

Sound Processing

The sensors are read with helper functions that have meaningful names, except for the microphone, which is called PDM (it is a digital microphone). Also different with respect to the other sensors is that the microphone runs in a kind of thread in the background that continuously pumps audio samples into a buffer. The other sensors are polled by the main loop every ten seconds. A final difference between audio data and the rest is that the audio samples are filtered by a low-pass filter with a cut-off frequency of 10 Hz to make it respond only to signals with low-frequency content (booms and crashes). After detecting a suitable sound, a message is sent with topic `microphone/alarm` (plus prefix, of course). Data from the other sensors is forwarded without any filtering.

MQTT Topics

The few helper functions to compose outgoing MQTT messages are quite self-explanatory. One thing to be aware of is that the `MqttClient print` interface only applies

Don't Try This at Home

Of course, technology can help you save energy in your home and office, but it comes at a price that may outweigh its potential gain. The project described in this article uses a small Arduino RP2040 Nano Connect board that communicates over Wi-Fi with a Raspberry Pi running Home Assistant. All these devices, including the Wi-Fi router, continuously consume energy as they are always on. You can estimate the cost by looking at your energy bill. These costs are visible, and you pay for them yourself. But there are also hidden costs that people like to forget about: the resources consumed to manufacture everything you use.

I have no idea what the carbon footprint of producing an Arduino or Raspberry Pi board is, but clearly it is not zero (don't forget to include the carbon footprint of the components). Similarly, the software used by our system, including our own little sketch, was developed on many computers worldwide and stored on servers in the cloud. Like our own little system, this huge ecosystem consumes (a lot of) energy to stay online. And don't forget the resources that were used to assemble and manufacture it all. Sure, these costs are shared by many users, but they should be considered.

Where did you buy the hardware for your system? Chances are that at least some of it was ordered online. No matter where you bought your stuff, it was transported from somewhere on the planet to you, again consuming precious resources.

Finally, technology keeps advancing, and so our home automation system will soon be obsolete and must be upgraded or disposed of, adding once more lots of hidden costs.

So, maybe you save a bit of energy in your home, but, to achieve this, you probably spent much more than you'll ever earn back from it. Therefore, it is much better for the planet not to try to save energy by doing clever home automation stuff. Getting used to switching on and off the lights yourself is cheaper and allows you to walk around a bit, keeping you in shape. Also, good old common sense is a powerful planet saver.

to the payload part of a message – the topic must be assembled in another way.

All incoming MQTT messages that start with the prefix

```
arduino/nano/rp2040/connect/incoming/
```

are accepted. This is handled by the *MqttClient* library that will call the function `mqtt_message_receive` when all receive conditions have been met. By default, the sketch subscribes to the wildcard topic, '#', meaning that every topic is valid. You can change this by replacing it with more specific topics.

You can, of course, subscribe to more than one topic simultaneously. Subscriptions are managed by the MQTT broker, not by the sketch, so it is the broker that may impose limits. The amount of available memory for the sketch has no real influence.

A Little Bit of JSON

The sensor data is added as payload to the topics. I used JSON-style formatting for this, but manually, without the help of a special JSON library. The advantage of JSON is that it is understood by lots of other programs, making parsing it much easier.

Conclusion

Alright, that's it for now. Even though I tried my best to keep things simple and clear, chances are that at some point you will run into problems. The subject is more complex than this article may have led you to believe. Don't hesitate to consult the Internet for more information — there are tons of sites on MQTT and Home Assistant (and even both), which are full of useful people, hints, tricks, and tips. Also, when you run into troubles and things stop working, revert to the last-known working configuration.

220420-01

WEB LINKS

[1] C. Valens, "Home Automation Made Easy," Elektor Magazine 9-10/2020: https://www.elektormagazine.com/200019-01
[2] MQTT in Home Assistant: https://www.home-assistant.io/docs/mqtt/broker/
[3] Downloads for this article: https://www.elektormagazine.com/220420-01

About the Author

Clemens Valens is Elektor's Creative Technologist. He holds a BSc in Electronics and an MSc in Electronics and Information Technology. Clemens started working for Elektor in 2008 as Editor-in-Chief of Elektor France. He currently produces engineering tutorials and product reviews at Elektor TV. Clemens is also responsible for the Elektor Labs community website, where electronics enthusiasts can publish their work and interact with peers from all over the world.

Related Products

- **Arduino Nano RP2040 Connect with Headers**
  www.elektormagazine.com/arduino-nano-rp2040-connect
- **Clemens Valens, *Mastering Microcontrollers Helped by Arduino* (SKU 17967)**
  www.elektor.com/17967
- **Elektor Ultimate Sensor Kit (SKU 19104)**
  www.elektor.com/19104

Go Professional with ∞ PRO™

By Sebastian Romero (Arduino Pro Team)

Back in 2005, Arduino started as a low-cost prototyping solution. It enabled creatives to prototype interactive objects without having a formal background in electronics. Soon after, the electronics folks got interested in Arduino too because it also simplified their prototyping efforts a lot. But it wasn't just a matter of simplification and ease of use — the hardware abstraction layer introduced by Arduino allows you to create firmware that is portable to different platforms. On top of that, with both hardware and software being open source, this meant no vendor lock-in. It was something that many people couldn't close their eyes to.

Almost two decades later, Arduino has evolved into a mature, professional platform with a wide range of hardware and software offerings to create smart and connected solutions for any use case you can imagine. Now, Arduino Pro brings the knowledge and the experience collected internally and from the millions of community members over the years to the professionals.

Traditionally, the people who use Arduino for prototyping would need to re-create an equivalent circuitry on a standalone PCB and adapt the firmware for a final solution that can be manufactured and deployed. While it is possible to adopt some bits and pieces, a good part of the prototyped solution may need to be discarded. Arduino Pro is here to change that by providing industrial-grade hardware that can be initially used for prototyping and then be integrated as a modular part of the final solution ready to be sold on the market.

Creating just one firmware for both the prototype and the final solution not only cuts down on development cost but also minimizes time-to-market.

Leveraging Knowledge

Arduino's mission has always been to enable anyone to innovate by making complex technologies open and simple to use. This is not different with Arduino Pro. The goal is to enable the professionals in the same way as Arduino has done for the makers, educators, creators and tinkerers.

Arduino Pro can be seen as a continuation of the learning experiences that many have had in their education programs. Any knowledge acquired previously to implement projects with Arduino can be leveraged and augmented with knowledge about the Arduino Pro products. Many of the users who now use the Arduino Pro infrastructure were once makers who developed their own DIY solutions in the past. Now, they can reuse that knowledge and bring it into a professional environment within the industry. That not only shortens the time to market for developing new products but also results in minimal training necessary for people transitioning from the maker to the Pro line of products. It also means hiring people in the embedded world is easier because many of them have at least experimented with Arduino infrastructure previously.

By having an abstraction layer provided by the Arduino API [1] you can avoid vendor lock-in, because your software can be easily deployed to multiple targets, even third-party platforms. Any acquired knowledge can be re-used for those targets and applications.

And in case you don't find a solution for your particular use case, there are more than 30 million people from the Arduino community that can help out with their collective knowledge. If that doesn't suffice, Arduino has its own support and customer success team dedicated to help getting your project on the road and solve your problems.

Industrial-Grade Hardware

Arduino Pro provides a variety of hardware solutions for all sorts of use cases. Some of the target markets are the following:

- Manufacturing Machines and Processes
- Agriculture, Construction, Outdoor
- IoT Building Automation / Physical Security
- Tracking Systems
- Wearables & Light Mobile
- Prototyping

To implement such solutions, Arduino Pro provides hardware for professionals, but what does that mean? Arduino Pro products are not just certified under a variety of labels but also comply with industry standards on temperature and vibration resistance. That allows the use of these products for complex use cases with demanding technical requirements such as a deployment at the edge. Thanks to these qualities, they can be used in final solutions to be sold on the market. Examples of real-world products are shown in **Figures 1 and 2**.

Arduino provides access to professional development environments not just for developing your own software — with professional debugging tools like Lauterbach TRACE32 GDB you can find bugs in complex application scenarios. That allows crafting rock-solid software running stable on your hardware when deployed in the field.

Arduino Pro hardware is extensible, which means that you can combine different products to get access to even more features. And if that's not enough, they are compatible with hardware from the Arduino maker product line to further extend the capabilities.

Arduino teamed up with partners such as Edge Impulse [2], OpenMV [3] and The Things Industries [4] to give their users access to cutting edge technology for running machine learning models, performing Machine Vision, or connecting the products to the cloud over LoRaWAN.

Figure 1: A reliable, Arduino powered, LoRaWAN-based solution to detect parking occupancy developed by Bosch. (Source: Bosch)

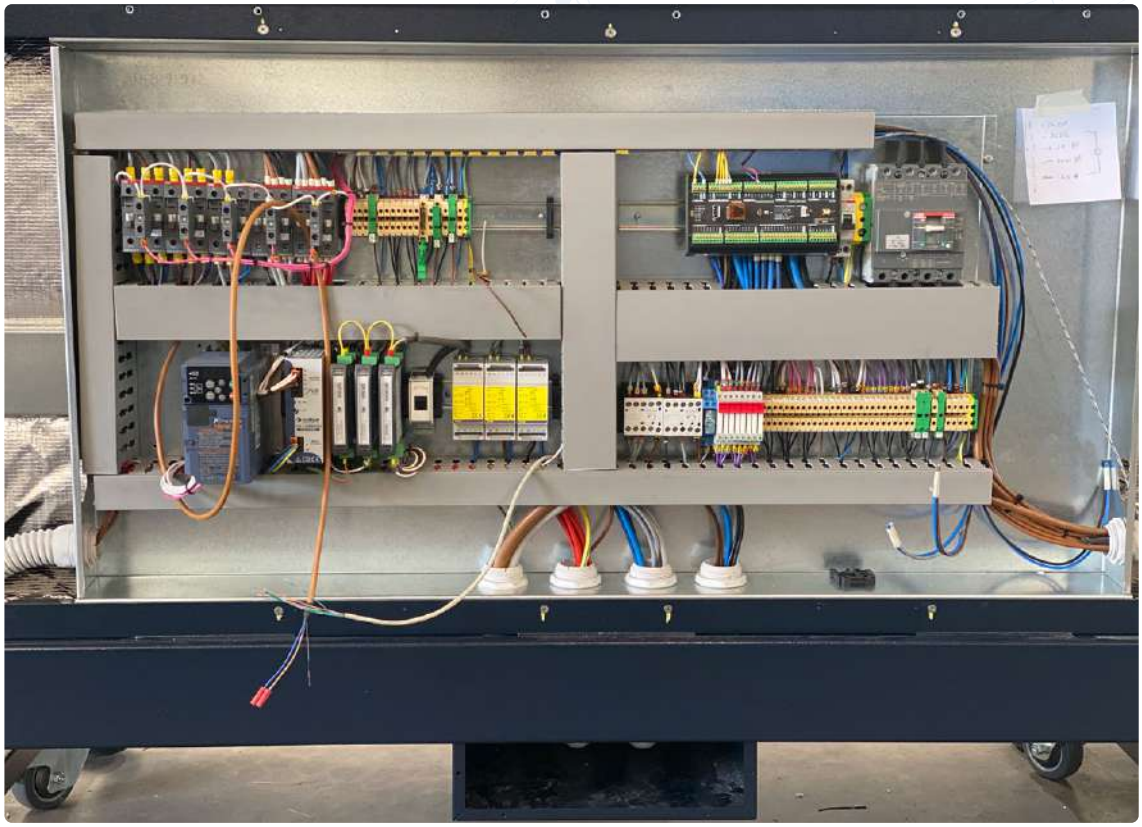


Figure 2: Smart oven built by Rinaldi Superforni using the Arduino Portenta Machine Control. (Source: Rinaldi Superforni)

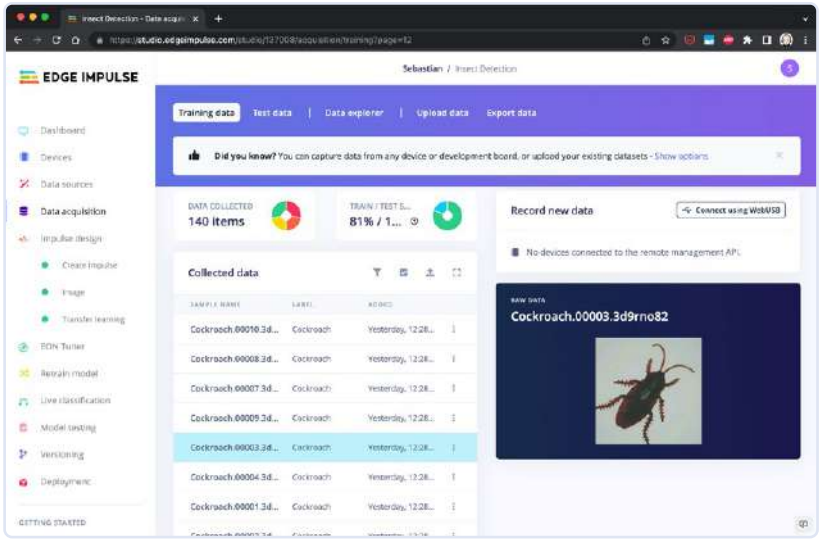


Figure 3: Data acquisition in Edge Impulse Studio for detecting insects.

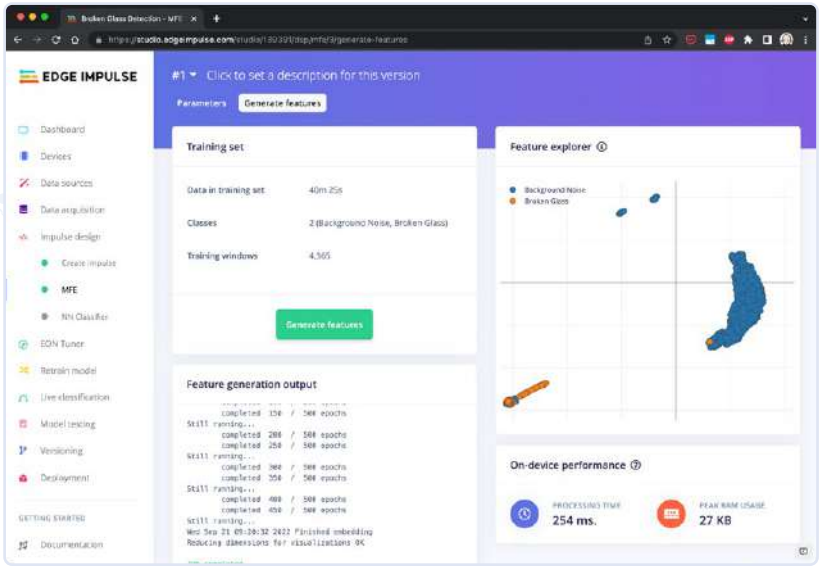


Figure 4: Feature generation in Edge Impulse Studio to detect the sound of broken glass.

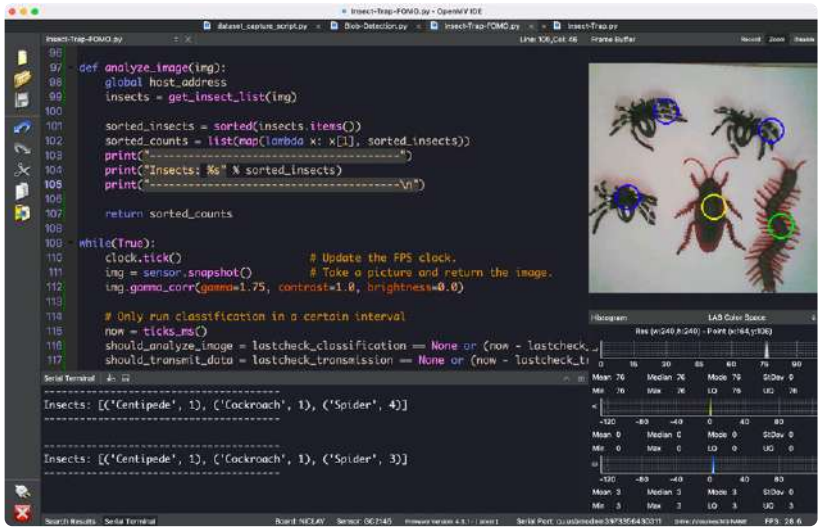


Figure 5: Detecting insects with an Arduino® Nicla Vision board in OpenMV with the help of TensorFlow Lite.

The integration with Edge Impulse Studio made it very easy to train machine learning models (**Figures 3 and 4**). That allows you to easily classify images, to find and count objects in images (**Figure 5**), to analyze sensor data such as vibration for predictive maintenance use cases, to use IMU data to understand motion and gestures and many more use cases.

Since a lot of Arduino boards have multiple sensors integrated (**Figures 6 and 7**), you can use sensor fusion to make your Machine Learning models even more advanced. By combining data from multiple sensors, you can get more accurate classifications and better understand the environment and conclude what is happening around the sensors. With Arduino Pro hardware, you can run these machine learning models very efficiently at a high frequency.

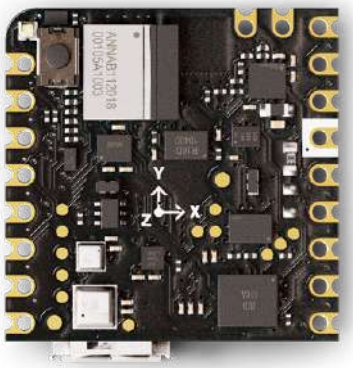


Figure 6: Arduino Nicla Sense ME with on-board sensors for measuring rotation, acceleration, pressure, humidity, temperature, air quality, and CO_2 levels.

The integration of Arduino products in the OpenMV platform allows you to run Machine Vision algorithms efficiently on Arduino hardware. You can detect objects in images and figure out their properties such as shape, color, orientation, distance and many more. In combination with Machine Learning this makes it possible to gain an even deeper understanding of what kind of objects are present in an image and how they relate to one another. It's even possible to analyze the movement of objects and determine their direction.

The collaboration with The Things Industries made it easier than ever to connect Arduino based solutions to the cloud even when deployed at the edge. This is especially useful in areas without cellular coverage. Using LoRa technology data can be transmitted with very little power whenever it's available which means that the devices can be powered with a battery or even directly with a small solar panel.

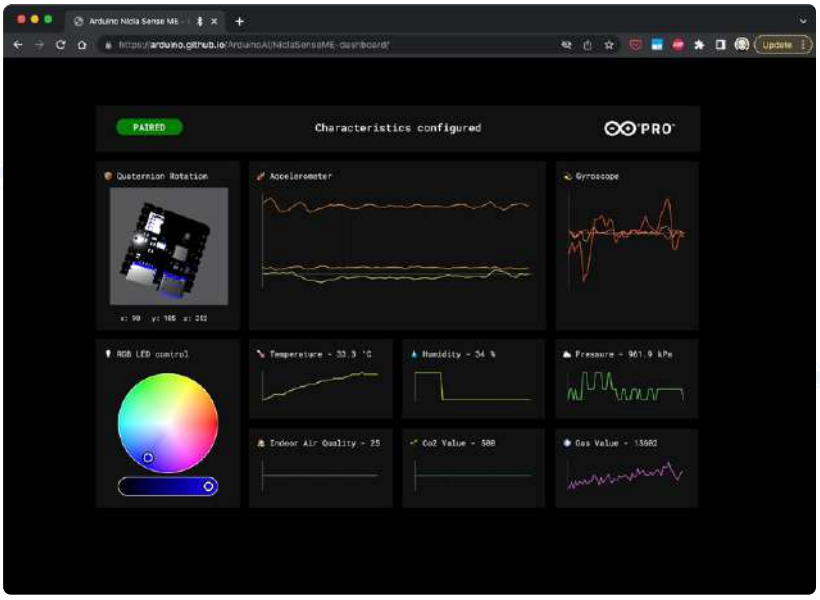


Figure 7: WebBLE-powered dashboard displaying sensor values from Arduino Nicla Sense ME.

All of that together opens up a whole world of intelligent applications to track the correct assembly of items on an assembly line, to detect wildfire, to observe animals in the wilderness, to intelligently perform maintenance on machinery before it actually fails, to enable farmers to do smart irrigation or to optimize traffic in an urban environment to just name a few examples.

Complex Technology Made Accessible

Advanced technologies often require sifting through complex documentation such as datasheets to understand how to even get started. At Arduino we believe that the professionals also deserve a better user experience. For that reason, Arduino provides sensible default configurations for all products.

For example, there are default configurations for sensors and power management controllers that work out of the box without any additional adjustments. However, if you need more fine-grained control over the settings, you can use a configuration API to do so. This is especially useful in early phases of a project when you want to evaluate feasibility or test the integration between different components or devices so you can get them up and running very quickly. Later, when you have a working solution, you can dig deeper into the configuration of the components to tune them exactly the way you want them to be. And since Arduino usually publishes the libraries and hardware cores' source code you can do modifications even at the bare-metal level should you need to do so.

In case you need to integrate different third-party sensors or actuators you can do that also very easily. Thanks to the huge Arduino community there are drivers for all sorts of standard and even exotic components available as Arduino libraries that can be used without any modification. All you need to do is to install the libraries via the Arduino IDE or the command-line interface and you're ready to go.

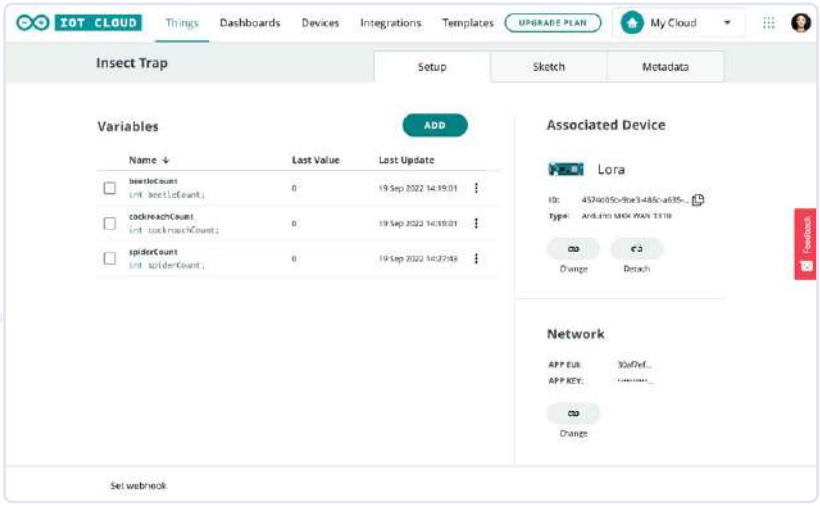
Professional IoT Solutions

Scaling solutions into the world of IoT is a huge challenge because either the setup is very complex, the stability is not ideal, or maintenance is difficult to perform. Arduino has a solution for that. It's called Arduino Cloud and it comes with a professional tier for all your business needs. It gives you access to sensor data wherever it's collected and allows you to remote control any device that you can think of as long as it can be connected with Arduino hardware.

For high-end security requirements, Arduino specifically worked out a solution that provides a root of trust at IC level using a secure element. Your connection is secure and cannot be compromised as the secrets are stored inside a dedicated chip. To ensure application-level security, Arduino recently introduced role-based access to the cloud projects so you can define exactly who should have access to what.

When you deploy IoT solutions on a multitude of devices, very likely you will want to update either the application logic, the operating system (if applicable), or both on a regular basis. For the most advanced use cases, Arduino recently launched the Portenta X8 board that runs a Yocto layer to provide a Linux foundation as the operating system. The application layer consists of Docker containers that can be updated individually in an easy and secure way. To simplify and automate this process Arduino partnered with Foundries.io [5] to provide a fleet management system that makes this a breeze. Alternative solutions that merge the operating system with the application logic have traditionally faced problems with unsuccessful updates that rendered the whole system unstable. Splitting the operating system and the application logic avoids this problem inherently.

To further simplify the configuration and setup of the devices to be used with Arduino Cloud, Arduino recently introduced the Arduino Cloud CLI. It's a command-line



▲

Figure 8: Arduino IoT Cloud "things" setup example.

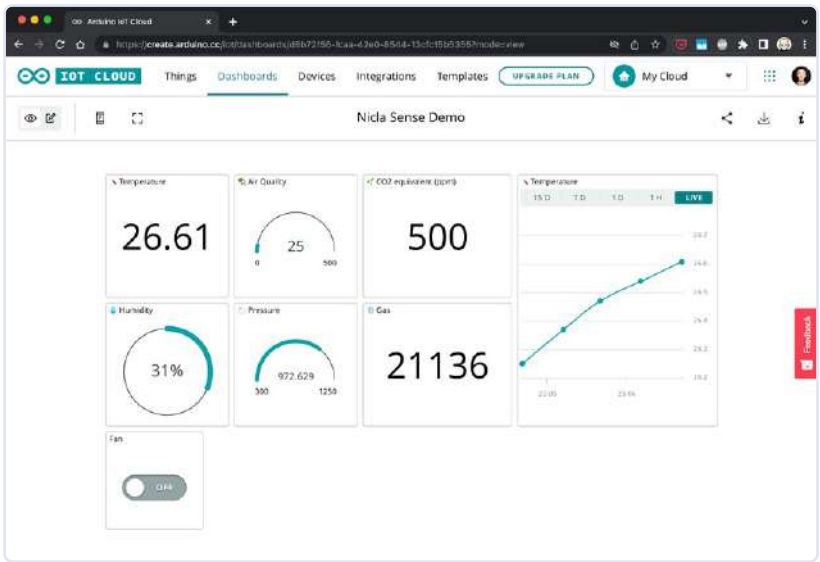
tool that allows you to use templates to set up "things" (IoT data containers) and dashboards very efficiently. For example, if you have a multitude of sensor nodes that are supposed to collect the same type of data, you need to configure them the same way. All you need to do in this case is to define one template with all the variables that you want to be populated (**Figure 8**) and deploy it as many times as needed while assigning each of the devices to the corresponding "thing". To visualize all data received from the sensor devices, you can set up a dashboard from a template and just connect it to those "things" you created previously (**Figure 9**). This approach makes it very easy to scale your IoT project once you go from prototyping to deployment.

Low Power

Especially for solutions deployed in places without direct access to electricity, low power is a key topic. For example, a sensor node installed in a place that's difficult to reach, like on a tree or buried in the soil, needs to run for months or even years on a battery. Consequently, it is important to consume as little current as possible.

Figure 9: Arduino IoT Cloud dashboard displaying sensor values from Nicla Sense ME.

▼



Arduino Pro puts a focus on low-energy components as well as on software to enable low-power applications. A board can be put to sleep until an event that is relevant to the application actually happens. During that time, all components stop doing their work to save energy. For example, the Portenta Vision Shield has a low-power camera with motion detection. When there is activity in the camera's view field, it can wake up the other components to process the vision data. This can include waking up the MCU to run a machine learning model based on an image that was taken on the camera, to classify the object in the camera image, and waking up the LoRa module which then sends that data efficiently to the cloud. In this example, the Machine Learning inference is executed on the device. Hence only little data (i.e., the result of the inference) needs to be transmitted. And that, in turn, saves energy. Such low-power solutions can run for a long time on a battery. If necessary, the battery can be charged using the on-board charging circuitry, for example, through a solar panel.

220552-01

About the Author

Sebastian Romero, head of content at Arduino, is an interaction designer, educator and creative technologist with a weakness for humans. With his team, he is responsible for crafting enthralling learning experiences to help millions of engineers, designers, artists, hobbyists and students to innovate.

Related Products

- **Arduino Nicla Sense ME**
  www.elektormagazine.com/arduino-nicla-sense-me
- **Arduino Nicla Vision**
  www.elektormagazine.com/arduino-nicla-vision
- **Arduino Portenta Vision Shield**
  www.elektormagazine.com/arduino-portenta-vision-shield
- **Arduino Portenta Machine Control**
  www.elektormagazine.com/arduino-portenta-machine-control

WEB LINKS

[1] Arduino API: https://github.com/arduino/ArduinoCore-API
[2] Edge Impulse: www.edgeimpulse.com
[3] OpenMV: https://openmv.io
[4] The Things Industries: www.thethingsindustries.com
[5] Foundries.io: https://foundries.io

Smart Ovens Take a Leap Into the Future

By The Arduino Pro Team

How a strong partnership and the Arduino Portenta Machine Control led Rinaldi Superforni to revolutionize its business.

The Challenge

Founded in 1946, Rinaldi Superforni stands out today as one of Italy's major manufacturers of professional ovens for pizza restaurants, pastry shops and bakeries. Led by the founder's three grandsons, the company is constantly looking for new ways to strengthen its products' positioning as the most technologically advanced and high-performance solutions for the increasingly demanding user.

Production-oriented and bold, they decided to partner with Arduino to develop a solution they could integrate into their ovens to make them truly "intelligent" and to provide clients with an improved experience. After some months of work side-by-side, we had perfected the Portenta Machine Control and enabled them to land on a new and exciting business paradigm.

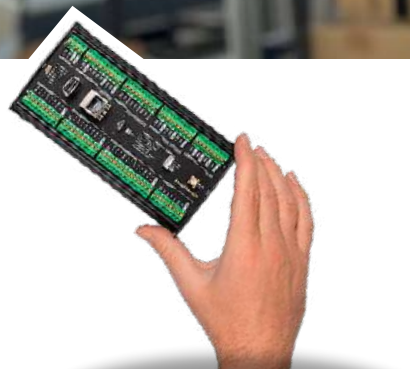
Watch the Rinaldi Superforni interview at: https://youtu.be/u5LHZVKXITY

Our Solution

Rinaldi Superforni integrated the Arduino Portenta Machine Control (PMC) into its ovens to provide a better user experience and better customer service, with top versatility and independence. Integrating the PMC into professional appliances means Rinaldi Superforni now can:

- Allow customers to connect to products remotely: bakers can start pre-heating the oven on their way to work or be alerted if the sourdough machine stops working over the weekend due to a blackout.
- Offer excellent maintenance service: a worn component can be automatically detected, triggering a preventive maintenance suggestion; the technical assistance department can check the product remotely and provide feedback or even carry out maintenance on the software remotely.

- Gather data to constantly improve: a connected device offers infinite opportunities to find out more about habits, usage, preferences and more — all precious information for the development of new models and innovations that make customers' lives better.
- Change business model completely: usage-based rental contracts are possible and straightforward, thanks to products' ability to store and transmit data.



Arduino PRO Portenta board.

Like most manufacturers, the company makes a variety of different oven models – from industrial tunnel ovens to smaller professional appliances destined to restaurants. The PMC can be used across the gamut, simply by programming it in different ways. Versatility means having one hardware, one supplier, for all the products in your catalog.

Furthermore, by testing and developing their solution with Arduino, Rinaldi Superforni acquired the know-how — and the freedom — to program and manage their products. They will never have switching costs if they decide to change: the PMC gives them the freedom and independence they need.

220404-01

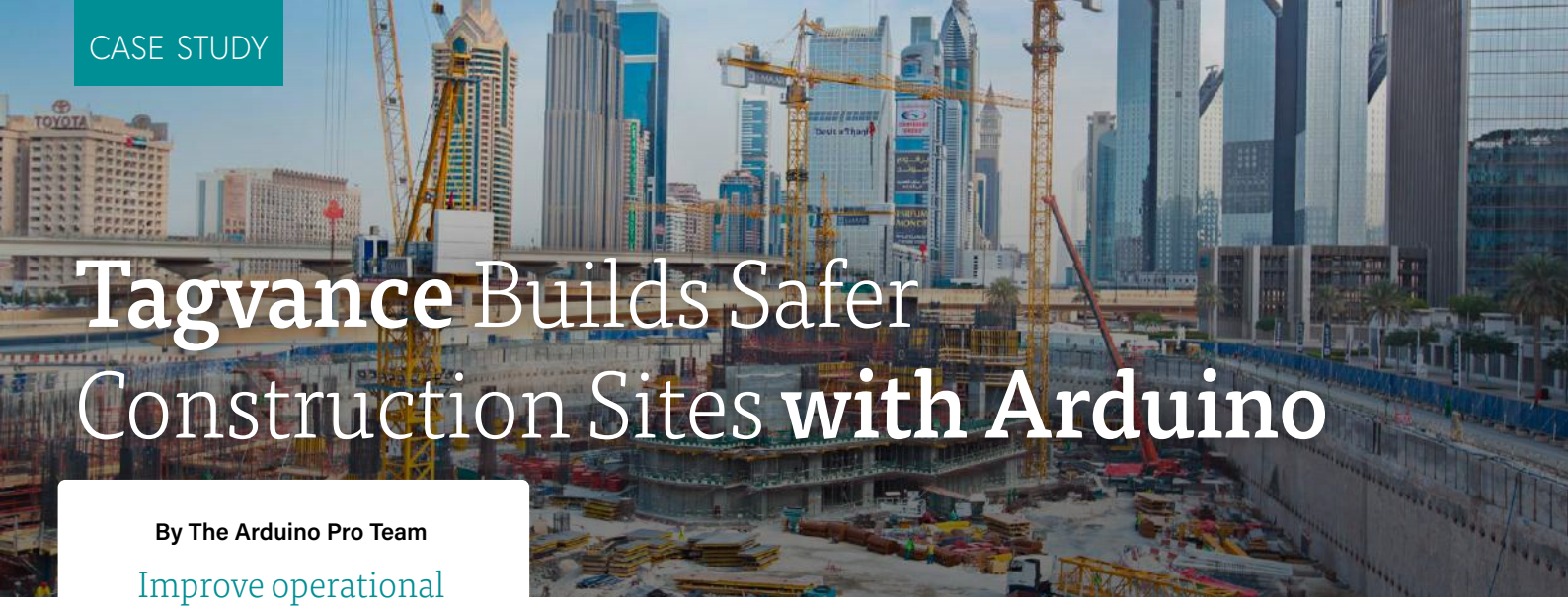
"The introduction of the Portenta Machine Control will revolutionize the way in which we sell our products. Some of our new models will be sold not as simple and static machines, but as a dynamic service."

Matteo Niscosi,
Head of R&D at Rinaldi Superforni

Related Products

- **Arduino Portenta Machine Control**
  www.elektormagazine.com/arduino-portenta-machine-control

CASE STUDY



Tagvance Builds Safer Construction Sites with Arduino

By The Arduino Pro Team

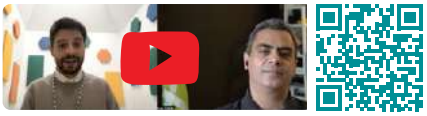
Improve operational efficiency, capacity utilization and work, safety in heavy industries with an always up-to-date digital dashboard.

The Challenge

A construction site can be a dangerous place, and increasingly stringent work safety regulations reflect the need to mitigate risks but also make the job of HSE managers more complex than ever.

Relying on manual headcounts, constantly updating asset logs, and simply checking everyone is wearing the correct protective gear upon starting their shift is time-consuming and, worst of all, not necessarily enough to keep everyone safe.

What if you could ensure fewer accidents, lower operational costs and optimize capacity utilization at the same time? Tagvance does just that. The Singapore-based innovation startup makes good use of cutting-edge technology to provide real-time data and automated reporting solutions to heavy industry clients who want to monitor the status and location of workers and assets, in complex environments such as logistic hubs, shipyards, and construction sites.



Watch the Tagvance interview at: https://youtu.be/SnsL6budrUw

Our Solution

Tagvance's system collects a wide range of environmental data by integrating the tiny Nicla Sense ME directly in workers' helmets. It also provides real-time localization via wearable tags.

Motion sensors such as accelerometers, gyroscopes and magnetometers are used in fusion with relative altitude from the barometer to classify activity and – for example – understand if a worker is in danger of falling from height. Smart cameras are deployed to validate safety compliance, and even audio sensors are used to detect equipment sounds.

IoT nodes and wearable tags running on microcontrollers use sensor fusion with TinyML to infer what is going on, based upon the data gathered matched with the location using Bluetooth Low Energy positioning. Inference data is then backhauled kilometers away using LoRa radios, so logs can be matched in the cloud against work permits, exclusion zones, and timesheets coming from ERP pipelines. In a matter of seconds, a visual dashboard is generated to report a complete digital map of locations, near-misses, incidents and metrics to accurately represent the situation.

Tagvance has made sure the system also works reliably indoors, where interferences and physical obstacles can prevent direct line of sight with people and assets: IoT nodes are spread across the tracking area and benefit from the long-range, deep penetration, and high robustness of LoRa, whilst TinyML allows results to be sent to the back office in small packets that don't require continuous transmission bandwidth.

Last but not least, the low-power components enable tags to run for several years.

Arduino's open approach and growing ecosystem of products enable companies like Tagvance to weave together a variety of different tools to build the best solution possible. In this case, the solution revolves around a hardware system based upon the following Arduino Pro components:

- Portenta H7-based IoT nodes collect information via Bluetooth Low Energy; then, thanks to the Portenta Vision Shield, the information is sent via LoRa to AWS and SAP cloud services.
- The Portenta Vision Shield's onboard cameras feed data to computer vision models — trained, for example, to detect missing safety nets and improper guardrails.
- The tiny but powerful and versatile Nicla Sense ME is embedded in worker helmets to beacon Bluetooth signal strength used for localization.

In regard to software, the programming stack also features an interesting technological mix, with the OpenMV IDE to run MicroPython using Portenta Vision Shield, the Arduino IDE to upload sketches on the Nicla Sense ME and Edge Impulse to bring together all the different sensors' data into TinyML.

220403-01

Related Products

- **Arduino Nicla Sense ME**
  www.elektormagazine.com/arduino-nicla-sense-me

Santagostino Breathes Easy

with Remote Monitoring that Leverages AI for Predictive Maintenance

By The Arduino Pro Team

Here we discover the Arduino Nano RP2040 Connect at the core of a reliable, cost-effective, and flexible solution to guarantee optimal HVAC performance across a network of medical centers.

The Challenge

A modular, scalable monitoring solution to constantly and automatically check ventilation is working properly: the key to ensuring air quality in medical centers.

Counting on a team of 200 employees and 1,200 doctors, Santagostino operates a network of 35 medical centers in Italy, mainly distributed across Milan, Rome and Bologna. Just over a decade after they began, they are able to offer a wide range of diagnostic tests and procedures — but perhaps the most basic service they need to guarantee is a comfortable and safe environment for the thousands of patients that come to their facilities.

Air conditioning and ventilation have always played an important role in this: while most of us have become hyper-aware of the importance of air quality only after the pandemic, the Engineering & Technical department at Santagostino has always worked hard to ensure the correct operation of these systems.

Their challenge? Staff and customers would often notice if the heating or air conditioning were not regulating temperatures properly, but it could take days or even weeks before a technician detected a malfunction in mechanical ventilation during a routine, on-site check. The company, therefore, set out to find a 24/7, remote monitoring solution, which also had to be modular and scalable to adapt to the variety of HVAC systems installed across their centers.

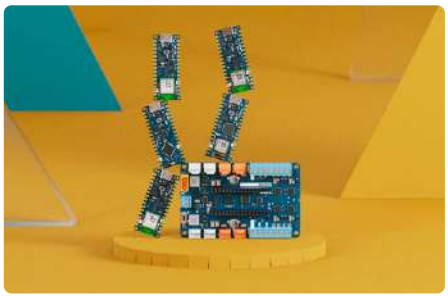
Our Solution

Santagostino developed internally an Arduino-based solution to monitor HVAC systems across 35 locations in Italy, feeding AI the data to detect — and even predict and prevent — any malfunctioning.

Watch the Santagostino interview at: https://youtu.be/S6bcF-9wTxs

The Arduino Nano RP2040 Connect, protected by a 3D-printed case, was placed inside heat pumps, A/C units, and mechanical ventilation systems at the network's centers. Regardless of the type, brand, and model used at each location, the installation was carried out easily, with no need to involve external specialists or alter the machines in ways that would effectively void their warranty.

Tiny, powerful, and reliable, the Arduino Nano family is perfect for wearables, drones, scientific experiments, and any other IoT/AI solution that needs to shrink size, not performance.



Arduino Nano family of boards and the associated Carrier board.

Santagostino chose the Arduino Nano RP2040 Connect because it includes both a Wi-Fi module and a high-quality accelerometer, while still standing out with an extremely compact form factor and competitive price.

In addition, its ARM processor is able to collect data from Santagostino's centers, send them to the Edge Impulse platform, and run the ML algorithm that autonomously reads and interprets systems' operational status.

220402-01

"Our solution is very simple but not obvious. Using Arduino allowed us to apply it to any type of system – whether analog or digital, made by any brand."

Andrea Codini, CTO at Santagostino

Related Products

- **Arduino Nano RP2040 Connect**
  www.elektormagazine.com/arduino-nano-rp2040-connect

Security Flies High with RIoT Secure's MKR-Based Solution

By The Arduino Pro Team

Arduino's open-source products are at the core of a solid answer to hacking and cyber risk in airports.

The Challenge

Airports clearly give security the utmost importance: stringent rules must be rigidly followed — but also quickly updated as needed, without creating vulnerabilities.

Stockholm-based company RIoT Secure (http://riotsecure.se) was founded to address the current and potential security issues our world faces, as billions of objects are connected to the Internet and IoT emerges as one of the strongest growing trends of our time. For them, working with SAS (Scandinavian Airlines) Ground Handling provided the ideal high-constraint project to prove security can be embedded at the core of any IoT solution.

In airports, service vehicles are tracked both for billing purposes and to ensure compliance with safety and security protocols – which constantly evolve. For example, geo-fencing boundaries must be checked in real time to avoid anyone entering forbidden zones, and staff must use RFID-based security badges to access and operate the equipment.

Therefore, in designing a new solution, the critical requirement RIoT Secure was asked to meet was to ensure that all network communications were secure, and that firmware updates could be performed over-the-air, instantly, and across the entire fleet of vehicles.

Our Solution

RIoT Secure developed a secure device lifecycle management platform based on Arduino MKR boards, for communications and over-the-air updates specifically targeting resource-constrained microcontrollers.

RIoT Secure: Secure Device Lifecycle Management with Arduino. Watch it on: https://youtu.be/RPUgTsawp5E

The Arduino MKR family was chosen for its modular approach and the capability to offer both Wi-Fi, 3G, and NB-IoT connectivity and a secure element to ensure a root of trust — an excellent foundation for the platform's network freedom.

RIoT Secure rewrote the networking libraries underlying firmware — which utilizes FreeRTOS for multi-threading support — with security-by-design in mind. This allowed them to create a solution that ensures:

- robust and failsafe communication
- long-term device reliability
- freedom to choose the best network topology
- freedom to use the most appropriate microcontroller for each task
- complete isolation from hacker attacks, minimizing security vulnerabilities

Arduino MKR boards.

Finally, while updates are made incredibly simple by using the Arduino IDE and RIoT Secure's lifecycle management platform, the solution's flexibility reaches future-safe levels with the freedom to upgrade or replace the Arduino MKR device as technology evolves, independently of firmware development.

RIoT Secure's lifecycle management platform is licensed to Ingwaz who has the business relationship with SAS ground handling. Ingwaz is a company co-founded by EIT Digital, which supports and spearheads breakthrough technology within digitization for Europe.

220401-01

"SAS Ground Handling can now ensure their equipment are securely connected to the cloud, and that they can enhance the safety and security protocols implemented at the edge in a matter of seconds."

Aaron Ardiri, CEO of RIoT Secure

Related Products

- **Arduino MKR Family**
  www.elektormagazine.com/arduino-mkr-family

CASE STUDY

Open-Source Brings a New Generation of Water Management to the World

By The Arduino Pro Team

Here's how smart irrigation took a leap forward thanks to the collaboration between Challenge Agriculture and Arduino.

The Challenge

The world of agriculture is changing at an ever-increasing pace. To take irrigation management to the next level, Challenge Agriculture embraced innovation.

The company has leveraged its agronomical, ecological and economical skills to contribute to the evolution of tensiometric irrigation management for the past 35 years. About 10 years ago, the company's CEO Xavier Eftimakis realized that a more powerful tool was needed to observe soil and monitor irrigation to achieve optimal yields, save water, and conserve fields' quality. He decided to invest in developing Challenge Agriculture's own board: the **R2-DX**, inspired by field experience and open-source electronics. But with the many evolutions of the agronomic industry and environmental issues more pressing than ever, Eftimakis felt it was time to take another step forward.

"My clever cousin Mike Eftimakis was already using Arduino, and I always loved the open-source concept: it is the philosophy I follow in sharing my experiences in agronomy. My son Marc, who is a developer, joined the project as well. With Arduino's help, we worked together efficiently to make our board." — Xavier Eftimakis, Founder and CEO at Challenge Agriculture

Our Solution

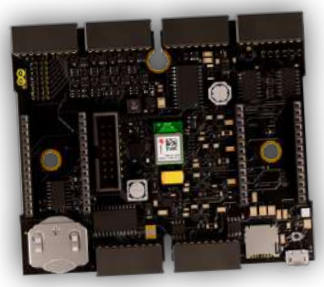
Challenge Agriculture and Arduino partnered to develop Irriduo, a smart solution for irrigation and many other water management applications.

Learn more on Irriduo at www.challenge-agriculture.fr/en/irriduo/

Irriduo is a tool for the professional observation of soil using tensiometry, a technique universally used to measure the natural tension of water in the soil. Its board allows for sufficient inputs to easily install sensors at different depths underground, measure humidity, and offer a clear, real-time image representation of constantly changing values. Multiple sensors per field provide accurate readings for 6 crop cycles of 3-4 months each — i.e., 4,000 measurements — or for as long as 4 years in the case of perennial crops. Irriduo's data gathering and processing capabilities support informed decisions on actions to take. The **Arduino Edge Control** can be used to deploy AI on the edge. It can be expanded with 2G/3G/CatM1/NB-IoT modems, Lora, Sigfox, and WiFi/Bluetooth connectivity and managed remotely via the Arduino or third parties' clouds. The Arduino Edge Control has an infinite variety of applications beyond agriculture: power plants, construction sites, parking lots, and swimming pools are only some of the possible contexts where it can make a difference. Specifically for precision farming, the Arduino Edge Control can:

- optimize the use of water, fertilizers, and pesticides
- improve plant health
- reduce human error
- automate tasks
- adapt to weather conditions
- share real-time insights on crop conditions

220397-01



The Arduino Edge Control is a remote monitoring and control solution, optimized for outdoor environments. It can be positioned anywhere and is suitable for precision farming, smart agriculture, and other applications requiring intelligent control in remote locations. Power can be either supplied via solar panel or D/C input.

In need of a similar solution?
Interested in what we do?
Contact the Arduino Pro Team at
www.arduino.cc/pro/contact-us

Senso

Detect Deforestation with Sound Analysis

By Andrei Florian (Ireland)

Illegal wood logging is a problem in many countries. The Senso project is a potential solution. The Arduino MKR Fox-based device alerts authorities when it detects logging noise.

When I went to Romania to climb some hills, I noticed that many people seemed upset about illegal wood logging, and I thought that there might be something I could do about it. After looking at the problem, I decided to identify logging by the sound produced by the power tools cutting down trees. The solution I came up with is described below.

The Sound of Deforestation

Deforestation is one of the most significant problems our generation faces. Forests are cut down throughout the world to make space for agricultural fields and housing. Parts of forests are set on fire to create space and illegal logging happens on an international scale. Deforestation is directly linked to climate change as burning organic material releases a lot of carbon dioxide, a greenhouse gas.

Logging is a big concern in both developing and developed countries. Romania is a good example. Although logging is regulated in Romania, part of it is done illegally. Romania accounts for 65% of Europe's virgin forests but they are rapidly disappearing, being cut down.

Most of the cutting is done using chainsaws, which produce a lot of noise. This is where Senso comes in, a low-power device equipped with a sound analysis module that can isolate different bands of sound. The device is programmed to identify the sound of logging machinery and tools such as chainsaws and alert the relevant authorities when such a sound is detected. Samples are taken every fifteen minutes. If a suspicious sound is detected, the device sends an event to the back-end where the areas where logging takes place are marked on a map.

Senso combats illegal logging as authorities can be instantly alerted when logging takes place, allowing them to take action faster than ever before and to concentrate on areas where logging is most prominent.

Device Overview

I used the Arduino MKR Fox (**Figure 1**) for this project because of its great low-power features. I have used it before in several projects that rely on low-power communication. It is very easy to use Sigfox on this device as it is built-in and Arduino libraries allow you to get up and running in no time.

In this case, I decided to power the device through a breadboard power supply with 5 V as the modules attached would drain too much power if powered by the board itself. As this is a prototype, the battery life of the device is not long enough to be used out in the field. I am still working on turning the sensors off when the device is in sleep mode to preserve energy.

Figure 1: The Arduino MKR Fox is at the heart of this project. (Source: Arduino)

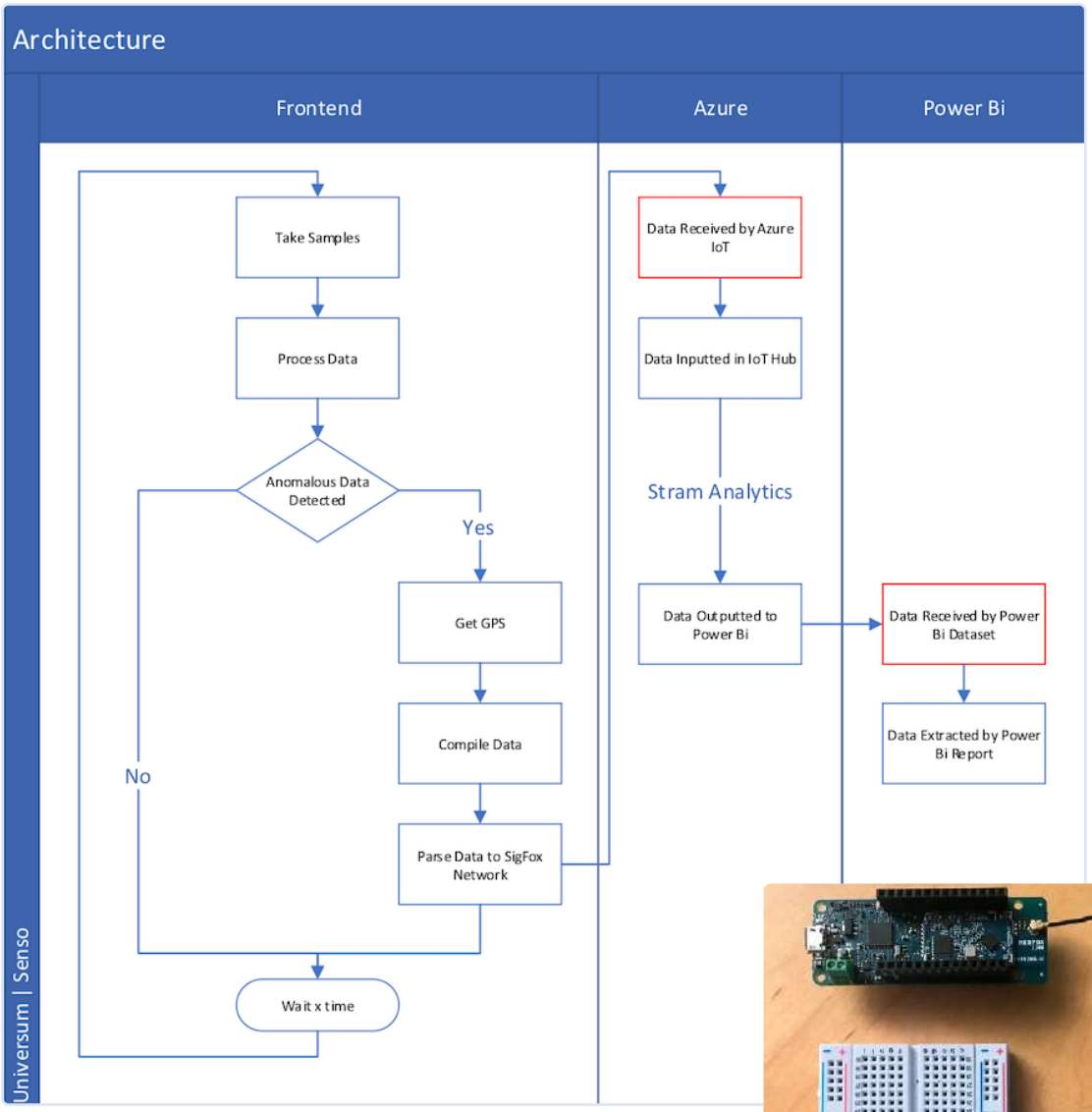


Figure 2: The system architecture as a flow chart.

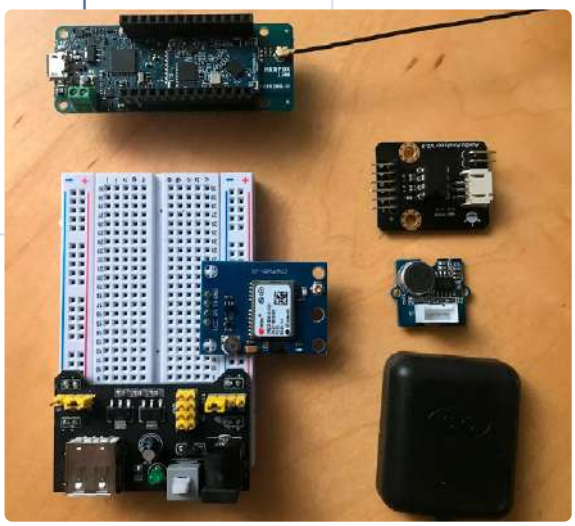


Figure 3: The components used in this project.

Project Architecture

The project is split into a front-end and a back-end. The front-end consists of the device in the forest taking sound samples, and the back-end consists of Sigfox, Microsoft Azure and Microsoft Power BI to process and display the data (**Figure 2**).

Front-end: The device wakes up at regular time intervals to listen for sound. It looks for activity in specific frequency bands that correspond to chainsaws and similar tools. If a match is found, the device sends its location and battery level to the cloud. Then it goes to sleep until it is time to wake up again.

An interesting feature that might be added would be position tracking. Using accelerometers, the device could detect when the tree it is mounted on is cut down. When this happens, it could send its location to the back-end every 10 minutes or so allowing the authorities to track the cut-down tree while it is being transported.

Back-end: The event detected by the sensor is received by the Sigfox back-end, which forwards it to Azure IoT by means of a callback function. Azure IoT then assigns the data to a hub. A data streaming service queries the data and inserts it into a Power BI dataset. Finally, Power BI extracts the values from the dataset to create a report that can be displayed.

Going further, a messaging application could be invoked to text the authorities whenever an event is received by the back-end. This would allow forestry workers to know exactly where to go when a tree is cut down.

Constructing the Project

Enough talk, let's build the project. This project requires quite a few components (**Figure 3**): an Arduino MKR Fox board, a GPS module plus antenna, a GSM antenna, an Audio Analyzer Module from

DFRobot (see Figure 3), and a microphone. A breadboard is used to wire everything together (**Figure 4**). It is powered from a 9 V battery though a 5 V/3.3 V breadboard power supply module. Note that the microcontroller is powered by the breadboard power supply.

I use a cheap GPS module I found online for €10. It is easy to use but it takes a while to lock on to satellites. A good antenna is needed.

The Audio Analyser Module provides the sound level in seven frequency bands: 63 Hz, 160 Hz, 400 Hz, 1 kHz, 2.5 kHz, 6.25 kHz and 16 kHz. After rigorous testing I found that chainsaw noise typically falls in the 2.5 kHz to 6.25 kHz range. During testing by playing back recorded chainsaw sounds, the 6.25 kHz band would show spikes. I was then able to come up with a calculation that would prevent other machinery such as cars and natural sounds from being recognized as a chainsaw. Although not perfect, it does a great job at rejecting similar sounds.

The code is split into three main sections, each described below.

Sample and Process Sound

This part of the program is executed every time the device wakes up. The device first takes samples from the sound sensor and then processes the captured data by comparing the value of the target frequency band to the other bands. This is done with the following lines of code:

```
long comparison = ((valueMean[0] + valueMean[1] + value-
Mean[2] + valueMean[3]) / 1.9);
if (valueMean[5] > comparison) ...
```

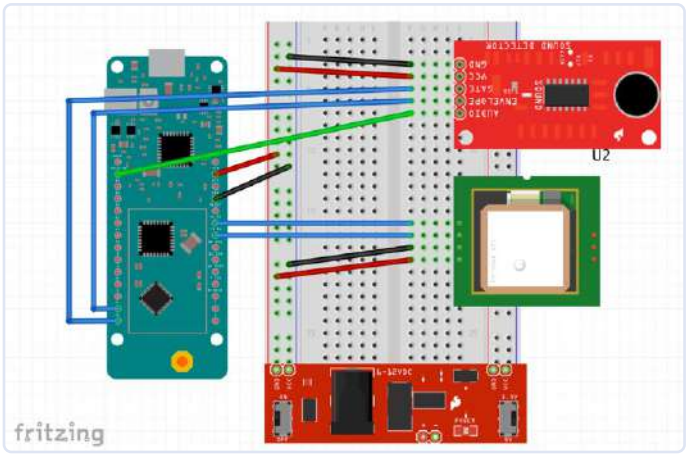


Figure 4: Wiring diagram of the Senso prototype.

The mean values of all frequency bands (except the target band) are added and divided by 1.9. If the value of the target band is bigger than `comparison`, the detection is considered positive (**Figure 5**).

Get GPS Position and Battery Level

The function `getGPS()` extracts device's GPS coordinates and checks them for correctness. Then the battery voltage of the attached battery (if any) is measured by the function `getBatteryVoltage()` and added to the data structure to be sent to the cloud.

Note that if the Arduino board is powered through VIN, the battery voltage will be equal to zero. It must be powered through its power pins on the breadboard to display the voltage.

Send Data to Sigfox

The functions `encodeData()` and `sendToSigfox()` encode the position data and battery voltage level in byte format and send it to the Sigfox back-end.

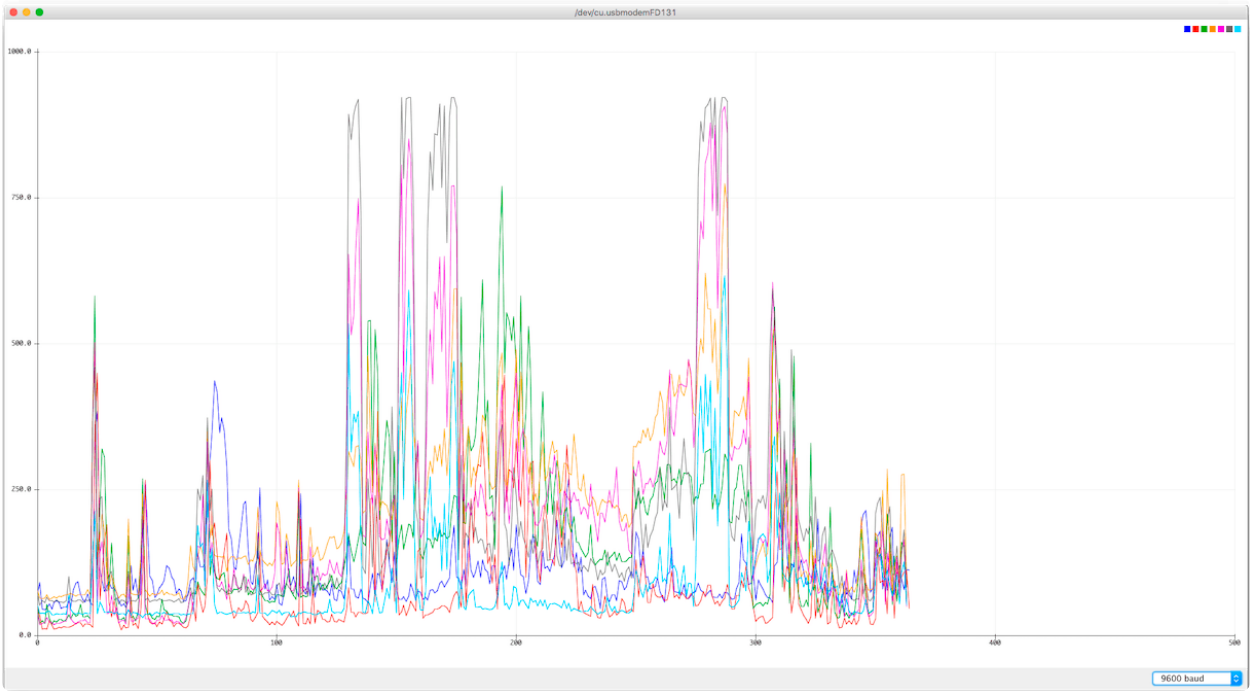


Figure 5: Sound testing. The spikes show chainsaw activity.

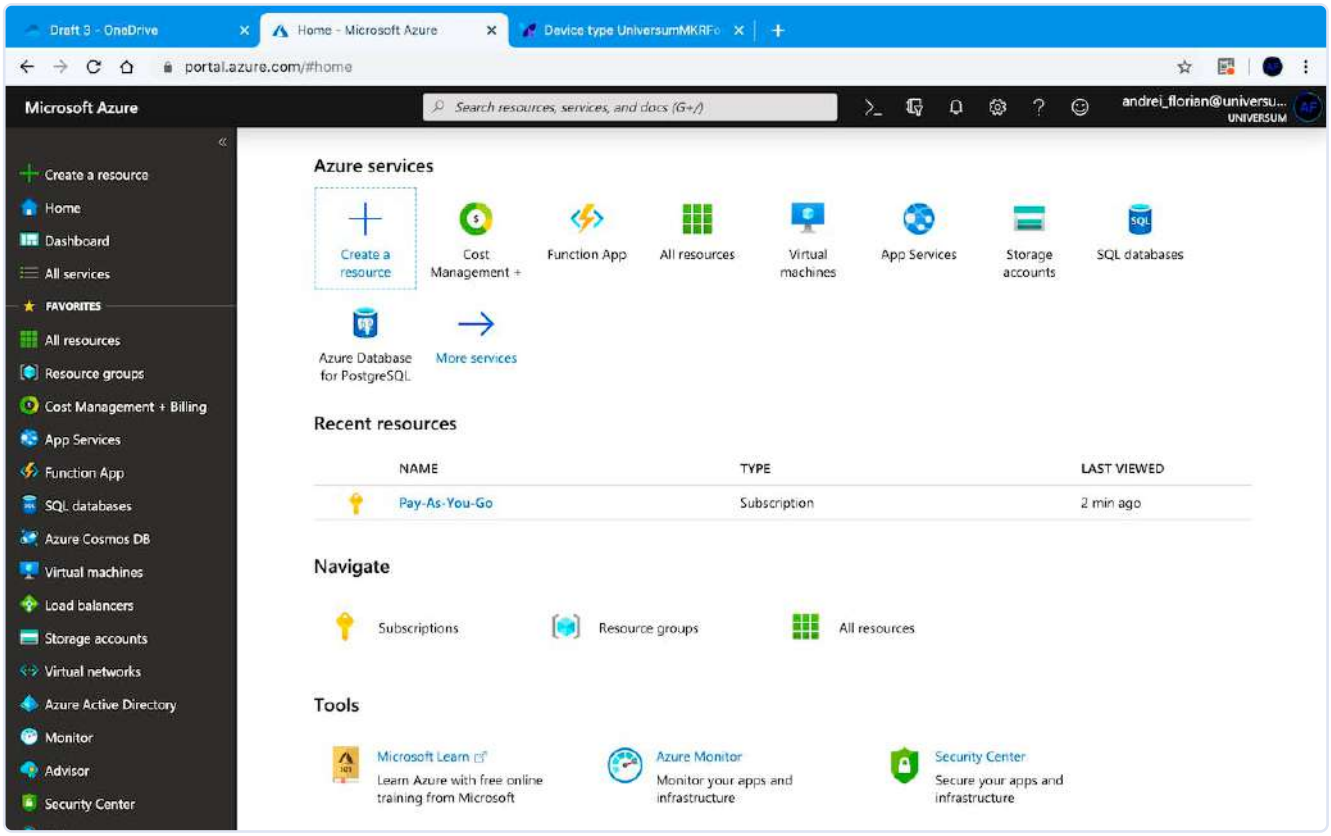


Figure 6: The first step of setting up a Microsoft Azure account. Nineteen more screenshots are available at [1].

Configuring the Program

There are two configuration variables that must be specified:

`proDebug` – set to true if debugging. If set to true, the device must be connected to a computer over a serial connection and the Arduino IDE's Serial Monitor must be open for the code to be executed. Set to false otherwise.

`nrSamples` – defines the number of samples to take. Each sample consists of 100 readings.

Preparing Microsoft Azure

This project uses Microsoft Azure as a back-end. There are a few prerequisites, though:

- Azure Account
- Azure subscription
- Basic knowledge of the application

Since a picture is worth a thousand words: **Figure 6**. Twenty screenshots would take up too much space here. Therefore, please refer to [1] for details on setting up an Azure account and the IoT hub that will store the data received from the device.

Preparing Sigfox

We also need to prepare the Sigfox callback. Again, there are a couple of prerequisites:

- A Sigfox back-end account
- Register the device in the back-end

For the same reasons as for setting up the Azure account, please refer to [1] for details on setting up a Sigfox account (**Figure 7**).

To inform the Sigfox back-end about the format of the data we are sending, three floating point values (latitude, longitude and battery level), enter the following line in the *custom-data-config* field:

```
geoLat::float:32:little-endian geoLng::float:32:little-endian battery::float:32:little-endian
```

Figure 7: Opening the Sigfox portal is step 1. Nine more steps can be found at [1].

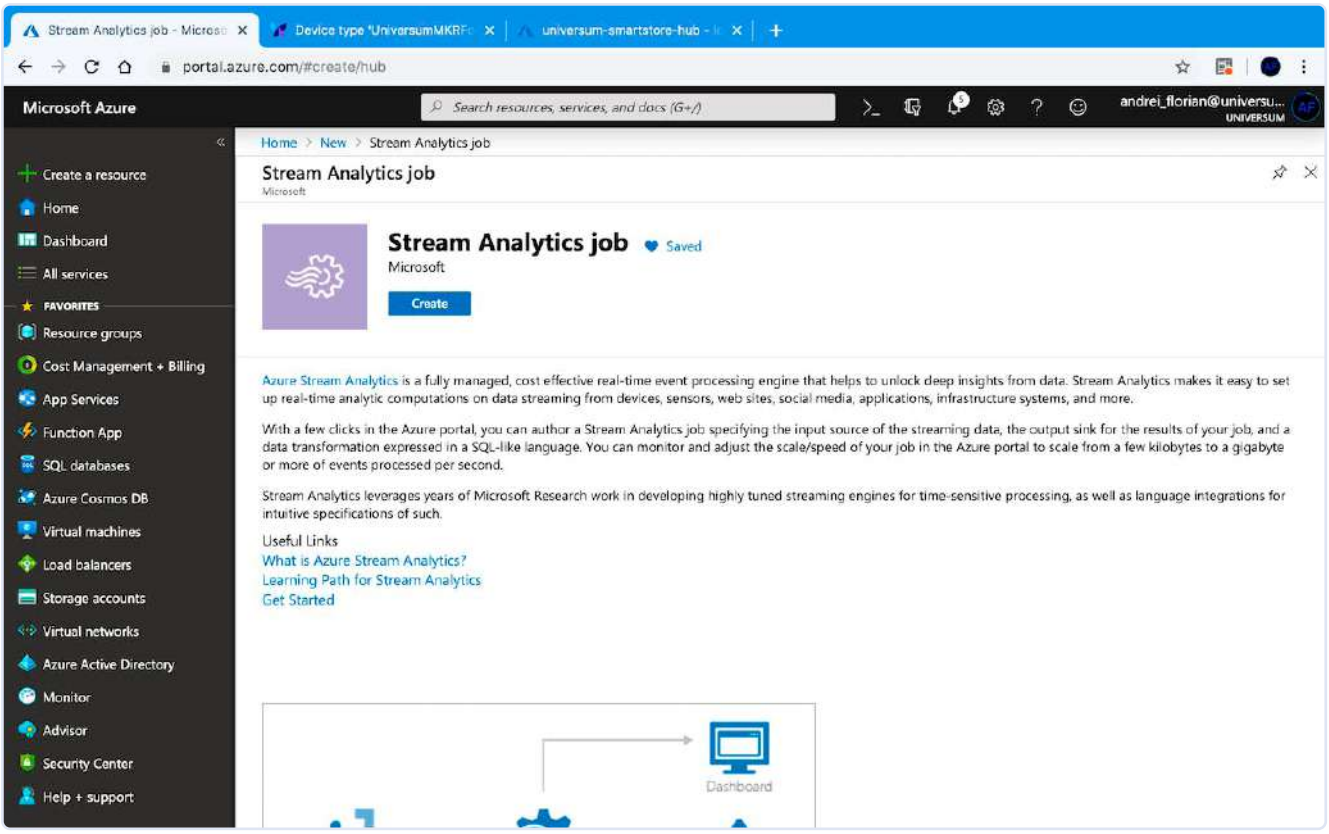


Figure 8: Configure the Stream Analytics job in Microsoft Azure.

Next, fill the JSON body of the message with the following data:

```
{
    "device" : "{device}",
    "data" : "{data}",
    "latitude" : {customData#geoLat},
    "longitude" : {customData#geoLng},
    "battery" : {customData#battery},
    "time" : {time},
    "duplicate" : {duplicate},
    "snr" : {snr},
    "station" : "{station}",
    "avgSignal" : {avgSnr},
    "lat" : {lat},
    "lng" : {lng},
    "rssi" : {rssi},
    "seqNumber" : {seqNumber}
}
```

This defines the values that we want to forward to Azure.

Stream Analytics Setup

The next step is to set up the Stream Analytics job (**Figure 8**) that will query data from the Azure IoT hub and push it into a Power Bi dataset. For this to work, you must first complete all the steps above, then refer to [1] for details on setting up this part.

In step 15 you must enter the following query:

```
SELECT
latitude as latitude,
longitude as longitude,
battery as battery,
System.Timestamp AS Timestamp
INTO
[OutputToPowerBI]
FROM
[InputFromIOTHub]
```

Upload the Code

Now we have to upload the code to test the connection. Ensure that the stream analytics job is running and then upload the code to the device. Simulate a chainsaw sound on your phone and make the device send an event to the cloud. You should see the graphs on your dashboard shifting after you sent the data. This indicates that the data is received, and you can move on.

If the event is not plotted on any of the graphs, you should start debugging with the Sigfox back-end and then work your way towards Azure.

Setting Up Microsoft Power BI

We will now set up Microsoft Power BI (**Figure 9**) to help us visualize our data. Note that a business account is needed to assign a Power BI subscription in Microsoft. Hopefully you have access to one. Otherwise, there are plenty of alternatives that you can use. The prerequisites are:

- Power BI account
- Completed all the previous steps

For the same reasons as for setting up the Azure account, please refer to [1] for setting up a Power BI account in fifteen detailed steps.

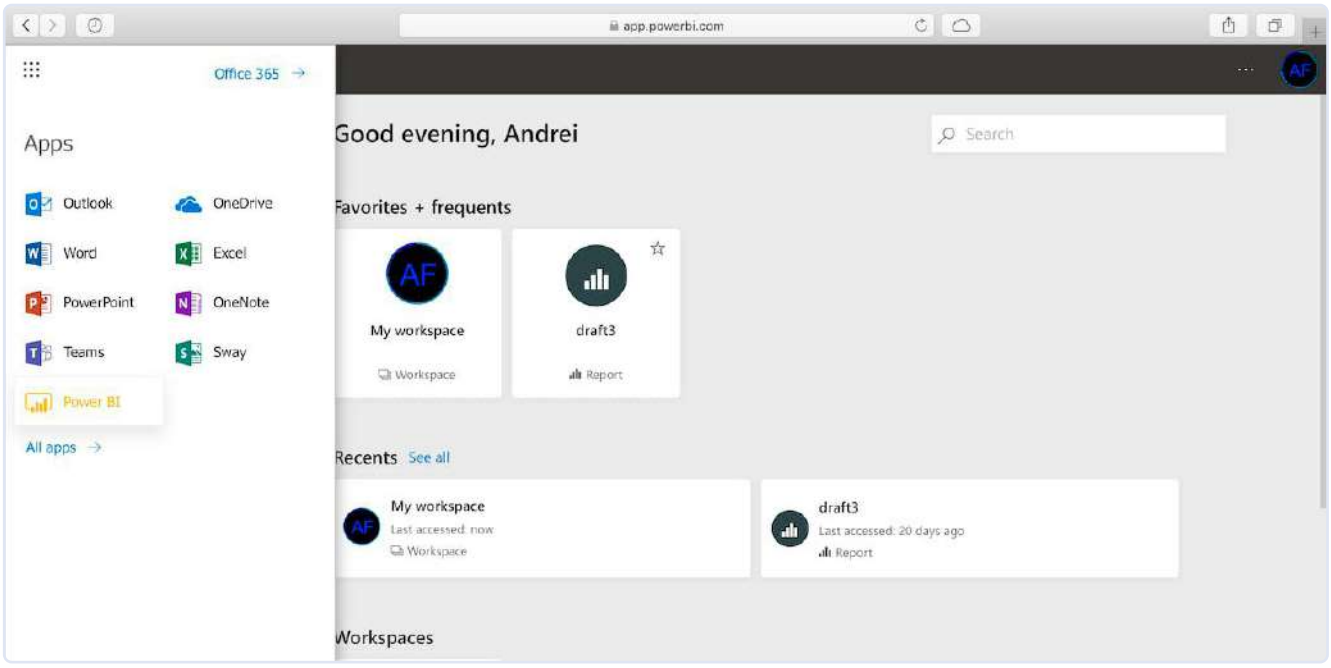


Figure 9: Navigate to Microsoft's Power BI in the Office Online app.

Enclosure

As a last step, after finishing setting up the application, we need to create an enclosure for the project (**Figure 10**). I decided to tie the device to a tree and camouflage it to catch the loggers by surprise. Make sure to place the microphone on the outside of the enclosure so it can pick up the sounds.

An IoT Solution

An array of Senso devices as described above hidden in the woods can look out for chainsaws and alert the authorities when trees are being cut down. I hope that countries will be able to stop illegal deforestation using IoT solutions. Senso is a first approach in finding cheap solutions to combat climate change one step at a time.

220423-01

Figure 10: The enclosure I came up with for my prototype.

About the Author

Andrei Florian is a second-level student based in Ireland who is passionate about inspiring positive change and global development through technology. He worked on multiple projects tackling various sustainable development goals specializing in Internet of Things and cryptography.

Questions or Comments?

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

Related Products

Looking for the main items mentioned in this article? Arduino and Elektor have you covered!

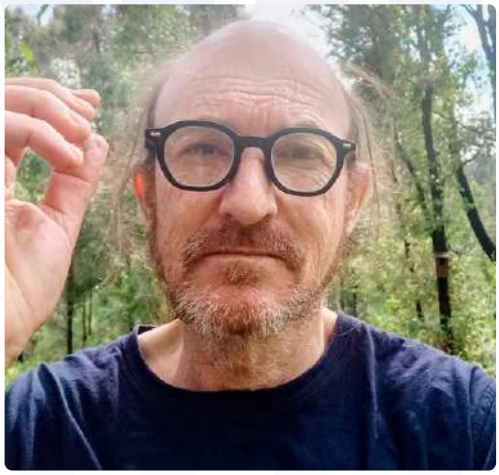
- **Arduino MKR FOX 1200 (SKU 19096)**
  www.elektor.com/19096
- **OPEN-SMART GPS – Serial GPS Module for Arduino (SKU 18733)**
  www.elektor.com/18733

WEB LINKS

[1] Senso at the Arduino Project Hub: https://create.arduino.cc/projecthub/andreiflorian/senso-c00153

The Mozzi Arduino Library for Sound Synthesis

Insights from Tim Barrass



▲ *Tim Barrass developed the Mozzi sound library for 8-bit Arduino boards.*

By David Cuartielles (Arduino)

The Mozzi sound library enables you to generate algorithmic music on an Arduino without additional shields or external synths. Curious how it all began? The developer, Tim Barrass, tells us about the project, the Mozzi community, and his work as an artist.

Tim Barrass is an independent artist living in the countryside in Australia who developed the Mozzi sound library for 8-bit Arduino boards (sensorium.github.io/Mozzi). In 2012, and together with his brother, Stephen, Tim developed a project called SweatSonics. The issues he found in the existing sound synthesis libraries got him to think about a better way to generate sound out of the early 8-bit processor architectures powering boards such as the original Arduino Uno, Nano, or Mega. The library's mode of operation was published in a paper a year later at the ICMC idea journal. It consists of a relatively small sound buffer that is called via software interrupts to produce a mono PCM sound output on one of the pins of the board. It exposes a couple of methods that you will have to override if you were to make your own music instrument or sound installation. One of the methods is dedicated to the reading of controls, the input to the system, while the other is meant to determine how those inputs will affect the rendering of the audio output.

I have been following the development of the Mozzi library over the years since I have always been very interested in the possibilities of microcontrollers for the production of sound. Tim has spent over 30 years performing as a musician and doing sound installations. He made Mozzi to support his work in the field of sound art. Mozzi is probably the longest-existing tool to easily create oscillators, drum machines, interactive sound art pieces, and innovative music instruments. While it first ran on 8-bit Arduino boards, it has lately been ported to other architectures. It is always nice to see how code can have a life on its own and be ported between different machines. In this interview you will get to know Tim, his work, the Mozzi project, and the community built around it.

David Cuartielles: After reading about the origin of the Mozzi library, I grew the suspicion that you must have a background in music and technology. How far am I from the truth? Can you briefly introduce yourself?

Tim Barrass: Yeah, I am an artist with a background in technology. I have been living in the country for a few years now mostly repairing fences and putting plants into the ground. I am not doing as much electronics any more, though I have some art exhibitions now and again. I play in a band, though we don't play very often. I worked for the last 30 years with a circus, a physical theatre. I played music for them. That fed into Mozzi, because the sound possibilities into the AVRs are comparable to the kind of sound I did for the circus.

Cuartielles: Do you have formal education in the field?

Barrass: I studied industrial design when I was young, a year of computer animation and music technology. I did some research in massive parallel processes such as how to program populations of ants, which was my master's research.

Cuartielles: I read this paper that you published with — I guess — your brother where you present Mozzi for the first time. You mention there that the work spun off from a project called Sweatsonics. Can you explain what this project is all about?

Barrass: That is something that my brother was pursuing. He was interested in how people were using sound while doing physical activities. That work was done in 2009 with iPods and accelerometers. It was a way for Steve to contextualize his work. Nothing serious came out from that, neither a company nor anything like that.

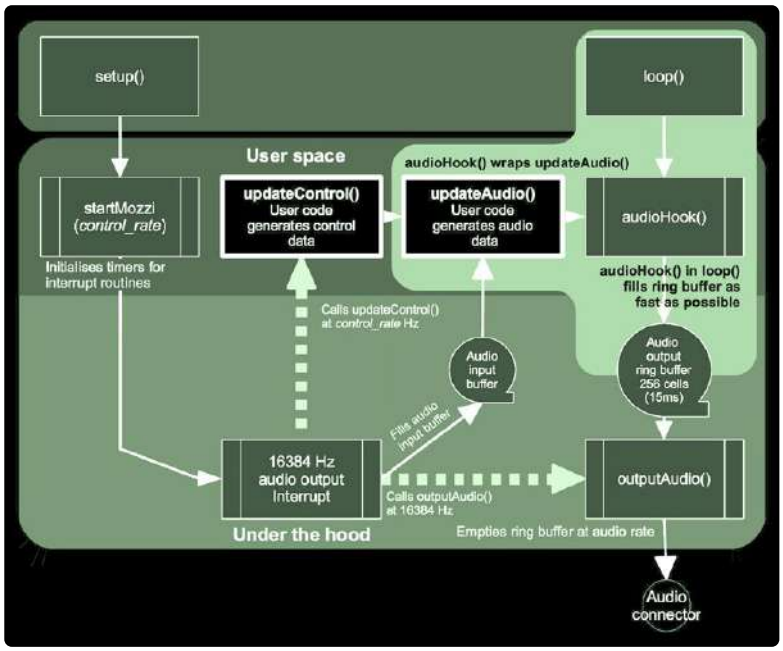
Cuartielles: Why Mozzi? Were you in need of a library to support your own development, were you trying to make it easier for others to create sound using microcontrollers?

Barrass: At the NIME conference in 2010, we did a couple of little objects using Arduino. We experimented with orientation and sound. We prepared some small objects which required some late night patching. It was like 5 o'clock in the morning and we were still programming. We were trying some code by Adrian Freed who did one of the earlier soundwave generations with an Arduino. As a result of that experience, I was frustrated, and felt like I needed to do something about it. My brother had made an attempt to do his own library, but he had lost the code. I started from scratch, becoming the first thing I had done with Arduino, and I must say that it was great fun. I was playing with it for a couple of years, maintaining this library, hopeful to be able to make some great stuff. I never made it to make the stuff, but I made the library.

Cuartielles: This gives a better perspective of the story. When reading the paper, I had gotten a different idea on how you came to make Mozzi. It is great to have the chance to talk to people to understand why and how things were done for real. So it sounds like you made this for yourself, but turned out that a lot of other people used it, right?

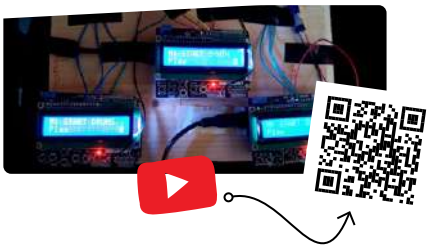
Barrass: I wanted to share it from the beginning. What's the point of writing all this code to just keep it to yourself? But then, it is quite a burden with releasing a library. People would tell me that I would be maintaining it forever and that I should just do my thing and move on. I had that in mind, and it did turn into that after a while. It was a lot of work, and I got tired after a couple of years. I learned C and C++ as I was making Mozzi. When putting this in public, I was getting stressed out by thinking that it was not good enough. After a while, I felt like I could not do it anymore.

Cuartielles: I have seen Mozzi all over the place. At universities and fairs. It keeps on coming back. If someone wants to make a sound-related project, there is a time when they are going to download and try Mozzi.



The interface between Mozzi and the Arduino environment consists of four main functions. (Source: https://sensorium.github.io/Mozzi)

Check out this video of the oscPocket0, an Arduino pocket synth and drum machine.



```
#include <MozziGuts.h>
#include <Oscil.h>
#include <tables/sin2048_int8.h>

#define CONTROL_RATE 128
Oscil <2048, AUDIO_RATE> aSin(SIN2048_DATA);

void setup(){
        aSin.setFreq(440);
        startMozzi(CONTROL_RATE);
}

void updateControl(){
}

int updateAudio(){
        return aSin.next();
}

void loop(){
        audioHook();
}
```

A minimal Mozzi sketch. (Source: https://sensorium.github.io/Mozzi)

Barrass: I am surprised myself because the framework is a very simple thing. More recently, in the last couple of years, really good programmers have ported it to other platforms. They're probably a lot more experienced than I am; it is quite intimidating. I am surprised that they haven't written it in a much clearer way right from scratch.

Cuartielles: I think that what works with Mozzi is the paradigm that you propose. To help our readers understand, you expose two methods to override called "updateControl" and "updateAudio." One deals with physical interfaces while the other one works with sound rendering. And that more or less does the magic. It produces mono sound signals. For many people — programmers or music instrument users — who are not necessarily luthiers and do not understand how to generate a sound output, which gives them great opportunities to create. I can recognize the beauty in the simplicity of this architecture. There are some highlights that I would like to ask about.

Barrass: There is a slow interrupt routine happening 64 times per second interfacing with your sensors, or MIDI signals, for you to set variables. There is also an audio rate interrupt where you can use those variables to affect the sounds in the synthesis phase of it. As a framework, I'd expected that people would extend it through classes. There are a number of synthesis classes, but no control ones. I did it so that others could extend things with their own code. In a sense, it is a bit like Forth, the first programming language that I learned; it has an input and an output and that's it.

The synthesis modules are made so that people could make their own. I made just some examples of ways of doing synthesis. I'd hoped that it'd grow, but it hasn't really done it in that way. There is room for it to grow and to make newer sounds. One of the early people working with it, Stenduino, in Poland made musical instruments pretty early — 2013 or 2014. They were interested in making their own sounds, not just changing the parameters on the sounds that were already there. I liked what they did with the design of the circuit boards and so on. Their work are now collectors' items and have become quite pricey.

Cuartielles: As I understand the original design was very constrained because of the design of the boards. You were pushing out sound at 16 kHz, on boards running at 16 MHz. But do you know how far people have been pushing it?

Barrass: There are people who have been producing boards that can work at higher rates. I have not been able to try them out, since I do not have the hardware. Then again, what surprises me is that they keep on using my code which was optimized to work in those original constraints and that was made in non-intuitive ways, with fixed-point arithmetic ... these new processors can do way better. And one of the guys told me that he liked the fact that he could write the code for one platform and still work on another. For me, the challenge in that case is to be expecting these small boards to make things that are designed for larger processors.

Cuartielles: At some point I was running a workshop where I designed a beatbox where you could record 1 second of your voice in the memory of the ATmega328 microprocessor (the one on the

original Arduino Uno) and use it as the basic set of sounds to make a looper or drum kit. Among the participants in the workshop, there were a couple of artists from France who had ported libpd (the sound engine of the PureData software) to run on the Arduino Uno, which blew my mind. I mean, I knew that people at the MTG in Barcelona (the masters in digital production at the Pompeu Fabra University) had ported PureData to run on 8-bit machines, but I had never seen it running on a very resource-constrained Arduino board. We have already spoken about what got you to make Mozzi, but were you aware of the existing alternatives and did those influence your decisions in how to create Mozzi in any way?

Barrass: I was not aware of the libpd port. There were few things we knew about which already had the concept of the interrupt timer and the wavetables. I didn't really look that hard, but I didn't find anything including the control interrupt that would allow you do anything interesting with the sound. There were just a few good attempts that were neither finished nor could they perform well. I don't think any of the examples I found had an audio buffer, and this is what made Mozzi more flexible in the end, a small amount of buffered audio that will let you do other stuff. This reminds me of an issue with Mozzi: some sensors were blocking the processor. There was someone writing some non-blocking code to work with sensors, and that is included within Mozzi now. I tried this with up to five different sensors. Some of them are hard to work with due to the small sound buffer.

Cuartielles: For the non-sound expert reader, I will have to explain that a sound buffer is a technique by which you prepare a certain amount of milliseconds of data that you will push out using a timed interrupt. This brings with it some latency, which is a time difference between the moment you send a command and until it affects the sound. In the case of Mozzi, it is 15 milliseconds. What is the perceived effect of this?

Barrass: It is like a chorus or a flanger delay. It is very unnoticeable, mostly hearable in percussion. But Mozzi was never really meant to create music instruments; it was more meant to create installations. I am surprised how many people created synths with knobs and cables and MIDI, and not-so-crazy sensors. Me and my brother are coming from a sonification background and were thinking about things such as a wind tunnel with a bunch of Arduinos that could alter the perception of sound. Making instruments was never that interesting to me / us.

Cuartielles: Do you have examples of these installations and sound production?

Barrass: There are some in the gallery page of the Mozzi library: sensorium.github.io/Mozzi/gallery/.

Cuartielles: I would like to know what you have in mind more than anything. This interview is about what inspires people to create things to help others following the spirit of open source. But I also have some funny questions. Let's start with the latter. Where is the term Mozzi coming from? I read that *mozzie* is the slang for mosquito in Australia and New Zealand. Is that so?

Barrass: At first, we were going to call it Project Cuttlefish, because one of these objects that we made was carved from a cuttlefish bone. And, at the last minute, we decided on Mozzi, because it (the animal) makes small sounds and it (the library) really was going to be used with little speakers, and nothing that was going to be amplified, or recorded. That really was suitable — a small sound. I am quite surprised the way it has gone. If you've got room (in your music instrument), why don't you use something that is much easier to use?

Cuartielles: There are some other platforms new or newer than Mozzi and that eventually already died such as Axolotl. It is (was) a very nice board with a piece of software that would allow you patch blocks to affect sound. Unfortunately, it was very expensive to produce and there seems to be no continuity to the project. There is something new called Bale that has a USB hub, and sensor inputs ...

Barrass: Just a couple of weeks ago I saw one called Daisy Seed.

Check out this video of the Turbulence Wind Sound Installation.

Cuartielles: Yeah, exactly. Since the arrival of Cube-X (an ancient sensor platform that could be programmed with Max/MSP), there have been platforms coming all the time. But there are situations where you just need a piece of code to add to your existing design, and there is where something like Mozzi can play a huge role. If the interrupts don't collide with the rest of the code, you can make it work. There is a difference with those dedicated, self-contained systems.

Barrass: I never thought Mozzi was easy to understand, and I am still not sure. I felt like I had to explain it a lot to people, for a long time. I am very pleased if it is easy for some people.

Cuartielles: Linking to this concept of explaining the library to others, I was also going to ask about dissemination. Your documentation page has 12 categories of examples. I read them in order, adding complexity as you go. Did you have a pedagogical intention when you made them, or were you just portraying the capabilities of the system?

Barrass: Both. I probably made the examples in the order I wrote the library. And, gradually, I moved towards more interesting ones.

Cuartielles: It is very typical in this world of technology and art, which is also linked to academia, to end up teaching workshops about the technology. Did you at some point run workshops using Mozzi with musicians and installation artists explaining about the potentials of the library?

Barrass: [laughs] My brother was working as an academic. This was very good for him to continue his trips around the world and teach a few workshops. But I didn't pick up with that. I would have liked to.

Cuartielles: What about collaborations and contributions? Looking at the Github repository I can see that there are some peaks in the production of the code, but how is it with the collaborations, adding other cores, PRs, issue management, etc.?

Barrass: About probably three years ago, Thomas Friedrichsmeier got in touch when he was porting things to his platform. Until then, we had the occasional bug report that I would try to keep up with. After a few months, I asked Thomas whether he wanted to be a collaborator, since I didn't even have his hardware or the time to look into it. I didn't feel like managing his codebase, and he started to take care of most of the development, which is mostly being porting, rather than growing the creative side of it. He is great.

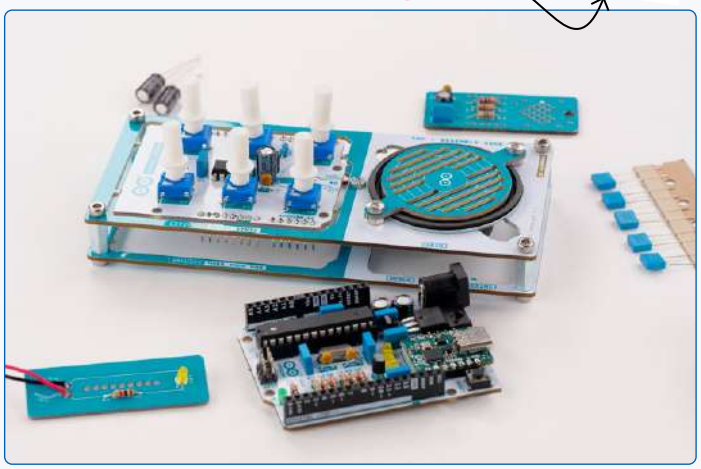
Another guy, Thomas Combriat, who came more recently, made a lot of contributions. The other Thomas seems to be glad to do the heavy lifting. Things are taking off. I like that the project has become autonomous from me. Same as in the forum where

The Arduino Uno DIY Kit

If you want to try out the Mozzi library, you could use something as small as an Arduino Uno and a simple piezo element connected to pin number nine on your board. You could also build a dedicated sound machine with the Arduino Uno DIY kit. You will have to assemble your own Arduino Uno and the accompanying synth shield including six potentiometers (five connected to analog inputs, and one to control the volume), an audio amplifier and a small — while loud — speaker. Depending on your level of soldering, it will take you between 60 and 90 minutes to assemble everything and start rocking. All single components in the kit are through hole. We have included a USB-C module to perform uploads to the main processor, power the board, and have serial communication back to your computer.

With the Arduino Uno DIY kit, you can play simple tunes using the default tone library in the IDE, or you can produce complex synth-sound by uploading the Mozzi library onto your board. Get started with the DIY Uno kit and Mozzi in three steps:

1) Assemble the Arduino Uno DIY + synth shield Kit (~60m).
2) Install the Mozzi library for Arduino in your IDE (~5m).
3) Try different examples.



people are helping each other. (Thomas Friedrichsmeier made the port of Mozzi for STM32, ESP processors, and the RP2040. Thomas Combriat, on the other hand, took care of the Teensy 4.)

Cuartielles: This leads to the yet unanswered question: What is your main interest these days?

Barrass: I moved here (Australia's countryside) thinking that I was going grow some vegetables, but the kangaroos and the wallabies keep on eating them. And then we had bushfires, which ruined our plans completely. We're still in the phase of recovering. Since then, my creative work is mostly towards my political punk protest band. It runs on the back of a trailer, and the band plays on while the trailer is moving. The band is really driven by the acrobats from the circus I've been a musician with for nearly three decades — I'm just supporting them, playing guitar for songs they're writing. We haven't played any proper gigs yet. I don't expect we'll get far! The songs are long and boring if you don't know the terrible politicians they are about. The wombats halfway through this video — https://youtu.be/E4qUPzUWxxA — are some I've looked after as a wildlife caregiver. One of my main activities now is to help a wildlife rescue organization raise and release animals after the devastating fires here in 2019-20.

220438-01

About the Author

David Cuartielles co-founded Arduino. He holds a PhD in Interaction Design and an MSc in Telecommunications Engineering, and he teaches at Malmo University.

Questions or Comments?

Do you have any questions or comments relating to this article? Feel free to contact Elektor at editor@elektor.com.

WEB LINKS

[1] T. Barrass, "Greenwash," 2013 : https://vimeo.com/69398645
[2] J. Deladrière, MozMo: Arduino Mozzi Synth in Eurorack Hardware, 2021: https://bit.ly/github-Mozmo
[3] A. Freed, "Arduino Sketch for High Frequency Precision Sine Wave Tone Sound Synthesis," 2009: https://bit.ly/freed-sketch
[4] Y. Nakanishi, Powderbox, The University of Tokyo: https://yoshihito-nakanishi.com/projects/powderbox/

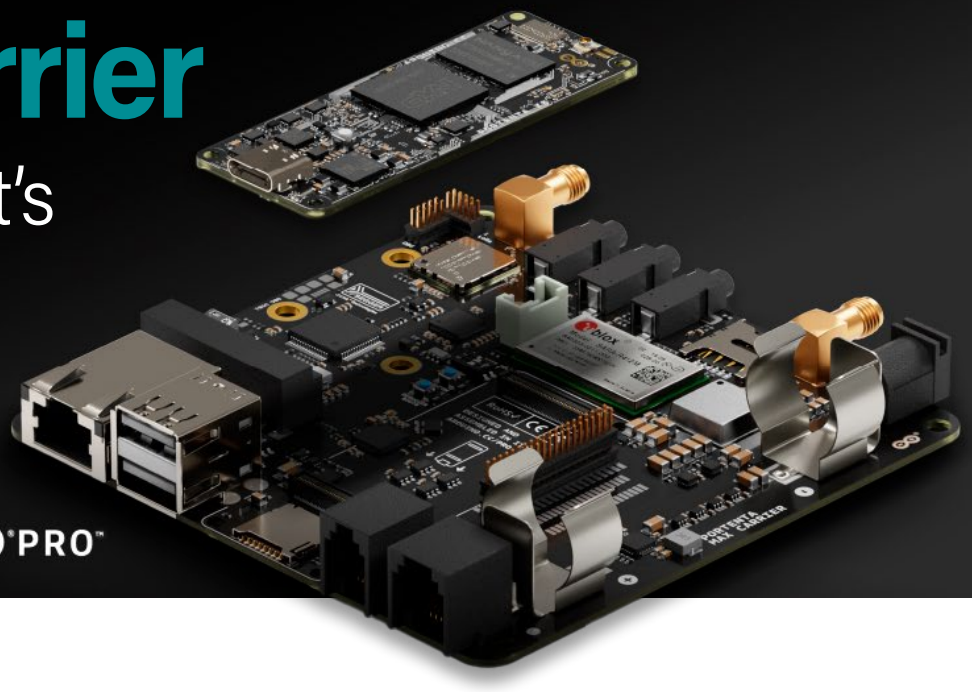
Related Products

Looking for the main products mentioned in this article? Arduino and Elektor have you covered!

- **Arduino Uno Rev3**
  www.elektormagazine.com/arduino-uno
- **Arduino Uno DIY + Synth Shield Kit**
  www.elektormagazine.com/arduino-synth-shield-kit

The New Portenta X8 (with Linux!) and Max Carrier

Redefine What's Possible



By Stefano Implicito (Arduino)

Arduino's mission has always been to give creators, makers and innovators the tools they need to turn their ideas into real projects. That will never change. What is changing, and at warp speed, is our definition of "possible".

Every day, Arduino's accessible, flexible and reliable open-source hardware grows more powerful, and our ecosystem more complete. As you may have heard recently, Arduino Pro has launched two new products in the Portenta range: the revolutionary X8 board, which merges Arduino and Linux for the first time, and Max Carrier, which gives you prototyping super powers to make your ideas come to life easier and faster than ever.

Portenta X8

Portenta X8 is a plug-and-play, industrial-grade SOM that comes with Linux OS preloaded onboard, making for a hybrid combination of microprocessor and microcontroller with the capability for AI and ML on the edge (**Figure 1**). It's basically two products in one, with the power of no fewer than nine cores. It features an NXP i.MX 8M Mini Cortex-A53 quad-core, up to 1.8 GHz per core + 1x Cortex-M4 up to 400 MHz, plus the STM32H747XI dual-core Cortex-M7 up to 480 MHz +M4 32-bit Arm MCU up to 240 MHz, and is capable of running device-independent software thanks to its modular container architecture.

With onboard Wi-Fi/Bluetooth connectivity, you can carry out OS/application updates remotely, so the Linux kernel environment is always at top performance levels. Enhanced security is guaranteed by X8's NXP SE050C2 Crypto element — keeping connections secure at the hardware level with PSA certification. The module has also achieved Arm System-Ready certification and integrated Parsec services, making it one of the first Cassini Products available to you on the market.

It's everything you need to develop your most ambitious projects for Industry 4.0, smart agriculture, connected buildings and smart cities: check out the full technical specs on the dedicated page [1] and empower your Linux applications with real-time execution.

Watch "Introducing the Portenta X8 by Arduino Pro"

Max Carrier

If that wasn't enough, Arduino Pro also introduced Portenta Max Carrier (**Figure 2**) — which boosts X8 or H7 by adding connectivity options (Fieldbus, LoRa, Cat-M1 and NB-IoT), industrial connectors such as RS232/422/485, USB, mPCIe — plus three integrated audio jacks, a MicroSD card and more. [2]

You can combine this Arduino Pro carrier with existing Portenta modules to turn them into single-board computers or reference designs for Industry 4.0 — bringing your deployment time virtually down to zero. This allows you to swiftly prototype and develop high-performance projects such as remote control of industrial machinery and equipment, smart digital kiosks that enhance users' experience, and custom HMI dashboards to control smart appliances, lights and systems in your home or office remotely.

As excited as we are about these launches, it's never just about a new product (or two!). It's about how many new ideas we can spark, the innovation we can fuel, and the new opportunities you can create, if you have the right tools.

220377-01

Portenta X8 is a plug-and-play, industrial-grade SOM that comes with Linux OS preloaded onboard.

Watch "Portenta Max Carrier From Arduino Pro"
https://youtu.be/GxN2-hs9288

Related Products

Looking for the main items mentioned in this article? Arduino and Elektor have you covered!

- **Arduino Portenta Max Carrier**
  www.elektormagazine.com/arduino-portenta-max-carrier
- **Arduino Portenta X8**
  www.elektormagazine.com/arduino-portenta-x8

About the Author

Stefano Implicito pairs a deep passion for technology with a natural ability to connect with people that transcends communication channels, countries and cultures. After over a decade of experience in customer relationship management in the high-tech B2B industry, as Product Marketing Manager at Arduino he successfully launched more than a dozen new products for the quickly-growing Arduino Pro business unit. He sees extending the company's successful open-source model – from the maker movement and educational settings to full-fledged industrial applications – as a way to generate innovation and opportunities for the entire community, shifting towards a world where technology is truly accessible to everyone.

Questions or Comments?

If you have technical questions feel free to e-mail the author at pro@arduino.cc or the Elektor editorial team at editor@elektor.com.

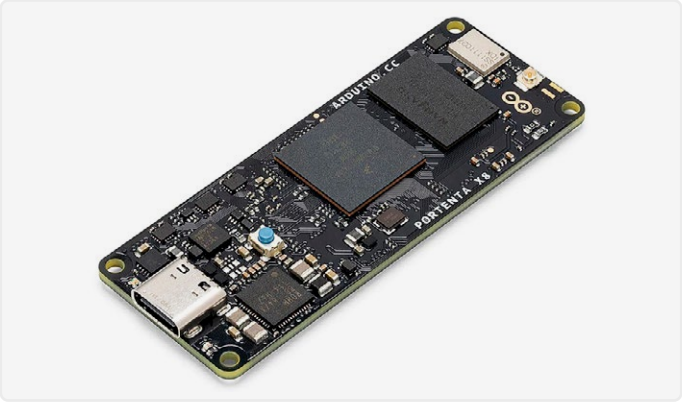


Figure 1: Portenta X8 is a powerful, industrial-grade SOM with Linux OS preloaded onboard, capable of running device-independent software thanks to its modular container architecture.

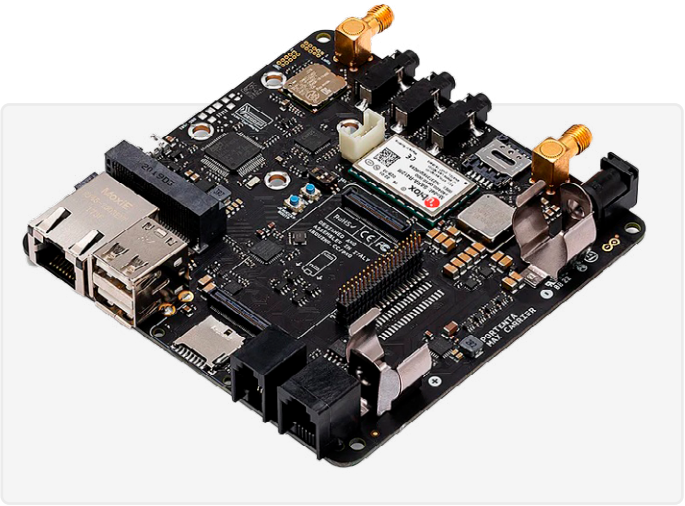


Figure 2: Max Carrier transforms Portenta modules into single-board computers or reference designs that enable edge AI for high-performance industrial, building automation, and robotics applications.

WEB LINKS

[1] Arduino Portenta X8: https://www.arduino.cc/pro/hardware/product/portenta-x8
[2] Arduino Portenta Max Carrier: https://www.arduino.cc/pro/hardware/product/portenta-max-carrier

How **Using Arduino** Helps Students **Build Future Skills**

CTC GO! is a comprehensive, hands-on learning program that guides students through the fundamental concepts of electronics and coding, and challenges them to assemble, wire, and program tabletop games.

By Keith Jackson (Arduino)

Arduino Education develops innovative STEAM programs that help students on their learning journey from middle school through university. The programs include a range of electronics, open-source software, online content, and guided training and support for educators.

Most Requested Skills by Employers

- Critical thinking and problem solving
- Creativity
- Teamwork
- Communication
- Digital literacy
- Adaptability

Future skills? What are they? Well, they're any "soft" skill that will be of use to current students in their future careers. Think of things like problem-solving, critical thinking, creativity, and collaboration — skills that you've probably naturally honed through your passion for tinkering with electronics, programming, or coding.

These types of skills have to be deliberately developed in schools (and at home), especially in our digital age, to extend students' skill sets beyond the standard curriculum and enhance their attractiveness to future employers. And using Arduino products supports educators in doing just that. How?

Arduino Is More Than Electronics and Coding

When you use Arduino products or kits, you're essentially creating something from scratch. You need to get creative with the build, creative with the code, figure things out when something goes wrong, research different ways of doing things, try something new, ask a buddy. This might seem like a natural process to you, but with each one of these steps, you're using a soft skill — and those skills don't come as naturally at a young age.

The same process applies to the classroom environment — or at home if you're using Arduino with your children. But in the classroom or science lab, you can use purpose-designed, educational Arduino kits that not only teach electronics, programming, coding, or engineering, but also focus on ensuring that students learn these soft skills.

The Arduino Effect at Penn State University

Pennsylvania State University ranks as one of the best research universities in the world. It's one of the thousands of institutions worldwide that was affected by the coronavirus pandemic, and Dr. Herschel Pangborn

and Dr. Pansy Leung from the Mechanical Engineering department found new and innovative ways to teach remotely using the Arduino Student Kit.

While Herschel admits his junior engineering students were amused that he recommended a kit geared towards 11 to 13 year olds, once they discovered its open-ended potential, they used it across projects on their course while learning at home. And not only did it help "actually understand how circuits work in real life," according to one student, it also provided a way for the department to teach students about problem-solving.

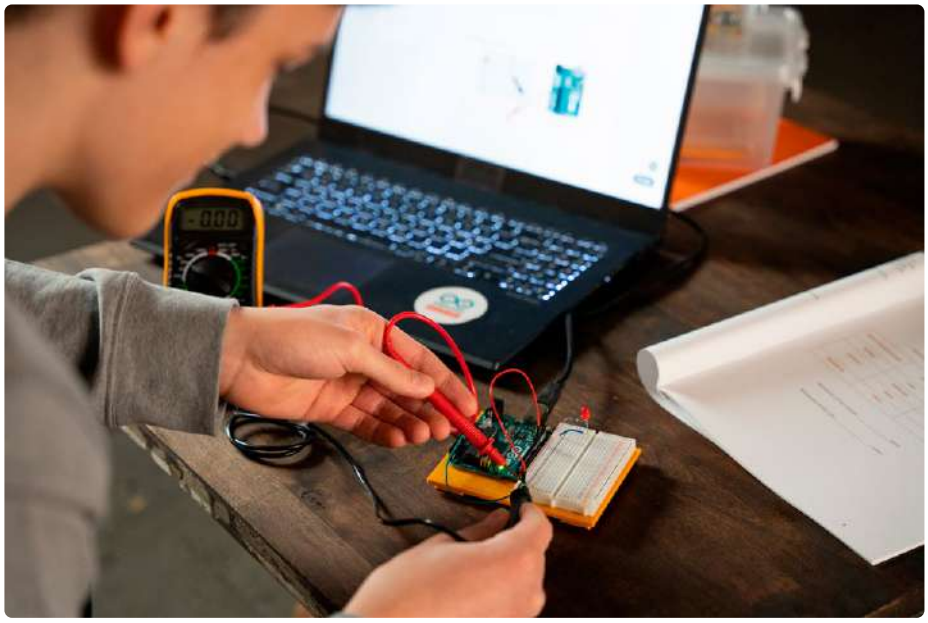
With the help of built-in sensors students can experience and play with their surroundings in an easy and hands-on way.

Pansy, who wants to transform her lab into a problem-solving process, says, "Students will apply what they learned with Arduino at junior level to solve real problems, instead of just getting an understanding of the theory behind problem-solving."

Getting hands-on is crucial to learning how to solve problems. It gets you invested in and accountable for solving the problem, sparks creativity in finding solutions, and pushes you to find innovative answers.

The Arduino Student Kit

Kids can start with the basics of electronics, programming, and coding with the Arduino Student Kit. No prior knowledge is necessary. The kit guides you through 11 activities, introducing concepts such as current, voltage, and resistance. Each kit includes hardware, access to online learning content, and dedicated support, making it ideal for remote teaching, home schooling, and self-learning. More info: arduino.cc/education/student-kit



The Arduino Student Kit

The Importance of Experimenting & Hands-On Learning

As the students at Penn State University found out, hands-on learning is crucial to understanding a concept and being able to break it down into smaller parts. This has always been true (remember Confucius' quote: "I hear and I forget, I see and I remember, I do and I understand"), which rather begs the question why anyone thought it was a good idea to have children sitting in rows learning from textbooks.

And getting hands-on not only helps you understand a concept. As students, you're 1.5 times more likely to fail a course when you don't engage in hands-on learning.[1] This type of learning also engages both sides of your brain, and scans also show increased activity in sensory and motor-related areas of the brain when you think about concepts you had hands-on experience with.

Science Journal App

Dr. Bates uses the Science Journal App, which allows you to gather data about the world around you by harnessing the sensors in your smartphone as well as sensors connected to Arduino.
More info:
arduino.cc/education/science-journal

Noah Kemp was the first student at Parklands College, South Africa, to get his Arduino skills certified!

Dr. Alan Bates, a physics teacher at Haileybury College in the UK, says, "What I like about Arduino is that you can use it for traditional experiments, but also to reinterpret those experiments. You can get students to design their own instruments for experiments, even, and that gives them deeper insight. And in doing that, students are gaining so many skills — I call them 'real-life skills.' You can teach students to be more self-driven as there isn't a prescribed answer — I don't even know the answer — they have to explore, fail, and try new possibilities, as it is in the real world."

Getting Your Skills Certified

Many schools recognise the importance of preparing students for the real world and a successful future by focusing on developing soft skills. This is the case at Parklands College in Cape Town, South Africa, where the mission statement is "Reaching Outwards, Growing Minds, Building Futures."

While they were using Arduino products in their STEM lessons and across class projects, they discovered Arduino Certification, which certifies your Arduino knowledge in the field of programming and electronics.

Arduino Certification

Arduino Certification is an online exam that provides official certification on your knowledge of Arduino-related electronics, programming, and physical computing. It is open to everyone aged 16 years and above. More info: arduino.cc/education/certification

Director of Technology Innovation, Richard Knaggs, said, "We're very excited because it really ties into our vision. And our vision is to make sure that we give our learners authentic opportunities to find their purpose so that they can start building their futures while they're still at school."

Certification is a useful string to students' bows, as it shows future employers that you not only have technical skills, but also the future skills they need their employees to have to succeed as a business, like problem solving and critical thinking. (And Arduino Certification is open to everyone!)

We know that using Arduino helps build future skills — and it's something we're really proud of. After all, the next generation is going to need problem-solving, critical thinking, and creative skills to help come up with solutions for many of the world's challenges.

220452-01

About the Author

Keith Jackson is a marketing consultant at Arduino with more than 25 years of experience working for a global businesses in the technology and electronics industry.

Related Products

Looking for the main products mentioned in this article? Arduino and Elektor have you covered!

- **Arduino Student Kit**
  www.elektormagazine.com/arduino-student-kit

WEB LINKS

[1] J. Arnholz, "Is Hands-On Learning Better?," BYF.org, February 12, 2019: https://byf.org/is-hands-on-learning-better/

MUST-HAVES for Your Electronics Workspace

What makes a great electronics workspace? Which tools top engineers use at their workbenches? Our friends at Arduino share details about their workspaces, and they offer some helpful tips for anyone interested setting up their own.

Arturo Guadalupi
(Manufacturing Test)

Optimize Your Space

I use my workspace for making custom audio equipment and as a home office. I have four separate areas for: working at the PC, storing components, doing rework, and making measurements or 3D printing something. One of the walls of the room is used to showcase my musical instruments collection while another one is for storing components.

Advice: Start step by step buying quality tools! Don't try to have all the things at once. Start with all the essentials and then expand or improve your lab when a new necessity arises. Try to optimize space using well-done organizers and store things in a way that they can be easily reached when needed instead of repeatedly searching in disorganized boxes.

Current Project: 50/30W tube amplifier with four channels and MIDI/BLE control.

Essentials

- Bench oscilloscope
- Bench power Supply
- A very good soldering iron
- Bench multimeter
- Bench signal generator
- Hot air desoldering station

Wishlist

- THD meter
- Spectrum analyzer
- Electronic load

Andrea Masi
(DevOps/SRE)

Every Tool Has Its Place

My 200 m² lab is an open space which serves multiple activities: woodworking, metal working, electronic tinkering, gardening and (soon) brewing.

Advice: I try to follow these rules: every tool must have its own resting space; no tools on desks (only components to repair or assemble); and the "rule of 3" before buying/searching for a tool (find yourself in need of that tool for real tasks at least three times). Safety and comfort: good lighting and air circulation systems are paramount and make even heavy activities like grinding and welding more enjoyable, safer and better executed.

Current Project: Prototyping an evaporation cooler with no nasty chemicals (e.g., no ammonia). It involves metal working, some chemistry, and electronics (control board for circulation pumps, humidity/temperature sensors reading and control algorithm execution). You know Arduino, don't you?

Essentials:

- Good multimeter
- An at least 100-MHz oscilloscope
- Good caliper and micrometer
- Soldering iron with hot air gun to work
- Transistor tester (Atlas DCA)
- Binocular magnifier headset
- Clamps and vises
- Circular and miter saws
- Laser cutter/engraver
- Industrial vacuum cleaner

Wishlist:

- Simul-focal stereo microscope
- A better welding machine
- 700 mm lathe
- Milling machine

Martino Facchin
(Sr. Firmware Engineer)

Don't Underspend on a Soldering Station

My workspace is primarily in the office due to the sheer amount of hardware I usually deal with. It may look messy, but everything is in its right place. I always keep a beefy oscilloscope (with protocols decoder, very important) ready for action on my right, and a single 24" monitor to avoid moving my head to see something. Since I need to connect everything via USB my setup is literally plenty of ports (no less than 12).

Advice: As a firmware engineer, it's very important to be able to patch a prototype in the fastest and cleanest way possible, so my advice is not to underspend on the soldering (rework) station. An oscilloscope is a must, but if you are on a budget, you can get good results with a PicoScope or a signal analyzer (Saleae or similar). A beefy main PC is also important, especially if you work on heavy stuff (Linux development/FPGAs) or if the tool you need to run only has a Windows version and you need to spin a VM.

Current Project: Super-secret stuff I can't really talk about.

Essentials

- Very good solder station with (de)soldering tweezers. (JBC, that stuff is amazing.)
- Oscilloscope
- Hot air rework station
- Powerful desktop PC (min eight cores/32-GB RAM/ good SSD/Linux). Self-assembled towers with Ryzen CPUs are very cheap.
- FTDI cable for reliable Serial-to-USB conversion
- JLink debugger (so stable)
- USB Protocol analyzer (Beagle 480). Not essential at all, but one of my favorite toys. Get it only if you need to debug arcane USB bugs more often than once a year.
- HDMI-to-USB converter (If you work with video signals, it spares you from getting a separate monitor.)

Wishlist

- Hall effect current probe for the oscilloscope
- An even bigger and more powerful PC. (Yocto builds shouldn't take hours.)
- An end to the chip shortage

Ubi de Feo
(Creative Tech Lead)

Quality Tools Make a Difference

Strange as it may sound, these days I don't use my electronics workbench a lot anymore. Being mostly busy with software (heard of the Arduino IDE and CLI?), I had to find another physical outlet. My workbench, as of a couple of years ago, is in a woodworking space, and my tools are track saws, drills and clamps, but I guess that's not of high interest, so here's my digital electronics workbench for you to nerd on. Note: the bench has been designed and built by me as one of my first furniture making projects.

Advice: I'm OCD at a pretty high degree, and I plan for months before I even start on anything. In the beginning I got away with cheap tools, but pretty soon I realized that if you want to be efficient and precise, you need better tools. Learn how to use a scope and a logic analyzer; they'll eventually save the day. Always clean up before leaving; you'll thank your past self the morning after. You need as much as empty surface as you can, because it'll get messy pretty quickly.

Essentials:

- Rigol DS1204
- Fluke 175
- Omnifixo
- Ersa i-con 2
- Saleae Logic Analyzer
- Segger J-Link
- Ultimaker 2
- Shapeoko 3 XL

Wishlist:

- I swear I don't know what else I need for my electronics bench. I bought everything. For the woodshop, trust me, a lot more money is bound to be spent.

David Cuartielles
(Co-Founder)

Identify Your Real Needs

My workspace is a container sitting in my garden including a full small-scale electronics design lab, a streaming system with front and top-view cameras, an archive of Arduino boards and derivatives (I have over 200 different models from all over the world), and a bike repair-shop. I have a 4m long shelf with components, a microscope, oscilloscope, oven, soldering station, and plenty of small parts that I collected during my trips to street electronic markets in Seoul and Mexico. On top of that, I have a NAS where I store files coming from recordings, manuals to parts, etc.

Advice: Identify your real needs, for example, I did not buy my microscope until I realized that more and more projects had gone down to 0402 sizes. Keep a list of the things you buy in a digital format and keep track of things you borrow. Keep the relevant tools at hand and make sure you have a list of good-to-have tools. Monitor second-hand markets periodically, i.e. I follow different auction houses where to find parts for different tools. This is how I recently got a 26.5GHz spectrum analyzer for almost nothing.

Current Project: Materials for my forthcoming course "introduction to embodied interaction" using the Nano 33 BLE Sense: breadboards, jumper wires, and some simple sensors.

Essentials

- Gaming computer with graphics card for video editing and two 27-inch screens
- Oscilloscope
- Weller soldering station (I am a big fan)
- 60x microscope
- 26.5 GHz Spectrum analyzer

Wishlist

- Automated light and heating control
- Solar panel and UPS
- LoRa gateway
- Nice chair

Giulio Pilotto
(Software Engineer)

Versatility Matters

My workplace is essentially a desk in my apartment, but it is very versatile. Since I used to work with other people in the nearest FabLab. My desk needs to be portable, so all the instruments are stored in different well-organized toolboxes. I have categorized them based on the use: heavy, medium and soft. In the heavy one, I have all the tools related to mechanical work, like a drill, oil to take out old screws, screwdrivers, and other tools. On the medium, I have everything related to mechanical work for a small-scale application, from a little electric screwdriver up to all the tiny tools used to disassemble a smartphone. Then I have the soft toolbox, which is basically everything about electronics, design and making. There you can find Arduino boards, electronics components , sensors and actuators, glue of different strengths. This is an example on how I use all my three boxes on a single project: www.instructables.com/Jungle-Reef-Bluetooth/.

Advice: First select at least 10 projects that you would like to work on, and from there, draw a list of your tools. Start to buy the most common tools in the list. Check on the web for a good price/performance tool. YouTube is full of tool reviews (e.g., "Top 10 PARKSIDE Tools"). Even a cheap brand is good to start. Take a clear picture of your empty workspace, print many copies of it on a white paper, and start to draw how you would like to place your tools! Remember that the style needs to fit usability. Some good examples: https://www.instructables.com/howto/workspace/.

Current Project: An Arduino with a 433Mhz module to decode a PIR sensor (https://github.com/giulio93/RevEng_433Mhz).

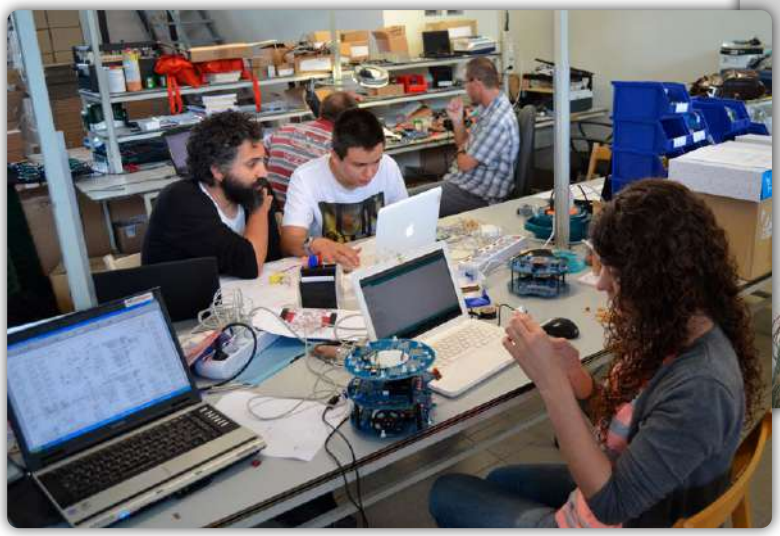
Essentials

- Multimeter
- Microscope
- Basic soldering station
- Dremel

Wishlist:

- Oscilloscope
- A Weller soldering station
- A good LED lamp with lens

220521-01



Nerea Iriepa

◄ Robotics in education. (Source: Nerea Iriepa)

The **Importance** of **Robotics** in **Education**

By Keith Jackson (Arduino)

Using robotics in education, although not well integrated (yet), can be transformative. It's critical to learning about STEM subjects (science, technology, engineering, and math), it's engaging, and it prepares students for the future.

Just think about where robots are used in the real world: medicine, elderly care, home appliances, and cars. The list goes on. Really, how can anyone think that teaching robotics in schools is not important?

But, it's starting to gain momentum, in no small part due to the many examples of people who've innovated, created, and solved real-life challenges because robotics was included as part of their education. And, we have three examples of our own.

World Robotics Champion

Meet Nerea Iriepa, a world robotics champion and Arduino Education sales lead. Back in 2003, Nerea's teacher brought LEGO Mindstorms robots to their after-school club and a passion was born. The club held a small robotics tournament at school before aiming higher – RoboCupJunior, an international tournament based on football. While the team came 26th out of 27, Nerea was hooked.

In 2008, Nerea discovered Arduino, and, after years of challenges with building a robot that was fit for a RoboCupJunior tournament, she found both a community whom she could ask for advice and much simpler components. And that was the year Nerea's team won – or first won, as they retained their crown for three years, including a record victory of 52–0 against Mexico.

One of the best things about this? Being able to give advice to other teams, in turn. Helped along with Arduino's open-source approach, nothing was a secret, and everyone could learn from each other.

As RoboCupJunior is a tournament for school students, retirement came all too soon. But, what happened next was amazing: Nerea met

with two founders of Arduino and showed them her team's winning robot. Suddenly, people wanted to buy it for their classrooms to teach robotics to their own students.

After two years of development, the Arduino robot was ready, and, once Nerea completed her university studies, she had a job with Arduino Education.

Nerea met with two founders of Arduino and showed them her team's winning robot. Suddenly, people wanted to buy it for their classrooms to teach robotics to their own students.

Do the Robot

The World Robot Olympiad (WRO) was established in 2004, and it is now in over 85 countries across the globe. It's a global robotics competition for young people aged eight to 19, and, in the last typical year, which was 2019, 29,000 teams participated.

Claus D. Christensen, Secretary General at WRO Association Ltd., noted: "We believe that hands-on experience, exploration, and play are the best teachers of all, and our vision is to pursue a future where every curious young person, regardless of their background, is inspired and equipped to achieve their full potential through science, engineering and technology."

This is also the heart of Arduino Education, which is why we're proud to be a WRO sponsor and support the great work WRO does in creating opportunities for young people to get involved in robotics and STEM.

World Robot Olympiad (WRO)

A School Brings Robotics to Early Childhood

In Copenhagen, a small Danish/French school has a big ambition: to ensure that every student, from age three upwards, has experience of robotics (using Arduino, of course!). Nicolas Guilbert, the school's founder and STEM teacher, says: "We chose robotics because it works. We actually do it in kindergarten with very small kids. We use a robot and have them type in simple commands – it's a very simple system, but they can make robots do things such as go forward three steps and then turn left. And they find it super fun to control the robot and make it drive around."

He goes on to say, "It's very easy to engage kids with robotics. And, robots are very useful. I mean, a dishwasher is a robot, and the kids get hands-on experience with that. Robotics also leads into a world that's asking for engineers and scientists, so you'll have a job for sure if you have those skills."

And, why does he use Arduino in his school? "It just works. You download the IDE (the development environment), you plug the UNO into your computer, upload a sketch, and it just works. And it also works quickly. You get results right away, and that's really important with children. There's a low barrier to entry, and a quick win really kickstarts their engagement."

Find out more about Arduino Education at arduino.cc/education.

220455-01

About the Author

Keith Jackson works in marketing for Arduino and is passionate about all things Arduino as it's more than a company or brand, it's a whole diverse community.

Related Products

Looking for the main products mentioned in this article? Arduino and Elektor have you covered!

- **Arduino Braccio ++ RP2040 powered Robot Arm**
  www.elektormagazine.com/arduino-braccio
- **Arduino UNO Rev3**
  www.elektormagazine.com/arduino-uno

Dependable IoT Based Upon LoRa

By Stuart Cording (Elektor)

For its Pro range of products, Arduino has selected LoRa and LoRaWAN as a focus area to support LPWAN developers. Massimo Sacchi, Corporate Partnerships Manager and Business Developer at Arduino, talks about LoRa and Arduino's related hardware and software solutions.

Wireless has typically been a double-edged sword for electronics engineers. On the one hand, it breaks the tether to enable mobility, supporting innovative use cases across all industries. On the other, it required experience in RF, had more regulations to fulfill, and was power hungry. So, for those relying on battery power in their application, it has sometimes been challenging to deliver the innovations specific markets require. However, ongoing development in semiconductors, especially in RF on CMOS technology, means that radio transceivers have been getting smaller and easier to integrate. Over the years, a wealth of modules and system-on-chip solutions with integrated microcontrollers have hit the market. Thanks to efforts to make better use of unlicensed spectrum in ISM bands (industrial, scientific, and medical) coupled with cloud services, low-power wide-area networks (LPWAN) have grown steadily.

This hasn't gone unnoticed at Arduino, who, through their portfolio of extensible boards and simple programming approach, have established support for almost all wireless technologies. With the creation of their Arduino Pro range for industry users, launched at CES in 2020,

Massimo Sacchi

Massimo Sacchi is the Corporate Partnerships Manager and Business Developer at Arduino. He joined the team in 2019, four years after moving to Australia. He grew up and studied in Italy, where he attained a degree in Electrical Engineering before spending 20 years working in the field of industrial automation. Massimo is passionate about the IIoT and cloud applications, supporting the growth of low-power wide area networks as vice-chairman of the Australian and New Zealand LoRa Alliance Task Force.

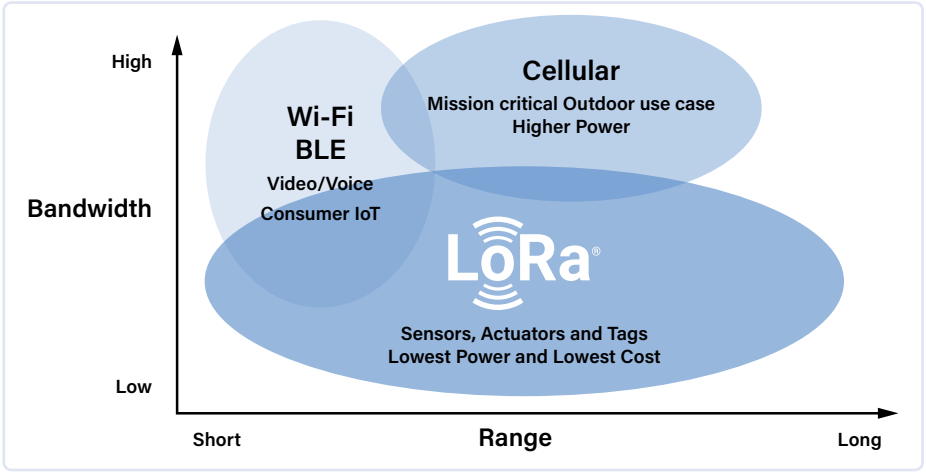


Figure 1: LoRa offers both short and long range connectivity for applications that only handle small amounts of data. (Source: Arduino)

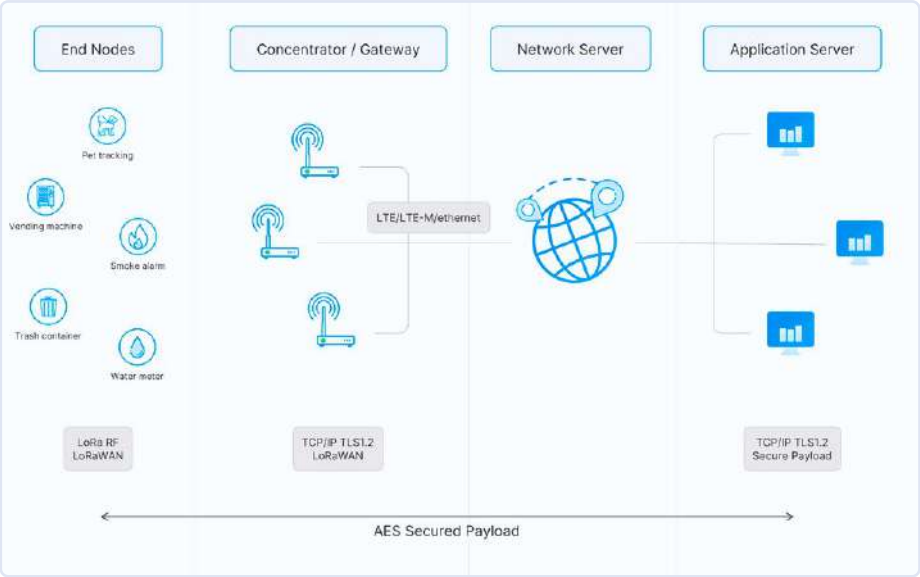


Figure 2: LoRaWAN is what enables LoRa to form a network, connect to cloud services, and operate securely. (Source: Arduino)

LoRa and LoRaWAN have been selected as a focus area to support LPWAN developers. Massimo Sacchi, Corporate Partnerships Manager and Business Developer at Arduino, is well placed to support this effort since he is also vice chairman of the LoRa Alliance Task Force in Australia and New Zealand.

"I love to explore new technologies, especially those in the IIoT," says Massimo. "It's only natural, therefore, that we at Arduino use our capabilities to innovate in the space of LoRaWAN."

What Is LoRaWAN?

There is plenty of choice for those looking to implement wireless networks for internet of things applications (IoT) or industrial IoT (IIoT). Cellular, such as 4G/5G, provides a functional infrastructure, allowing developers to concentrate on their applications. However, despite the rise in lower-power solutions such as NB-IoT [1], the promised 10-year battery life is not a given.[2] Wi-Fi can also be considered ubiquitous, but the range is limited and, prior to Wi-Fi 6, the power optimizations needed for IoT node power consumption weren't available. Furthermore, nodes are limited to operation in range of the router and any repeaters due to the lack of handover support.

LPWAN networks focus on the needs of the majority of IoT applications: small packets of infrequent data, long range, and very low power consumption. LoRa is one of the most successful of these technologies (**Figure 1**), a radio technology using a wireless modulation technique known as Chirp Spread Spectrum (CSS) that provides its robustness in the field. Operating in the sub-gigahertz bands of 433 MHz, 868 MHz, and 915 MHz in the spectrum reserved for ISM use, one or more of these frequencies are available almost everywhere around the world.

To turn LoRa-capable nodes into a network, the LoRaWAN software layer is added (**Figure 2**), an open-source specification supported and maintained by the LoRa Alliance. Thus, LoRaWAN end nodes where applications are implemented, can communicate with gateways connected to the Internet. From here, bi-directional communication is established with cloud-based application servers that can process the data or issue requests to the nodes.

Keeping It Simple But Secure

"The great thing about Arduino is that creating a LoRa node and connecting it to a network is so simple," explains

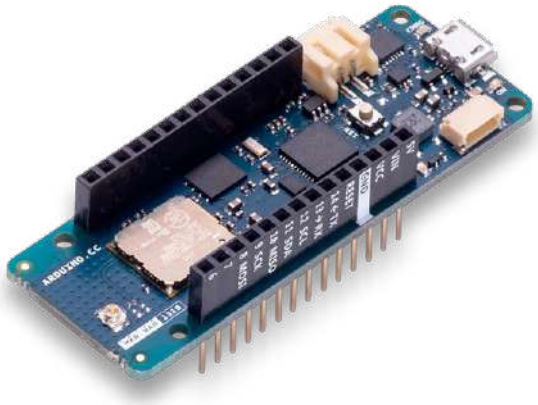


Figure 3: The MKR1310 is an excellent starting point for IoT using LoRa, featuring a SAMD21 low-power microcontroller and Murata CMWX1ZZABZ radio module. (Source: Arduino)

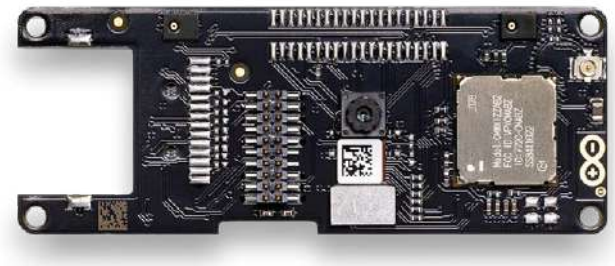


Figure 4: The Portenta Vision Shield – LoRa adds industry-rated LoRaWAN support to Arduino Pro hardware, such as the Portenta H7. (Source: Arduino)

> *The great thing about Arduino is that creating a LoRa node and connecting it to a network is so simple.*

Massimo, "just requiring an appropriate shield and software library." For makers, the go-to-boards have been the MKR1300 and MKR1310 (**Figure 3**). Featuring the low-power SAMD21 microcontroller from Microchip, it is paired with a Murata CMWX1ZZABZ LoRa module and a battery charger circuit. When correctly configured, the board consumes just 104 µA. But, with cyberattacks on industrial systems growing, these technologies must also be secure. This is covered by an ECC508 cryptographic chip that places encryption and decryption in hardware that complies with industry-standard security conventions. "This makes such hardware secure from the first clock cycle," added Massimo.

For professionals, there is the Portenta H7 that, coupled with the Vision Shield (**Figure 4**), gains its LoRa capability using the same RF module and the more powerful ECC608 cryptographic co-processor. Of course, the next challenge is preparing a robust gateway to attach the nodes to. Thanks to a new collaboration with RAKwireless, Arduino now offers two industrial-ready gateways (**Figure 5**) with their WisGate Edge Lite 2 for indoors and WisGate Edge Pro for deployment outdoors. [3] Supporting Power over Ethernet (PoE) for ease of installation, the gateways provide a secure solution with deep coverage inside buildings with support for 16 channels instead of the more commonly supported eight. The gateway can provide access to platforms such as The Things Network. However, if preferred, teams can also configure their own private network with an MQTT client and network server.

Simple Hardware Needs Simple Software

While the hardware may be ready to connect, no progress can be made without software. As is to be expected, Arduino makes this simple both for the board and the connectivity to the cloud. The team recently launched version 2.0 of their IDE, bringing debugging for both on-board and third-party debuggers. With it comes the ability to explore variable contents and code execution like with IDEs from the microcontroller vendors. A core new feature is its command line interface (CLI), allowing professional developers to automate tasks and integrate the Arduino environment with other tools, such as continuous integration (CI) for testing.

IoT applications then have the connectivity dimension to deal with connecting to cloud services. The Arduino Cloud [4] is linked to The Things Network (TTN), a well-known global platform supporting LoRaWAN. Thanks to the use of standard cryptography hardware, provisioning is simple and secure.

"It's important to note that we don't want to lock users into the Arduino environment," explains Massimo. "That's why we ensure we remain compatible with other LoRa standard systems and platforms, giving developers choices as they move from prototyping to deployment. But, above all, our core aim is to make LoRa easy to use."

One of the limitations of LPWAN technologies is the available bandwidth. LoRaWAN can achieve anything from around 5470 bps over 2 km to 290 bps over 14 km (**Table 1**). This is the price that must be paid for long battery life and long range. Developers must then innovate by handling more decisions within their IoT nodes and reducing communication down to results rather than passing

Figure 5: The WisGate gateways provide robust and secure LoRaWAN connectivity for professional deployments, both indoors and outdoors. (Source: Arduino)

Spreading Factor	Data Rate	Range	Time On Air
SF7	5470 bps	2 km	56 ms
SF8	3125 bps	4 km	100 ms
SF9	1760 bps	6 km	200 ms
SF10	980 bps	8 km	370 ms
SF11	440 bps	11 km	40 ms
SF12	290 bps	14 km	1400 ms

Table 1: Comparison of data rare and range for LoRa according to the spreading factor used.

data to the cloud for processing. Inevitably, machine learning (ML) is a popular solution, allowing the microcontroller to make sense of a range of complex input data efficiently. Here, engineers can turn to Edge Impulse, a mature ML solution optimized for low-power microcontrollers.[5]

Can I Fully Commit to LoRa?

New LPWAN and IoT solutions seem to pop up regularly, so it can be challenging to know what to select when your application requires support for the next two or three decades. The recent announcement by Google to retire its IoT Core service [6] and the troubles at Sigfox [7] naturally raise concerns as engineers search for a reliable, long-term IoT platform.

"Around 90% of applications that don't need cellular capabilities are deployed using LoRa," shares Massimo. "And we're seeing further growth, especially in the area of smart cities, such as improving street lighting maintenance, and in agriculture."

When combined with ML, LoRa's long range enables farmers to geofence livestock and attain more insights into the health of their animals.

LoRa, thanks to the members of its global alliance, is also actively maintaining and developing the technology.

"At the physical layer, it is doubtful that any changes will be made," Massimo explains. "However, LoRaWAN can offer further improvements in software, enabling it to adapt to new market requirements."

Some of these changes will require software updates, but these will primarily impact gateways so they can support the new features deployed in freshly developed LoRa IoT hardware. Other work focuses on expanding global support so that bandwidth is legally available for LoRa networks in more countries.

"We're also exploring ways of collaborating with other radio standards, but, again, this will result in software, rather than hardware, changes," adds Massimo.

Seamless Move from Maker to Pro

Looking at the market, LoRa definitely seems to have the upper hand when it matches the IoT application's requirements. Recognizing that many applications start with maker hardware, Arduino offers a competent combination of hardware and software for skunkworks projects exploring available technology. As the solution evolves, the Arduino Pro hardware, WisGate gateways, and an abundance of software offer a path to professional LoRa deployment. Furthermore, Arduino offers a growing range of additional services through their global network of partners who help with system integration and manufacturing. And, for those still on the fence regarding LPWAN technology, LoRa offers stability thanks to its active alliance of members and continued development.

220513-01

About the Author

Stuart Cording is an engineer and journalist with more than 25 years of experience in the electronics industry. You can read many of his Elektor articles at www.elektormagazine.com/cording.

Related Products

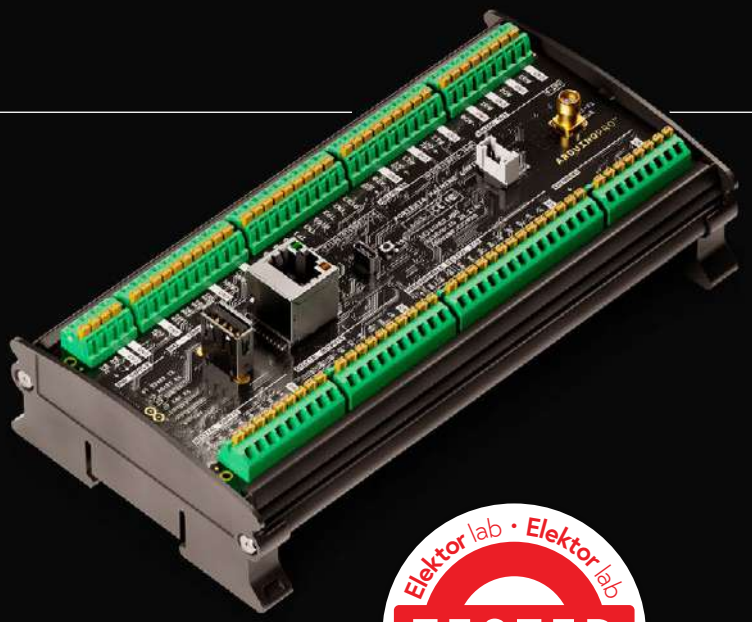
Looking for the main items mentioned in this article?
Arduino and Elektor have you covered!

- **Arduino MKR WAN 1310**
  www.elektormagazine.com/arduino-mkrwan1310
- **Arduino Portenta Vision Shield LoRa**
  www.elektormagazine.com/arduino-portenta-vision-shield-lora
- **Arduino WisGate Edge Lite 2**
  www.elektormagazine.com/arduino-wisgate-edge-lite
- **Arduino WisGate Edge Pro**
  www.elektormagazine.com/arduino-wisgate-edge-pro
- **C. Kühnel, Develop and Operate Your LoRaWAN IoT Nodes (Elektor 2022)**
  www.elektor.com/20147

WEB LINKS

[1] GSMA, "Narrowband – Internet of Things (NB-IoT)": https://bit.ly/gsma-nb-iot
[2] u-blox, "Powering ten years of NB-IoT connectivity with a single battery," January 22, 2019: https://bit.ly/ublox-nbiot
[3] Arduino, "WisGate Edge Gateways for LoRaWAN Connectivity": https://store.arduino.cc/pages/wisgate-lora-gateways
[4] Arduino Cloud: https://docs.arduino.cc/arduino-cloud/
[5] S. Romero, "Image Classification with Edge Impulse," Arduino, September 21, 2022: https://bit.ly/arduino-nicla-vision
[6] S. Evans, "Google Cloud to Shut Down IoT Core Service," IoT World Today, August 23, 2022: https://bit.ly/google-cloud-iwt
[7] R. Daws, "Sigfox Enters Insolvency Proceedings Following Difficulties," IoTnews, January 27, 2022: https://bit.ly/sigfox-iotnews

Unboxing the Portenta Machine Control



By Brian Tristam Williams (Elektor)

Arduinos have entrenched themselves as the go-to boards for educators and makers over the past 15-plus years, with millions of boards shipped, and that's without counting the compatibles, thanks to the ecosystem's open-source ethos. In 2020, the Arduino Pro ecosystem was launched, taking aim at the industrial/PLC market.

The Arduino Portenta H7

Since it forms the heart of the Arduino Portenta Machine Control, let's take a brief look at the first in the Arduino Pro line of boards. The Portenta H7 gets its suffix from the on-board STM32**H7**47XI microcontroller. As an Arduino product, it immediately sets itself apart as part of the Pro line with its black printed circuit board. With the Portenta H7, Arduino has come a long way since 2010's runaway success, the Arduino UNO, which ran an 8-bit MCU at 16 MHz. The Portenta H7 features two cores: an Arm Cortex-M7 running at 480 MHz and another Arm Cortex-M4 at 240 MHz.

Processor aside, there are some other notable features on board:

- 16 MB flash memory, 8 MB SDRAM
- Wi-Fi and Bluetooth 4.1
- The STM32H7 MCU has a built-in graphics engine.
- The on-board USB-C connector supports DisplayPort video output (at a resolution of up to 1280x720), bringing an HMI (human-machine interface) closer to your project without extra driver hardware.

You can still program the board using the Arduino IDE, but the platform offers MicroPython and JavaScript out of the box. With the two cores, you can do things such as use the TensorFlow Lite library for machine vision, recognizing objects on one core, while your MicroPython or JavaScript interpreter processes your code on the other.

The board has two 80-pin high-density connectors on the underside — not as hobbyist-friendly as the UNO's female SIL pin headers, but similarly enabling expansion via an Arduino Pro selection of "shields," or, in the Portenta family's case, complete carrier boards such as the Portenta Machine Control. That doesn't mean that standard 2.54 mm-pitch headers are out of the question — the board still has an MKR-compatible pinout on the edges, where headers can be soldered to break out basic peripherals such as analog I/O and serial buses.

At the price, you're no doubt going to do more than run the Blink sketch on the H7.

Portenta Machine Control

The Portenta Machine Control [1] is a DIN-rail-mountable carrier and versatile interface board for the Portenta H7; in fact, it comes with a Portenta H7 connected on the underside of the board — all at a €299 price point. We had to dismantle the enclosure to find the H7, but it is there, plugged in using the H7's aforementioned high-density connectors (**Figure 1**).

Unfortunately, the H7 being embedded in the innards of the Machine Control means that its USB-C connector is not readily accessible, so, those convenient DisplayPort applications won't be as practical to realize. Besides, the H7 that ships in the Machine Control is paired closely with the carrier board, so Arduino does not recommend separating the two and trying to use the H7 in a standalone capacity – in fact, they say, it may void the warranty.

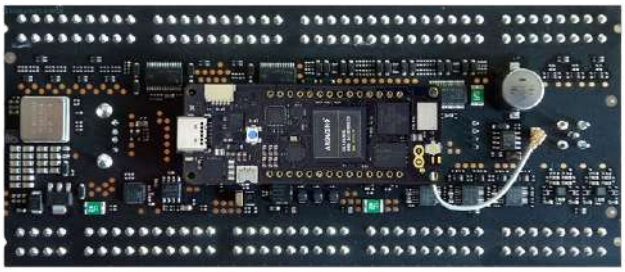


Figure 1: Arduino Portenta H7 on the underside of the Portenta Machine Control.

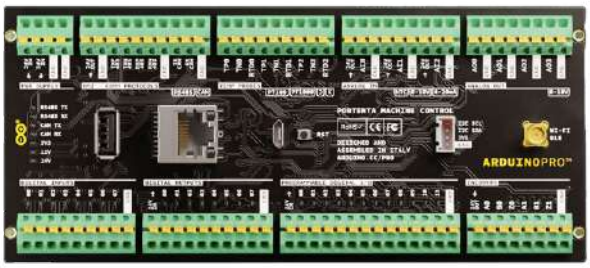


Figure 2: Arduino Pro Portenta Machine Control.

Like the H7, the Portenta Machine Control carries Arduino Pro's differentiated black and olive-green branding, right from the box it comes in.

Measuring in at 50 mm × 90 mm, the Machine Control board breaks out an impressive range of I/Os via 80+ green push-in terminals, broken up into 9 different sections (**Figure 2**). Clockwise, from top-left, we have:

- Power supply: 2 × 24 V inputs, with their associated grounds
- Communication protocols: Balanced RS-485 (or -422 or -232) and CAN
- Special inputs just for temperature probes, which can be PT100/PT1000/J/K
- Three analog inputs, suitable for NTCs, 0-10 V, 4–20 mA, with each input having an associated 24 V output and ground
- Four analog outputs providing 0-10V
- Encoders
- Programmable digital I/O — just like with the traditional Arduino, you decide whether these are inputs or outputs in software
- Digital outputs
- Digital inputs

Between the rows of push-in terminals, the center of the board squeezes in some more general I/O, including:

- A neat column of LEDs on the left side gives an instant indication of whether there's any activity on the RS-485 and CAN buses, as well as the presence of 3.3, 12, and 24 V supply voltages.
- USB-A female full-speed connector, which can be used as host or device
- On-board Ethernet connector with built-in transformer
- Micro USB female half-speed connector – the Portenta H7 can be programmed from here
- A reset button
- I²C on a Grove connector
- SMA connector for the Wi-Fi and Bluetooth (broken out from the H7's RF connector)

Oddly, the assembly line workhorse doesn't come with an antenna, but the SMA connector gives you the flexibility to use either an on-device antenna or have one placed in a more suitable location within your environment.

With the Ethernet jack available, you may of course opt to keep the airwaves clear in your industrial environment by skipping the Wi-Fi and hard-wiring the network.

The plastic DIN rail mount/enclosure doesn't feel as substantial as one would expect from a typical PLC, but this does bring the weight down to under 200 g.

Overall, the Portenta Machine Control manages to bridge two worlds, bringing IoT capabilities to standalone industrial machines, bypassing the use of traditional PLCs, and offering multiple development environments, which offer a lower barrier to entry for those who don't spend 8 hours a day writing industrial automation software.

The plethora of inputs and outputs mean that the solution that will likely connect to any sensor or output in a manufacturing chain with no extra complex circuitry. So, at that price point, you get a drop-in hardware solution with flexible programming options from a selection of languages and development environments, and the ability to run two blazing-fast cores independently and simultaneously, with both having access to all of the available resources.

220532-01

Questions or Comments?

Do you have any questions or comments relating to this article? Please feel free to get in touch with the author at brian.williams@elektor.com or contact the Elektor team at editor@elektor.com.

Related Products

Looking for the main items mentioned in this article? Arduino and Elektor have you covered!

- **Arduino Portenta H7**
  www.elektormagazine.com/arduino-portenta-h7
- **Arduino Portenta Machine Control**
  www.elektormagazine.com/arduino-portenta-machine-control

WEB LINKS

[1] Arduino Portenta Machine Control Product Datasheet: https://content.arduino.cc/assets/AKX00032-datasheets.pdf

Q&A With Kevin Bates

Founder, Arduboy

8-Bit Gaming with Arduboy

Arduboy FX .

By David Cuartielles (Arduino)

Back in early 2014, Kevin Bates built the Arduboy gaming system prototype using Arduino. About a year later, he launched a Kickstarter campaign, and the rest is history.

David Cuartielles: Before we get into the Arduboy and its history, tell us a bit about yourself and your interests. When did first get involved with electronics? When did you start using Arduino?

Kevin Bates: I've always been interested in taking things apart and putting them back together for as long as I can remember. Building my own PC as a teenager was my first hands-on experience with electronics, being active in things like overclocking and adding LEDs to my case.

Figure 1: Kevin Bates .

I had a career as a wind turbine technician. I would fly out to various wind farms in the US and Canada when the team on site needed help with a turbine that wasn't behaving correctly and help troubleshoot it. Growing up, I had various 100 in 1 Electronic Kits with the little springs on them. I'd made FM radio before, but this job gave me a formal education in learning to read schematics and forced me to respect Ohm's law because you get a lot more than magic blue smoke when you hook these things up wrong.

I was surprised to learn that these giant multi-million dollar turbines often run on 8-bit microcontrollers. Around this same time (2008-2009) Arduino was starting to get popular and I thought: "Wow, you can really just plug in a USB cable and it works?" I honestly didn't believe it would be that easy. From the rush of the first blink sketch, I was hooked.

Cuartielles: Tell us about the Arduboy prototype, which went viral in early 2014. Can you describe the characteristics of the original Arduboy? How did Arduino help you in making the first model?

Bates: Well, I started with the idea of making a digital business card, using a PCB as the actual form of the card. These kinds of functional artwork pieces had been done before. There was one with a USB connector

and a kind of rubber ducky script that would type out a resume and open a website. Originally, I set out to play a simple game of Simon Says with four LEDs and touch pads. As it turns out, I had an OLED display from another project and found it worked on very low power of a coin cell. So, I figured this was a great opportunity, and it needed controls, so I added a direction pad and then, an OK and Cancel option (**Figure 2**). This was at the schematic stage, and it wasn't until I was trying to figure out how to arrange them, that I realized what kind of shaped device it had to be.

The real trick, which made it go viral in some ways, was the fact that I used cutouts in the PCB to place all of the components mid mount. The buttons were capacitive so that the entire device was not really any thicker than 1.6 mm of the PCB. My idea to do this came from dropping a capacitor into a through hole one day and wondering if it would be possible to mount components like this (**Figure 3**).

The open-source community surrounding Arduino is really what facilitated this. Online retailers that were sharing their board files, schematics, and source code allowed me to learn from their examples. Arduino specifically was a software and platform that "just worked." I don't have a lot of patience to set up a traditional build environment. When I see someone's code repository online calls for using Make and then has several pages of how to install dependencies, it makes me want to puke. Arduino bypasses all of that. When you get error messages, you can just google them. It just works (**Figure 4**).

I also had the chance to meet some of the founders after it went viral, and they have all been very awesome and friendly people as well!

Cuartielles: You launched a Kickstarter crowdfunding campaign in May 2015. How did it go? What were the results? When did you start shipping?

Bates: Wow! Long time ago. It went amazing, almost half a million dollars. I was overwhelmed! The pressure of shipping to almost ten thousand people gave me a lot second guesses on making the final design. I spent half a year trying to come up with a better audio amplifier to just scrap the design entirely.

The community was with me the whole time though. The development was done in the open, and there are a great crew of people on the Arduboy forums that have supported the process. A ton of the credit goes to the people that contribute there, because without it, the Arduboy wouldn't be what it is today.

Eventually, I shipped about a year after the Kickstarter campaign finished, which was about twice as long as expected. Considering the final product though, it was worth it. Everyone seemed very happy with the results. It was a fun and amazing experience for me since I was overseas in China doing the development firsthand, so I also owe a lot of thanks to all my backers for making it happen.

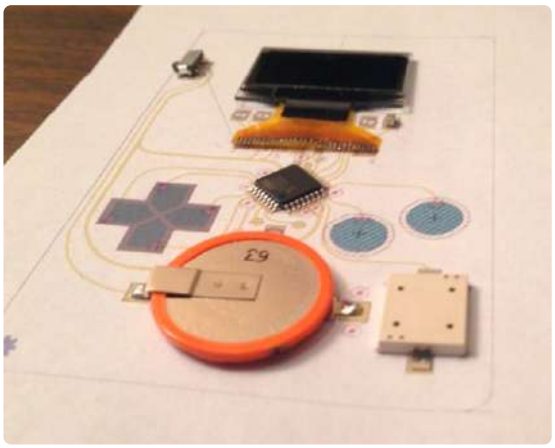


Figure 2: Arduboy prototype.

◀

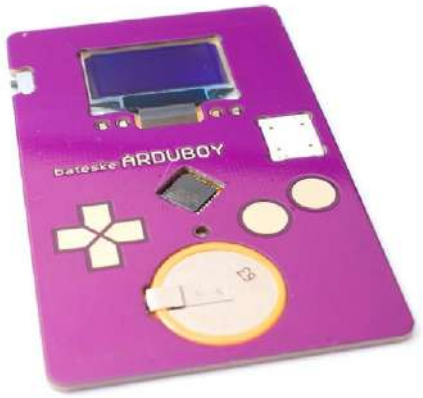


Figure 3: Board with cutouts.

◀

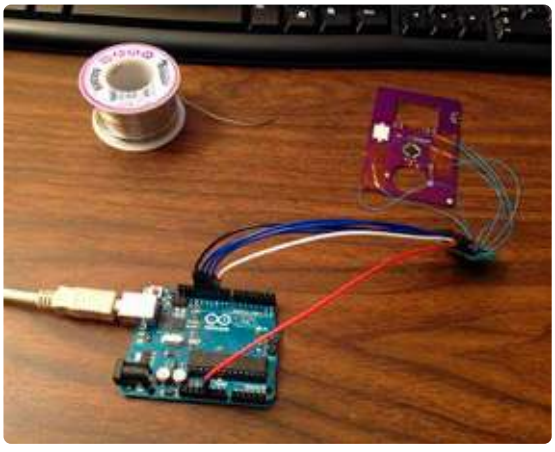


Figure 4: Working with Arduino.

◀

Figure 5: The Arduboy Nano .

Cuartielles: You seem to be very active in redesigning the console. How many different editions have you made?

Bates: This question makes me laugh! I often feel like a one-hit wonder band trying to come out with a second album. Trying to make lightning strike or something. I never intended the Arduboy to become any kind of product, so now when I go forward, it's a difficult mindset to be in.

Figure 6: The Arduboy hardware in glasses.

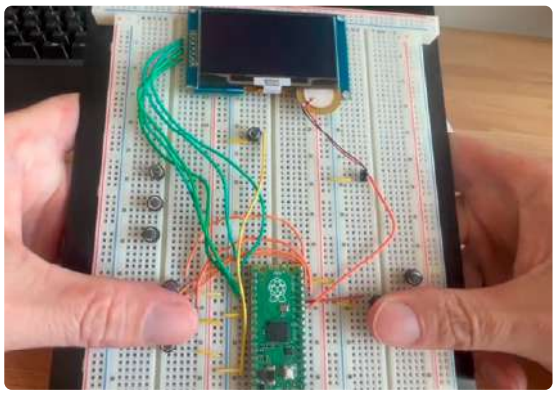


Figure 7: Messing around with the RP2040.

The whole philosophy of the Arduboy was to be a minimum viable game platform. I feel like it achieved incredibly well, but where do you take that kind of a maxim? How do you become "more" minimal?

As soon as I start adding new features, it starts to compete with other platforms or runs into challenges of being too expensive or impractical to build at small to medium scale. For a long time, this kind of decision-making process forced me into a state of paralysis because I was trying to solve some arbitrary problem. For the past few years, I've freed myself from that and have been doing what I was doing when I made the first Arduboy prototype — having fun and making something that makes me laugh. You can always follow along at my Twitter: https://twitter.com/bateskecom.

So, in this vein, I went as small as I could borrowing design techniques from a friend at Tiny Circuits to make the Arduboy Nano, a 3D-printed game system that was smaller than a quarter (**Figure 5**). This used a smaller display of only 32 × 48 pixels, but used the same display controller, so only slight modifications were required to the library. I had been working with a transparent OLED for a few years, and with some help from people on Twitter, I was able to get them deployed into a pair of glasses using PCB as the frames. The Arduboy hardware powers it and is built into one of the stems (**Figure 6**).

Since then, I've been messing around with the RP2040 and learning its capabilities and what kind of displays might pair well with its performance (**Figure 7**). There is a larger 2.3" version of the OLED used in the Arduboy, so I tried that, and it's quite a performance increase over the 32U4.

I picked up a subscription to Fusion 360 and was having fun imagining what a new case design for the new hardware could look like. During that time, I also got turned onto a new display module. The translative displays have become more common now and with the Playdate console making the 400 × 240 monochrome a "format" for gaming, it opens a lot of potential to use that same screen and open up development.

Because minimum order quantities are a huge challenge in trying to build anything with injection molded parts, I've been curious about how to use existing shells that are affordable and available all over the world. So, I'm trying to cram all of that into a piece of plastic designed and produced by someone else.

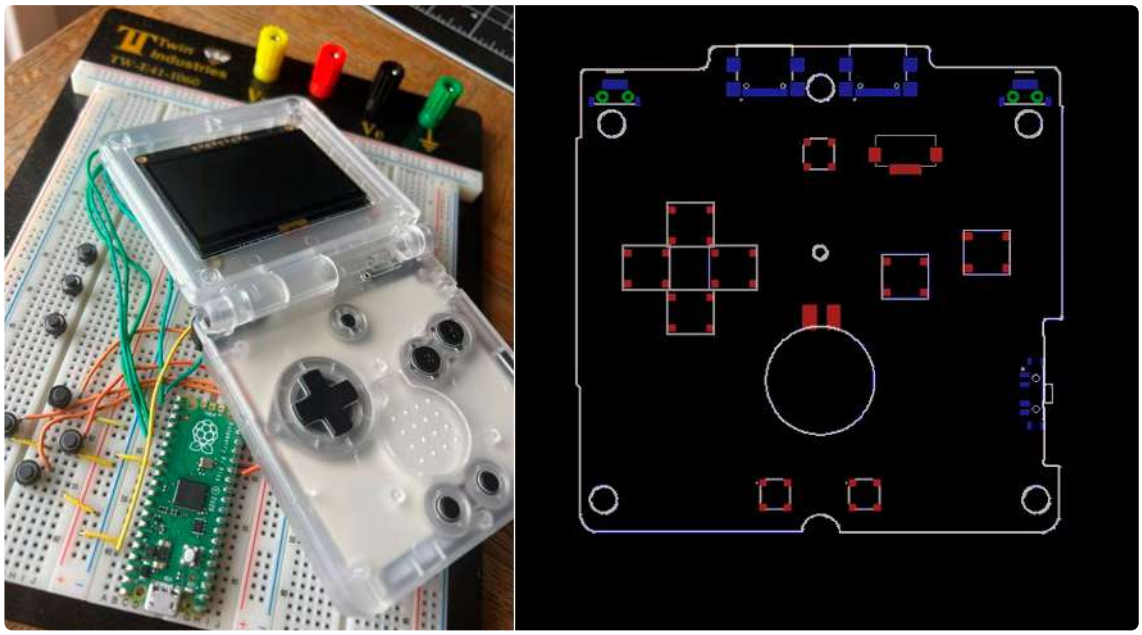


Figure 8: RP2040-powered console .

Finally, I'm trying to go full circle with a product that Super Impulse licensed to make the Micro Arcade and replace the PCB inside and reuse the internals to make an RP2040-powered console, with the color screen and credit card shape people are familiar with (**Figure 8**). People already have *Doom* running on this chip, so it's only a matter of time before you could potentially have Doom in your wallet. You can follow along at my Patreon where I'm sharing all of my prototypes and shipping out extra PCBs and parts I make in the process: patreon.com/bateskecom (**Figure 9**).

Cuartielles: You have a great community behind the console. Can you mention the main contributions you have had so far?

Bates: The Arduboy community is everything. Everything besides the actual layout of the PCB and the website has been something contributed by someone else – if that puts it into perspective. In the beginning, I shared the hardware development kits and received feedback. There were changes made to the schematics based on suggestions from users. The library has been completely rewritten and optimized on a per-cycle basis uses inline assembly by users and is moderated entirely by them. You can now get over 60 fps with over a hundred bitmaps animated on screen.

All of the games are made by members of the community. There are tutorials that users have created that provide step-by-step instructions for a beginner to learn how to code their own game. Technical and coding support is 24 hours and provided often times in multiple languages.

An emulator of the hardware has been created and now functions as a way to experience the Arduboy platform without even requiring the hardware. Users of the community can play Arduboy games right in the browser.

The hardware modification created in the community to add external flash memory was incorporated to a mod-chip to add all of the Arduboy games on board and is now built into the current Arduboy FX.

Arduboy continues to have the power to change peoples lives by providing an open and creative space to learn about coding video games and developing game hardware, and it's entirely due to the ongoing support of the people who share on the forum.

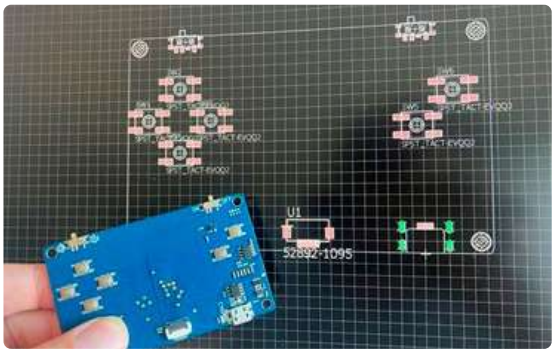


Figure 9: Another prototype.

The whole philosophy of the Arduboy was to be a minimum viable game platform.

I'm extremely grateful to them and try to give thanks as much as possible. I've put their names on the back of the Arduboy PCB, and most recently featured their graffiti tag names on the back of the newest project: the Arduboy Mini (**Figure 10**). You can find out more info about it here: arduboy.com/mini.

Cuartielles: Do you have a favorite Arduboy game?

Bates: Can I choose my own game, *New Blocks on the Kid*, a non-trademarked falling block game? Actually, to be honest, I think the best game out there for the Arduboy is *Circuit Dude*; it's gone on to be available on mobile, PC and the Nintendo Switch. It's been exciting to see its success. Other than that, it's a pretty basic, but *Snelk* is a snake remake, and it's a classic.

Cuartielles: Where are you today? How many people are involved in making the consoles?

Bates: Well, I'm still here. And so is Arduboy. It's going to be 9 years old soon! Currently based out of San Francisco, it's been weird not travelling. I previously thought I'd be in China at least once a year. Many things have changed, but much is still the same. It's just me at the company. I've had some employees and contractors for the years, but running the business, it's just myself. If you catch me using "we," it's like the royal "we," and I'm referring to the work from the community. Seeed Studio is still producing the Arduboy FX. They subcontract out to a factory that does the assembly. I'd like to go back sometime soon; it was cool to be able to have lunch with the people actually doing the work.

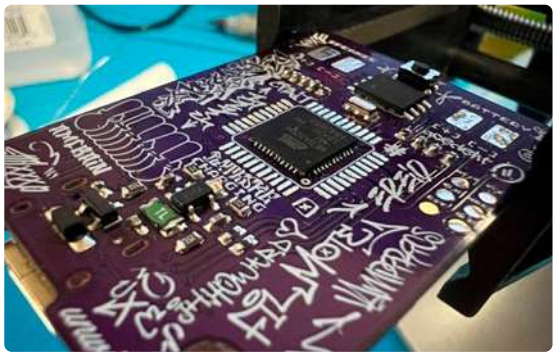


Figure 10: Arduboy Mini .

Cuartielles: What sort of challenges do you face today? Getting parts has been hard for many electronics companies over the past last 36 months. How will you address this moving forward?

Bates: Besides the challenge of trying to come up with the new product, mostly normal business stuff. I run all the back office stuff, like paying taxes so that all takes time away from the fun stuff.

Luckily, the hardest of work, sourcing the components, is something that Seeed Studio does. They just tell me how long it will take to produce, and when those ETAs are over 9 months or more than a year, then it's difficult to explain to the customer to wait that long. This last batch was saved by the folks over at Arduino who were kind enough get me some chips they had in stock. It was a small amount to them, but 1,000 units of shipped Arduboys makes a huge difference. So a great deal of thanks goes out to them!

Arduboy has continuously been plagued with the problem of not being able to have enough supply to meet demand. I've never been able to keep them in stock, so from that perspective, it's not all that different.

The political situation is worse off than it was when I started the company too, which now sees 25% taxes in the import of my product. Basically, that all came out of what I pay myself. Thankfully, I had a rocket ship start to this company that allowed me to make a big name and presence. I was at the right place at the right time. It would be impossible to start Arduboy today. So, like I said before, I'm still here so ... I've got to keep going!

Cuartielles: **What's next for Arduboy?**

Bates: I'm trying to address a bigger question, a bigger problem. Arduboy has been successful beyond my dreams, and there have been dozens of examples of people who Arduboy has changed their life. I have emails from people who tell me stories about how they achieved their dream of becoming a professional game developer after starting to learn to code and make their first game with the Arduboy. And it's not an isolated incident; this is a pattern of people who find the console, and they not only enjoy it, but it improves their lives.

This is what I'm trying to understand and capture — and scale. In the world today with so many obvious problems, how can I justify working on a little video game system? Well, I think learning to code games teaches you skills like problem solving, goal setting, critical thinking, communication skills if you are

doing it with other people. These are the so called "soft skills" that I think are most important to learn and are commonly not taught in schools.

So, how can I take what I've found that works, this platform of open-source game development, and make it more accessible? Right now, the easy answer is to make it cheaper, and the first step is with the Arduboy Mini. It's way easier to kit out a classroom with a set of these when you skip the expensive bits of the experience.

I'm also working on continuing the work of the community built emulator and trying to find a way to promote the platform without the hardware. Making the emulator into an app that runs on mobile and chrome devices with a built-in curriculum would be the ultimate evolution of what I started to build.

But that dream is big, and so I guess you could say I'm looking for partners to work with, or potentially people who are already in that space who I could join. If you want to go fast, go alone; if you want to go far, then go together. And I've gone about as fast as anyone can, and I'm going to be 40 next year, so I start to think about legacy. I'm trying to figure out how I can turn Arduboy into something that I can walk away from knowing that I took it to its absolute maximum. I would also like to disrupt the graphing calculator market, but that's a whole other story.

220543-01

About the Author

David Cuartielles co-founded Arduino. He holds a PhD in Interaction Design and an MSc in Telecommunications Engineering, and he teaches at Malmö University.

Related Products

Looking for the main items mentioned in this article? Arduino and Elektor have you covered!

- **Arduboy FX - Open Source Card-Sized Gaming Board**
  www.elektormagazine.com/arduino-arduboy
- **Arduino Nano 33**
  www.elektormagazine.com/arduino-nano33
- **Nano RP2040 Connect**
  www.elektormagazine.com/arduino-nano-rp2040-connect

Reducing Water Usage at Horseback Riding Tracks

An IoT Solution to Constantly Monitor Soil Humidity and Temperature Levels

By David Cuartielles (Arduino)

For horse racing, keeping track conditions just right has long been a manual task requiring expert care from skilled groundskeepers, but still, it's impossible to monitor everything, everywhere, all at once. Anders Åkerberg, Project Manager at Jägersro horseback riding arena, went knocking at Arduino's door for some smarter solutions.

Jägersro horseback riding arena, probably the most important one in southern Scandinavia, needed to cut down on water consumption. Anders contacted us to discuss a smart way of doing this. "We are, of course, interested in making our track as environmentally friendly as possible. Currently, we have an expert who checks the arena and decides when the watering tractor should be called in to help maintain the correct soil moisture level. There are many variables involved in this decision, from the angles of the different parts of the lane, to the temperature differences due to the incidence of the sun, or even how the wind blows over the track. There are simply too many external factors to take into account." But, what seemed like a simple automation process opened up a very interesting project for exploring the use of the Arduino Portenta H7 as an on-site data collection and decision-making tool. This article describes the first steps we took in the project, as well as the kinds of sensors chosen for the project and why. All the learnings presented here can be applied to many other IoT projects, from arenas requiring the use of real grass, to fields that need to be carefully irrigated for the production of vegetables.

Background

In 2017, the decision was made to build a new trotting sports facility in Malmö, Sweden (**Figure 1**), as a replacement to the existing one, which could not meet the expectations of competitors, horses, spectators, and even society, of what a modern sports facility should be. The plan includes a redesign of the entire facility, which should be ready by 2025.

Horseback riding is a widely followed sport in Sweden and wider Scandinavia. There are different types of sports; jockeys can ride the horses directly, or they can sit in a competition trolley (called a sulky) pulled by a horse in what is known as *harness racing*. Regardless of the sport, the arena is the same. It consists of a 1000-meter-long track built with organic materials such as sand, gravel, and stones in different sizes. The quality of the track is supervised by specially dedicated staff who constantly monitor and correct its condition. One of the main aspects to take care of is the track's humidity. To minimize the impact it will have on the horses' hooves and legs, it shouldn't be too wet or too dry. The effect of the properly watered gravel is called damping, which, due to the horses' weight, turns the moist gravel into a mix between a spring and shock absorbers. The quality of the ground is subject to constant controversy

between the contestants and the track's personnel. Currently, handheld devices are used in parallel with the experts' surface observations to determine the track's optimal moisture. Irrigation is done using a dedicated tractor with water sprinklers. Such a situation offers great opportunities for exploring possible automation to help the track's expert get better information faster and to ensure a uniform and predictable track quality. Theoretically, this will also bring a reduction in water consumption by the track, through a more precise determination of the locations and quantity of required water.

The Problem

The real challenge lies in being able to determine the right amount of water to be used for each section of the track precisely. The track's very nature is to blame for this because:

- the inner track and the curves have different inclinations (2 degrees versus 18);
- it is impossible to make the layers of gravel on the track's surface have exactly identical thickness and composition;
- even if we managed to achieve that, the horses would destroy it as soon as they ran their first race;
- the orientation of the track gives it nonuniform solar illumination (remember evaporation?);
- the wind affects conditions occasionally;
- periodically, or by decision of the track manager, the surface is "shaven" and a few centimeters of gravel will be removed and substituted.

The Solution

As you can imagine, the variables affecting the quality of the track are far too numerous to solve the problem of where to water the track and when using techniques such as computer vision. The most reliable way to know where to water the track requires the installation of a permanent sensor network buried under the track's thinner top layer of gravel. After having collected data, it should be possible to model the track and use other mechanisms to determine moisture levels. This project looks at that first stage. Our goal is to create a connected track where moisture will be calculated in real time at multiple locations at once.

Our idea consists of building a system that will be used to monitor the track's quantitative data (humidity and temperature) and correlate it with observations made by the track managers. As mentioned above, it is expected that similar sensor information coming from different points on the track will result in different quality determinations. Thus, it's important to figure out how to train a system that will be able to provide the track manager with data in a way they can relate to and not with simple raw sensor values. Ideally, this system should be able to cut down the time needed to wander along the track collecting data, and decisions could be made by simply looking at a visual representation thereof.

▲

Figure 1: Area for new Jägersro track. (Source: Jägersro Hästcenter)

For Anders, this was the first step towards the arena's watering system automation and water consumption reduction, while improving the track's conditions by keeping them as uniform as possible. To me, this project represents the opportunity to try out some complex technologies. And, I am talking about non-trivial, expensive sensors (priced at between € 150 and € 350 a piece). To create this article, I studied a family of sensors that have been around for a while, but which are typically out of reach for most of us. I hereby present the first steps of this project as a way to illustrate how others can pursue a similar approach. The same technology could be applied in the field of smart agriculture, maintenance of grass fields for sports, detection of leakages of liquids… essentially, any kind of project having to deal with water and soil of any kind.

The ultimate solution, covering the entire track, will have a yet-to-be-determined number of sensors and Arduino Portenta H7 boards. Part of the research being done requires an understanding of how many meters of separation there should be between sensor points. It's also unclear whether we should measure at different points across the track or, given that it is tilted toward its interior, it will be enough to use a single measuring point. On the other hand, it's possible to have sensors that need up to 70-meter-long cables, but this makes the sensors even more expensive, since they have to be hand-calibrated at the factory, and they are typically not manufactured to handle such long cabling.

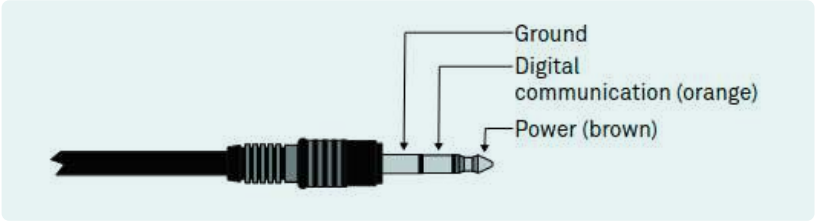


Figure 2a: Jack connector. (Source: METER Group, TEROS 11 Integrator Guide)

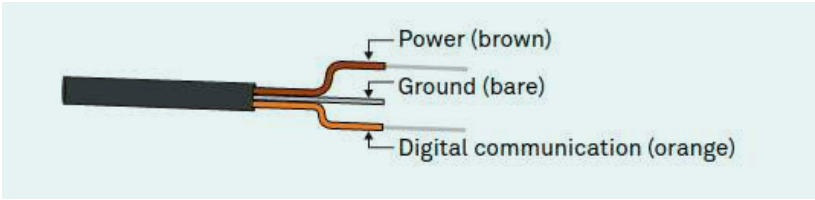


Figure 2b: Bare, tinned wires. (Source: METER Group, TEROS 11 Integrator Guide)

Table 1: Basic comparison of the different sensors under study.

Feature	TEROS 10	TEROS 11	TEROS 21
Resolution	0.001 m^3/m^3	0.001 m^3/m^3	
Temperature	-40 ... 60 °C	-40 ... 60 °C	-40 ... 60 °C
Measures	humidity	humidity, temperature	water potential, temperature
Analog sensor	1 ... 2.5 V	1 ... 2.5 V	
Communication protocol	analog out	DDI or SDI-12 serial	DDI or SDI-12 serial
Volume influence	430 ml	1010 ml	
Frequency	70 MHz	70 MHz	70 MHz
Voltage	3 ... 15 VDC	3 ... 15 VDC	3.6 ... 15 VDC

Water Potential

One relevant concept I learned during the preliminary research for this project is "water potential." To better understand what this means, you should think in terms of thermodynamics. Excess heat always flows in such a way that renders two touching surfaces equal in temperature. Water potential can be seen similarly: Water will flow from areas with higher water potential toward those with lower. From that perspective, we cannot measure water potential as an absolute amount — it has to be interpreted as a relative one. Pure water provides the reference value for this potential. Having various solvents in the water, or having the water distributed throughout the soil, results in a change in water potential. So, if you're about to dive into the datasheets of any high-end soil humidity sensors, it's important to bear this in mind.

The Sensors

For this article, I compared some sensors from METER Group's TEROS family. These sensors have been around for a while (the company was called Decagon before). These are digital sensors – they have a microcontroller on the sensor that takes measurements and sends the information back to a different device, whether in analog or digital form. Most companies that sell such sensors also offer handheld data loggers for use in the field. Therefore, it's possible to find the sensors with either a reader-friendly jack connector or with stripped and neatly tinned tips, as shown in **Figure 2a and 2b**.

I guess most readers are familiar with cheap soil humidity sensors, which look at either soil conductivity or soil capacitance. They come in a blade shape with between one and three tips, and send back a signal that can simply be read using an analog pin on a microcontroller board. These cheaper sensors are great for indoor use and for projects where they are easily changed out in case of failure, but burying them and expecting them to survive sub-zero winter temperatures and endless mechanical traction due to moving gravel doesn't sound like a good idea.

On the more professional end of the spectrum, we find the sensors I chose to evaluate. There are two different families of sensors – those with metallic terminations and those using piezoelectric technology. **Table 1** shows some basic sensor characteristics, just to give you an idea of how they work. I refer you to the references for their datasheets if you need further information.

Sensors TEROS 10 (**Figure 3**) and TEROS 11 (**Figure 4**) use the same technology for humidity calculation, and are thus identical when it comes to what they can do, apart from the area they can cover (known as *volume influence*), which is more than double for the TEROS 11. As you can see in the sensors' pictures, these first two devices seem identical, besides one having two pins while the other has three. The TEROS 11's third pin is a thermocouple for measuring the soil's temperature. This is a big difference between the two sensors, and it also affects the way data is sent back to the microcontroller. The TEROS 10 sends a raw analog signal, which can be read directly by an Arduino board by using the `analogRead()` function. I tested this by simply connecting the sensor's power/ground/signal wires to pins on my Portenta Breakout

board. Powering the sensor at 3V3 was enough for it to work, despite the 5 m-long cable between the sensor and my Portenta. On the other hand, the TEROS 11 and TEROS 21 (**Figure 5**) can deliver more than one kind of value (humidity and temperature, for starters), so it's necessary to implement a digital protocol for communication with the sensors. This is an advantage, since it's now possible to configure the sensors to communicate using SDI-12, a specific type of bus technology (more on this later). This is also the reason why the TEROS 11 and TEROS 21 need higher voltages for operation. In my tests, I had to use an external 12 V power supply to make sure the sensors were operating as expected (**Figure 6**).

DDI vs SDI-12

When using SDI-12 (Serial Digital Interface at 1200 Baud) it is possible to connect multiple sensors at 1200 bps in parallel and read information from them using special, human-readable AT commands (see SDI-12's site for more information [5]). Also, these sensors have implemented a fallback technique called DDI-Serial, which forces the sensor to send a text string to the communication port upon reboot. **Figure 7** shows a screen capture of Arduino's Serial Monitor terminal receiving information during multiple power cycles of the sensor. In this case, I had just thrown some ice cubes in a container with water, and you can see that it's possible to see how the humidity value stays consistent while the temperature drops. The final characters at the end of every line are the sensor-specific identifier, checksum byte, etc. Therefore, one possible way to realize this project with a TEROS 11 or TEROS 21 is already solved using the DDI data bursts after each sensor power cycle. I would just need to implement a board where the different receiving pins could be joined together and have different MOSFETs to power sensors up separately. I would also need to check the impedance and calculate how many sensors I can have simultaneously and if I would need a driver of sorts.

Both DDI and SDI-12 are 1-wire 7N1 serial communication protocols. It is possible to find one well-documented and maintained comprehensive serial library for AVR-based Arduino boards [6], but this makes use of Arduino's *SoftwareSerial* library plus hardware interrupts to ensure that it receives signals properly. The library requires minimal hardware to connect to the sensor. And since *SoftwareSerial* is half-duplex, you

Figure 3: Meter TEROS 10. (Source: Kristoffer Engdahl / Arduino)

◀

Figure 4: Meter TEROS 11. (Source: Kristoffer Engdahl / Arduino)

◀

Figure 5: Meter TEROS 21. (Source: Kristoffer Engdahl / Arduino)

◀

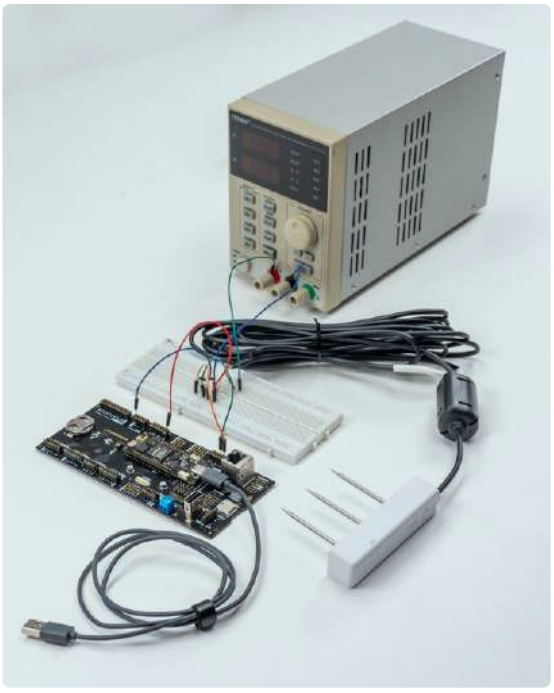
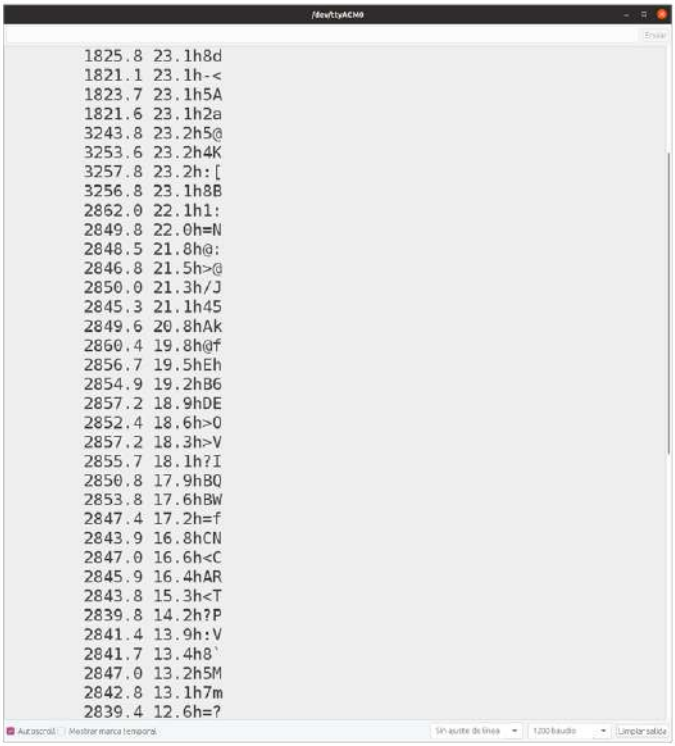


Figure 6: Experimental setup. (Source: Kristoffer Engdahl / Arduino)

◀



▲

Figure 7: Arduino's Serial Monitor displaying DDI messages from TEROS 11.

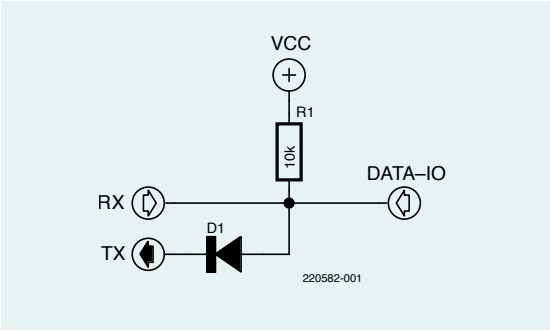


Figure 8: Two-wire-to-one-wire serial converter.

can directly connect the data cable from the sensor to a pin, and it will work (as long as you have a 3V3 board, or else you will need voltage conversion to protect the sensor). Portenta has no problem with hardware serial ports, so we won't use a *SoftwareSerial* implementation (and because we compile over mbedOS, which doesn't play nicely with interrupt-driven firmware). In my case, I used the trick of using a hardware serial port (pins 13/14 on the Portenta H7), with TX connected via a diode to a pulled-up RX pin. This is a well-known trick that should work in this case (spoiler alert: It does, but only for receiving, not for sending). **Figure 8** shows the schematic for the 2-wire-to-1-wire conversion.

Therefore, the next step for me, now that I have DDI communication, is to implement a hardware driver to enable sending data requests to the sensors, thus supporting SDI-12. This will give me a second way of obtaining information.

Power-Cycling the Sensors

Something that's explained in all the datasheets for these sensors is the need to turn them on briefly, just to read the data, and then turn them off again. The reasons for this are both to reduce power consumption and to extend the life of the electronics in them. As a matter of fact, the readings could be compromised if the sensors are left "hot" for long periods of time. This applies to all three of the sensors under test. This concept of power-cycling implies that I will have to include MOSFETs to activate and deactivate each one of the sensors – something I was already planning to do as a way to recover information via DDI. Since I am powering the sensors at 12 V, I can look for relatively strong MOSFETs (such as the IRFZ-24, one of my favorites) which would allow me to reuse the board that I'll need to design for other situations.

Choosing the Right Sensor

It is clear to me that I need to capture temperature information and not just soil moisture, so the TEROS 10 is not an option. After some initial tests in the lab, the TEROS 11 gave me a better impression in terms of mechanical robustness and waterproofing. The sensors will have to be buried in gravel, and I expect them to be in direct contact with water, so TEROS 11 also seems better in that sense. However, I cannot make up my mind without doing any kind of fieldwork. I plan to bury a couple of TEROS 11s and TEROS 21s in the track for a couple of months to gain some insight into their resistance. If price were a limiting factor (it might be, later in the process), the TEROS 21 costs over € 350 per unit – more than enough reason not to select it. The TEROS 11 comes in at a little more than half that price. If I were to put a group of three sensors every 10 m on the track, I would need 300 sensors. I'll leave it to the reader to do the math of how much that would be, just for sensors.

Next Steps

This project is far from over. I am currently working on the design of a carrier for Portenta's Breakout board that will host the power inlet for the sensors, the power-cycling MOSFETs, and the drivers for 1-wire serial communication. Once I have that in place, I will choose a rugged 12 V power supply, a standard ABS

enclosure, and I will install it on site and both save the data to an SD card and report it over Wi-Fi to the Arduino Cloud (there is a Wi-Fi connection on the track). In a couple of months, I will finally know which sensors are best. If you also want to know, follow my progress on the Arduino blog [7], where I expect to publish the different steps of the process as I discover new things.

220582-01

Related Products

- **Arduino Portenta H7 Development Board (SKU 19351)**
  www.elektor.com/19351

WEB LINKS

[1] Jägersro Hästcenter: https://jagersrohastcenter.se/
[2] METER TEROS 10 manual: http://publications.metergroup.com/Manuals/20788_TEROS10_Manual_Web.pdf
[3] METER TEROS 11/12 manual: https://publications.metergroup.com/Manuals/20587_TEROS11-12_Manual_Web.pdf
[4] METER TEROS 21 GEN 2 manual: https://publications.metergroup.com/Manuals/20854_TEROS21_Gen2_Manual_Web.pdf
[5] SDI-12 Support Group: https://sdi-12.org
[6] Arduino-SDI-12 library on GitHub: https://github.com/EnviroDIY/Arduino-SDI-12
[7] Arduino Blog: https://blog.arduino.cc/

The Panettone Project

A Sourdough Starter Management and Maintenance System

By Daniel Fantin (Australia)

This Arduino project is born of the frustration of many, many a failed Panettone. For those who don't know, Panettone is a traditional Italian sweetbread cooked for Easter and Christmas. It has an 'impossibly' light and airy texture, which requires a highly active yeast from a sourdough starter.

Figure 1: A wonderful Panettone, isn't it?

Take a look at **Figure 1**. This is what a successful Panettone looks like - light, airy, packed full of butter and fruit. If your mouth is watering, this article provides everything (technical) that you need to bake such a great cake perfectly yourself. Buon appetito!

What's a Sourdough Starter?

Simply put, it's a community of bacteria and yeast that, fed with fresh flour and water, releases gases that create an airy, risen bread. The yeast needs to be strong enough to fight the gravitational load of the bread or the Panettone dough. In the case of bread, it's not so bad, but for Panettone, we're going up against vast amounts of butter, fruit, and sugar – a tough ask for any yeast.

When is a sourdough starter strong enough to be used in Panettone? Simple: When it triples in height in four hours, while being maintained at 27 °C. That sounds simple, at least, but maintaining this is difficult. Nobody really wants to heat their house up to 27 °C just for the sake of their starter, let alone keep that consistent for weeks on end while the strength of the starter increases. And, how do you know if it has tripled in size in four hours? Would you check constantly? Set alarms and keep a tape measure in the kitchen? Too much, too difficult, too involved. That's best left to a robot.

What happens when the starter isn't strong enough? The resulting Panettone is dense, difficult to eat, and looks like failure. What about when it is strong enough? See Figure 1, again. You just know that it's delicate, fluffy, airy, velvety, sweet, crunchy, buttery ... everything you want it to be.

So, it's May at the time of writing, and I have a while before my next batch of Panettone is due (December). When we were asked in my Deakin University subject to solve a "real-world" problem, Easter had just passed, and I'd just failed with a bunch more Panettoni. So, it seemed like a real-world problem worth solving.

First, I thought about buying a fermentation device or something similar, but they are EXPENSIVE items, especially considering that what I intended to build had even more features (and was, on the whole, cheaper).

Project Criteria

What do we want the project to do?

1. Manage the sourdough starter's temperature
2. Tell us whether it's strong enough for Panettone

So, I need:

- something to track the height of the starter inside the jar (this will tell me whether the starter has tripled in height in four hours)
- something to track the temperature of the starter inside the jar
- something to heat or cool the jar to ensure it always remains at around 27 °C,
- something to notify me of important

events (e.g. the four-hour mark)
- a GUI so I can reset the sourdough starter after a feed
- a GUI so I can track key variables (temperature, growth) remotely

I wanted to utilize component-based design principles so that I could always isolate issues to one area. Whilst this project could be easily condensed into a single board, I wanted to create something that tested interfacing between different systems, giving me the knowledge to undertake more complicated projects in the future. As such, I decided upon:

- **Sensors**: A Raspberry Pi 3B+ to manage the DHT and (laser-based) distance sensors and transmit that data wirelessly (to the Particle board, see below) and via USB serial (to the Arduino).
- **Heating / cooling**: An Arduino UNO to turn on the heat pads and fan when required, to maintain the temperature of the starter.
- **GUI / notifications / rise**: A Particle Argon [1] board to give the user a remote GUI, an LCD GUI, and provide alerts about significant events.

Figure 2 shows a quick diagram of the system architecture.

Webhooks / IFTTT

Everything should be able to talk to everything else. I didn't have significant experience in this field, and I felt that the IFTTT platform [2] provided the simplest and easiest way to get this communication happening. So, I created an IFTTT account and then generated an applet with a webhook as the 'IF.' Essentially, **IF** the webhook receives a web request with a JSON (**J**ava**S**cript **O**bject **N**otation) payload (i.e. a request from the Raspberry Pi), **THEN** it triggers a Particle function. The Particle can then use the JSON data in whatever way it needs to.

The website PiMyLifeUp [3] helped me tremendously. Most importantly, the webhook will give you an address of the form:

```
https://maker.ifttt.com/trigger/
tempsent/json/with/key/XXXXXX
```

Where XXXXXX is your personal key. All we need to do then is use that URL in our code to trigger this applet by posting at the appropriate time.

The second IFTTT is a connection between the Particle and the phone - this is quite simple. The **IF** waits for the Particle to publish an event of a specific name, then sends a notification to the IFTTT app on the phone if and when received.

The Raspberry Pi

Now, let's get all the sensors working, and the RPi talking to everything. To connect the circuit up:

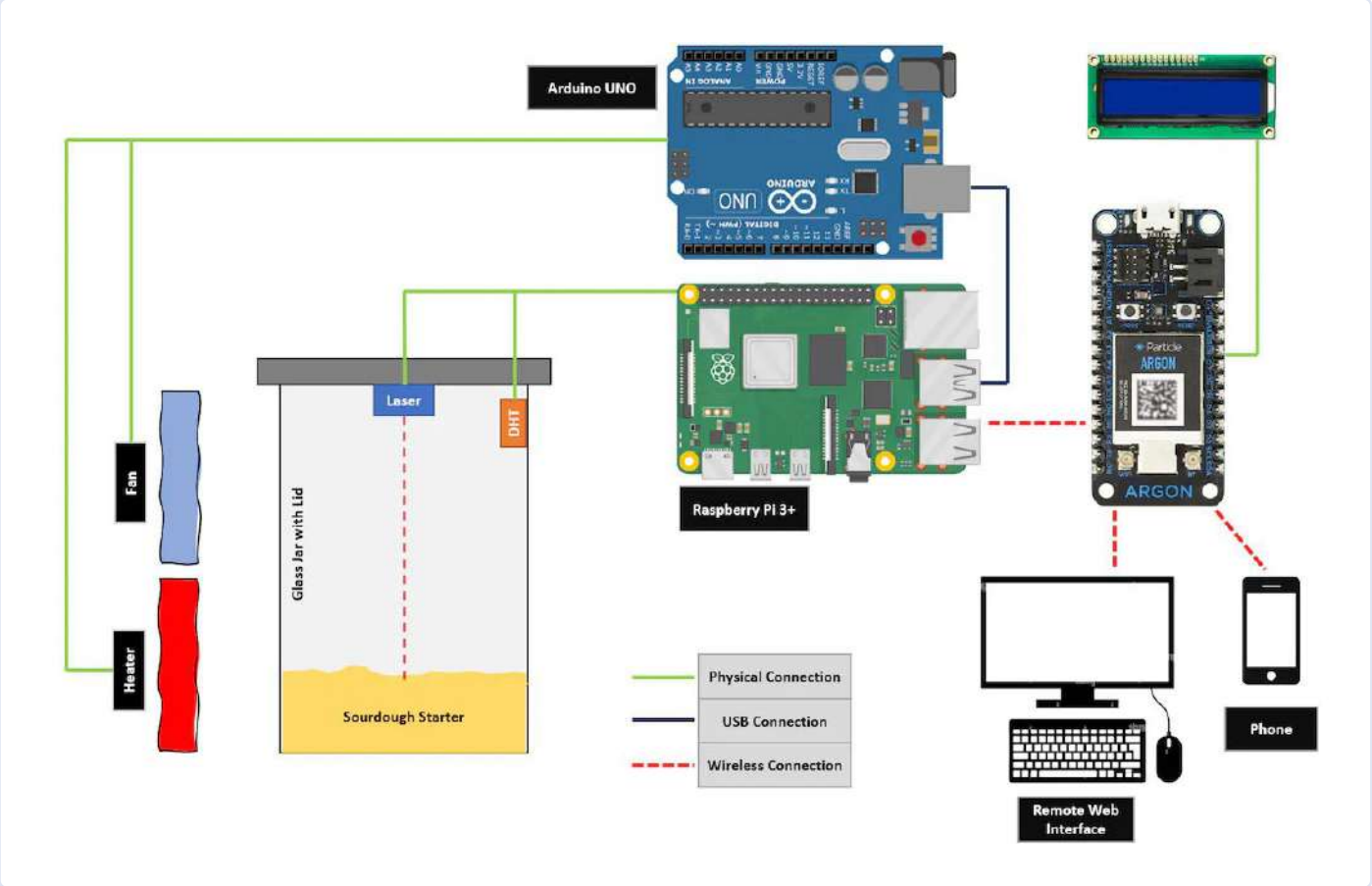


Figure 2: System architecture diagram.

- **DHT**: Connect power pin on DHT to RPi 5V, the second pin to RPi GPIO4, and the fourth pin to RPi GND. The third pin on the DHT sensor is unused. Note: It would have been preferable to use a DHT22 for accuracy, but mine was faulty and I replaced it with a DHT11 for the final product.
- **VL53L1XLaser**: Connect Vin to RPI 3.3V, GND to RPi GND, SDA to RPi GPIO2, and SCL to RPi GPIO3. This enables serial communication between the Raspberry Pi and the Laser.

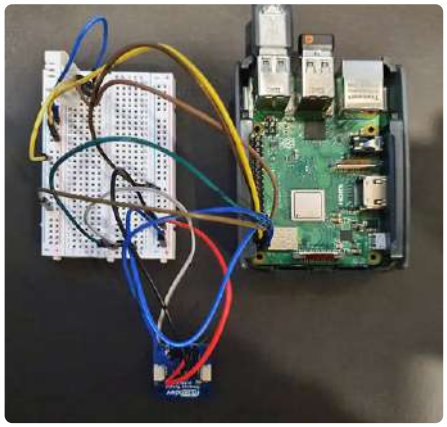


Figure 3: Test setup for the laser and temperature modules.

I note here that I chose a laser to measure distance, rather than an ultrasonic sensor, for a few reasons:

- Accuracy – laser-based measurements are much more precise.
- A laser is more reliable, especially since the surface of the dough is soft.
- I don't like annoying my cats with constant ultrasonic noises all day.

Figure 3 shows my test setup for the laser distance-measuring and temperature modules; the connections should be made as in **Figure 4**. Net ports are shown in the schematic to improve the wiring's visibility. I note that the schematic does not show the USB connection between the Arduino and the Raspberry Pi, but that is a critical component for communication.

Now, code time. I created a new Python file on the Raspberry Pi and started editing. I used the Python IDE, Thonny [3], as I find it really easy to use. Let's start with the imports:

- **Adafruit DHT** [5] in order to use the DHT sensor
- **PiicoDev-VL53L1X** from Core Electronics [6] for the laser
- **Time / Sleep**
- **Requests** to use the webhooks and communicate wirelessly
- **Serial** for communication via USB with the Arduino

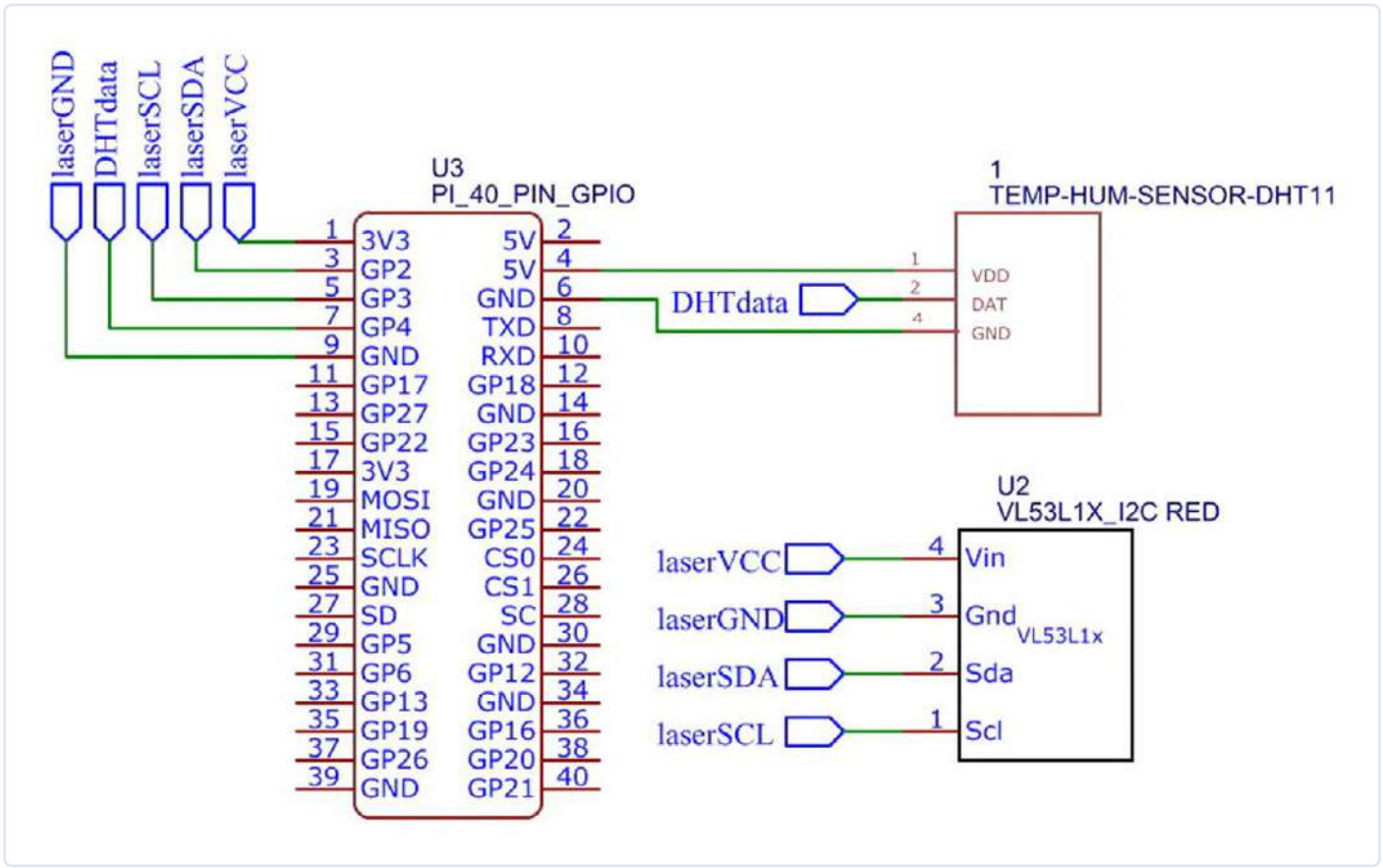


Figure 4: The connections made in Figure 3.

A Few Setup Items

Flags were used to signal the Particle that the fan or heater are on. The 'airmessage' is serially transmitted as an integer to the Arduino to 'do something.' The serial port was set up at the correct address by checking it in the terminal:

```
ls /dev/tty*
```

I created an 'average readings' method for robustness - this takes five consecutive readings and averages them out before reporting back to webhooks / serial, etc. This ensured single erroneous readings didn't mistakenly trigger events. It also included a fault-management code check that sends the user a notification if 100 faulty readings are observed.

There are many comments in the main loop, mainly used for debugging purposes. This loop gets the average and then decides what to do:

- If the temperature is less than 24 °C, turn on the heater.
- Allow temperature to rise to 27 °C, then turn off the heater.
- If temperature > 30 °C, turn on the fan.
- Allow temperature to drop to 27, then turn off the fan.
- While the temperature is in the range of 24 ... 30 °C (the optimal zone for the starter), no device is active.

This results in the Python script *PiCodepy*, a part of the software, which can be downloaded for free at the article website [7].

Arduino

The Arduino setup is more challenging from a power supply perspective, but thankfully, my dad is an electronics expert. The fan needs 12 V, which is far more than can be provided by the Arduino. Further, the heat pads take 5 V but have a resistance of 6 Ω. The required current is therefore close to 1 A – way above what an Arduino pin can support. Therefore, an external power source supplies the power needed to the fan / heater when triggered by the Arduino.

The setup:

- **Heater**: Connect Pin 12 of the Arduino to one side of the relay coil, and the other side to Arduino GND. Connect +5 V of the power supply to the common rail of the relay. Connect the positive pole of the heat pads to the 'always off' side of the relay.
- **Fan**: Connect Pin 10 of the Arduino to one side of the relay coil, and the other side to Arduino GND. Connect the +12 V of the power supply to the common rail of the relay. Connect the positive pole of the fan to the 'always off' side of the relay.
- **GND**: Connect the GND of the heat pads and the fans to the GND of the power supply.
- **Arduino**: Connect a 9 V battery (or power supply) to the Arduino. Only the Arduino is powered by the battery, NOT the Raspberry Pi.

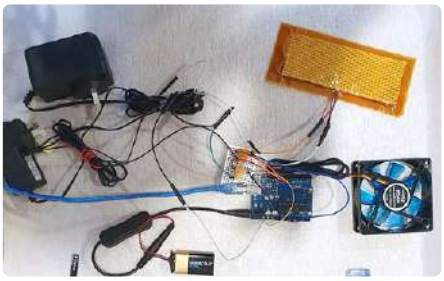


Figure 5: Quick-and-dirty system setup.

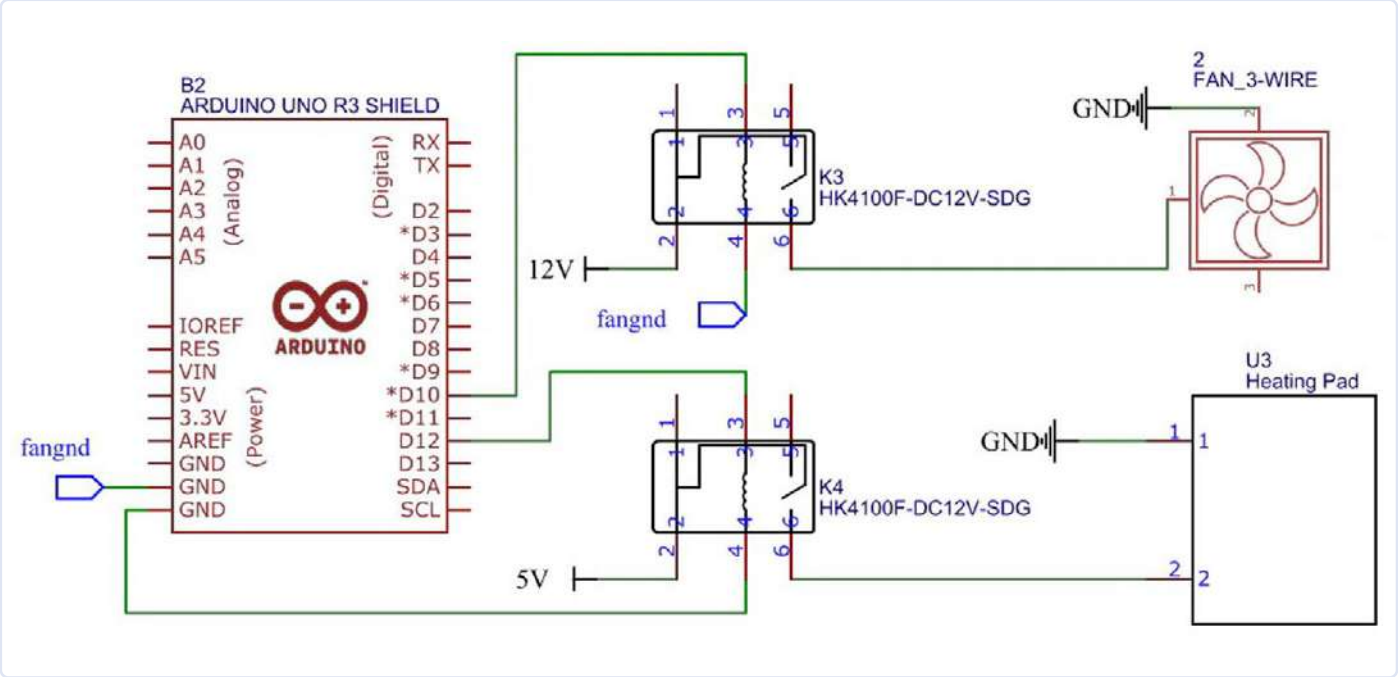


Figure 6: Connections around the Arduino board.

Figure 5 shows the experimental setup and **Figure 6** the circuit of the system around heating and fan. The Arduino gets data from the RPi via USB (serial). The messages are encoded in UTF-8. All we need to do is get the serial code and perform the relevant action depending on that code. This is managed via an `if` statement in the main loop of the code below. The website Automatic Addison [8] was very helpful for this purpose. The Arduino code can be found as a file at [7].

Particle Argon

The Particle Argon plays a pivotal role in communication between the user and the code plus devices. The LCD, potentiometer, and the LEDs need to be connected to the board. The LCD can use I²C, but I had to connect some more lines (**Figure 7**). The Hackster website [9] helped me to do it right.

I then connected two LEDs: A green one for 'starter is good!' and red one for 'starter is not good.' These were connected to Argon Pins D6 and D7, and then grounded via a 220 Ω resistor. After adding the potentiometer, the Particle subsystem was ready (**Figure 8**). The complete schematic can be found in **Figure 9**.

The Particle code [7], generated within the Particle IDE, is the longest and most complicated code, as it ties everything together. Walking through:

- Creation of the `lcd` object, empty strings, initialization as necessary.
- Hard-coding of the initial jar height and dough starting height constants, as these are known and fixed values.

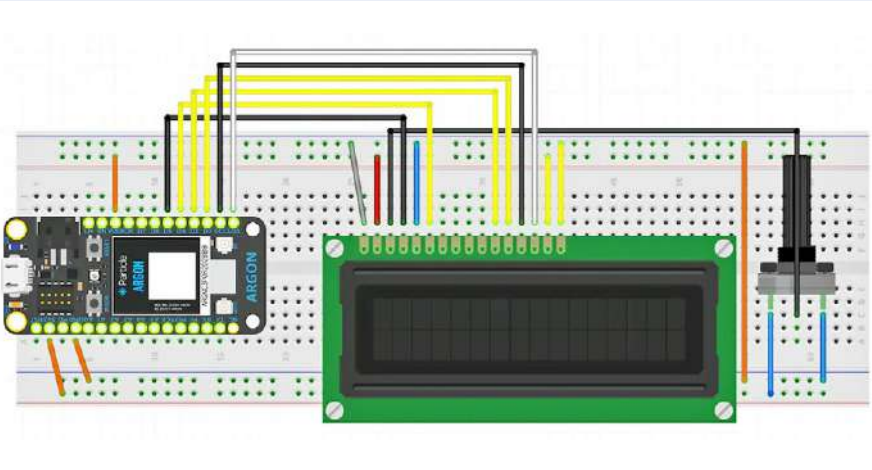


Figure 7: Display board and Particle board.

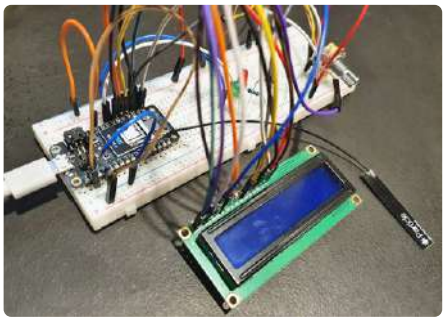


Figure 8: Particle board setup.

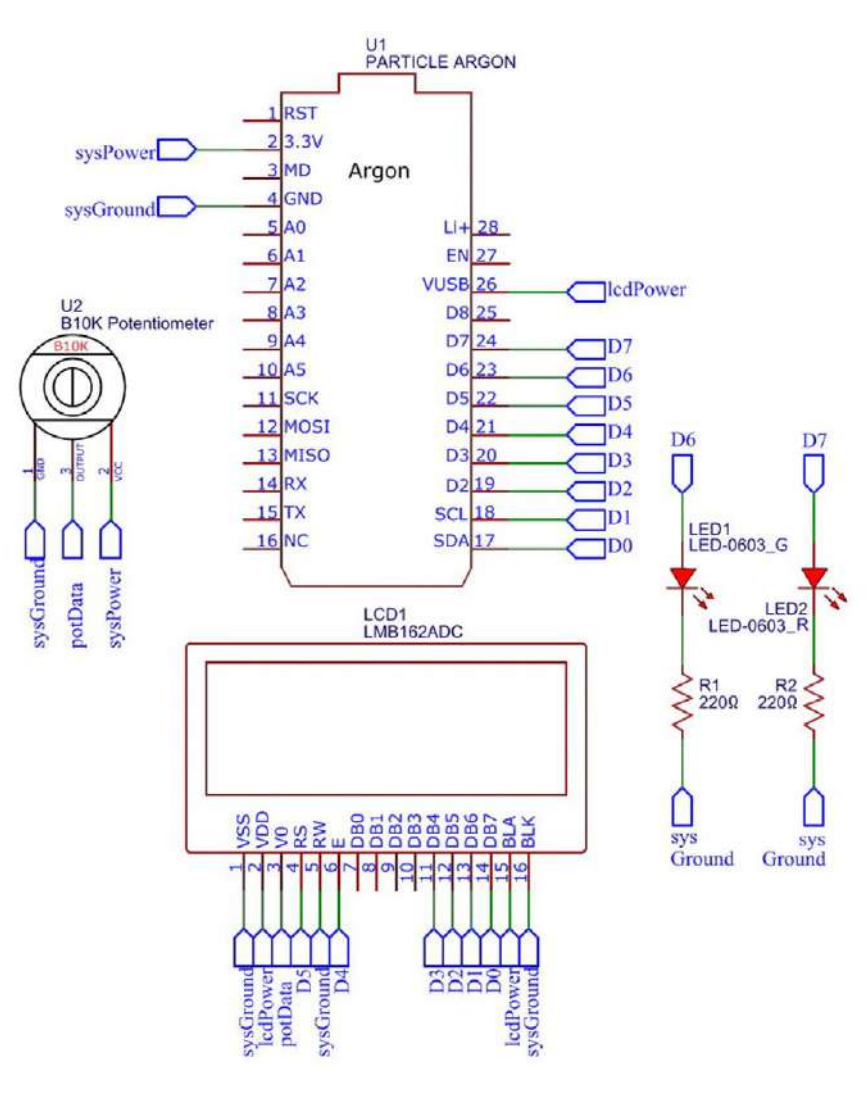


Figure 9: Schematic of Particle and components.

- Creation of the `resetbutton()` function. This is activated remotely through HTML to reset the timer once the four hours has expired (and the results are clear).

The setup

- Clear the LCD and display an intro message.
- Initialization of the two functions that interact with IFTTT. `displayTemp` runs the `tempDisplay()` function and `reset` runs the function that resets the button.
- Recording the time at the start.
- Turning off the LEDs.

The loop

- There is a ten-second delay so that the LCD screen is not bombed with continuous redraws.
- Printing the current commands (cmd1 and cmd2). These are set in the `tempDisplay()` function.
- Calculation of the growth of the starter.
- If the timer surpasses four hours, a new message with the final outcome appears on the display.

The tempDisplay() function

This is where the JSON data is received from the webhook mentioned earlier. This is all the critical distance data from the Raspberry Pi, transmitted wirelessly to Particle. Unfortunately, I could not get JSON to parse so that it was easily accessible. Instead, I just took it as a string and broke it down into substrings, which was very simple. I ensured the RPi formatting rules were strict so that this would always work correctly. The JSON contained the following data: Distance, Temperature, and whether the fan or heater was on. This data was used to manage both the HTML and the LCD screen. This code can also be found at [7].

HTML

HTML was used to enable remote access to the system. This let me reset the timer when the four hours are expired, as well as monitor the temperature and growth from anywhere where I have internet access.

The code is too long to print here, but we can show a few functions. Most of the code is taken from the Particle tutorial [10]. The HTML's key functions were:

Listing 1: The particle.getEventStream() calls

```
particle.getEventStream({ name: 'tempEvent', auth: sessionStorage.
  particleToken }).then(
{
    function (stream) {
        console.log('starting event stream');
        stream.on('event', function (eventData)
        showTemp(eventData)
    });
});

particle.getEventStream({ name: 'growEvent',auth: sessionStorage.
  particleToken }).then( function (stream)
{
   {
    console.log('starting event stream');
    stream.on('event', function (eventData)
    showGrow(eventData)
   });
});

particle.getEventStream({ name: 'timeEvent', auth:sessionStorage.
  particleToken }).then(
    function (stream) {
    {
       console.log('starting event stream');
       stream.on('event', function (eventData)
       showTime(eventData)
    });
});
```

Listing 2: resetControl() function

```
function resetControl(cmd) {
 //const deviceId = $('#deviceSelect').val();
    $('#statusSpan').text('');
    particle.callFunction({ deviceId: 'X', name: 'reset', argument:
  cmd, auth:sessionStorage.particleToken}).then(err);
    function (data) {
        $('#statusSpan').text('Reset completed'); },
    function (err) {
         $('#statusSpan').text('Error calling device: ' +
} );
}
```

- Display live temperature data (by subscribing to an event stream).
- Display live growth data (by subscribing to an event stream).
- Reset the timer (by calling a function when pressed).

The `particle.getEventStream()` calls (**Listing 1**) show how data is extracted from the event stream. The `resetControl()` function (**Listing 2**) is triggered by pressing a button (Particle ID removed).

Action Shots

Figure 10 shows some photos of the LCD display at various points. The last part, *d*, shows the notification of *c* on a smartphone. **Figure 11** shows the complete robot system (a), the box with the electronics inside (b) and the boards and wires on top of the big box (c). And, if you ask yourself how this starter will look, **Figure 12** demonstrates the fierce growth of the starter between before and after the four-hour session.

Conclusion

That's it, everything is working, and all synced up. Now, it's time to automatically produce the first sourdough starter with this dough robot. And, of course, when the starter is strong enough, get baking and eating!

220416-01

Figure 10: Different content on the LCD (a, b, and c), plus the smartphone message after four hours.

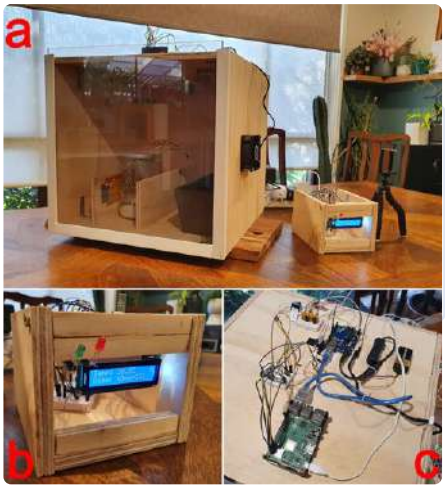


Figure 11: The complete robot (a) with control box (b) and some boards and wires (c).

Figure 12: Sourdough starter before (a) and after reaching the correct height four hours later (b).

About the Author

Daniel Fantin is a second-year Computer Science student at Deakin University. He has a passion for food and technology, so combining Italian cooking and robotics comes as a natural progression.

Questions or Comments?

If you have technical questions, feel free to contact the Author through Hackster [12] or the Elektor editorial team at editor@elektor.com.

WEB LINKS

[1] Particle Argon: https://docs.particle.io/argon/
[2] IFTTT: https://ifttt.com
[3] PiMyLifeUp: https://pimylifeup.com/using-ifttt-with-the-raspberry-pi/
[4] Thonny IDE: https://thonny.org
[5] Adafruit DHT: https://github.com/adafruit/Adafruit_Python_DHT
[6] PiicoDev_VL53L1X: https://elektor.link/piicodevgithub
[7] Project software: https://www.elektormagazine.com/220416-01
[8] Automatic Addison: https://elektor.link/rpiarduino2way
[9] Hackster website: https://elektor.link/particlehackster
[10] Particle tutorial: https://elektor.link/particleapitut
[11] Project at hackster.io: https://elektor.link/hacksterpanettone
[12] Daniel Fantin at hackster.io: https://www.hackster.io/danielfantin

Supporting **Arduino Resellers**

This guest-edited Arduino Edition of Elektor Magazine was made possible with the support of these members of the Arduino reseller community.

Check them out for your Arduino-related needs.

GOTRON
AALST GENT HASSELT

www.gotron.be

HELLAS digital

www.hellasdigital.gr

TINYTRONICS

www.tinytronics.nl

Paradisetronic.com

www.paradisetronic.com

Techni Science.

www.techniscience.com

WHADDA
MADE BY VELLEMAN

www.whadda.com

KUBII

www.kubii.fr

GO TRONIC
ROBOTIQUE ET COMPOSANTS ÉLECTRONIQUES

www.gotronic.fr

Check out any one of these resellers for your Arduino-related needs.

Space Invaders with Arduino

Figure 1: The Space Invaders project.

By Colin Dooley (Spain)

Would you like to combine your love for gaming with your passion for Arduino? Here is an innovative example of how to use an Arduino to build a compact Space Invaders machine.

I grew up programming video games in the 1980s, so when I got my first Arduino, I naturally wanted to program video games on it. Arduinos have the same sort of memory and CPU limitations as the early home computers, so I immediately felt right at home with it. Things have advanced since then. I now have an IDE and a high-level language, but the basic steps and optimization techniques for programming games on an Arduino are the same. There are hexadecimal numbers, a lot of bit shifting, time spent trying to pack everything into tiny amounts of RAM, etc. My old assembly language skills were still very handy; I ended up doing a lot of code disassembly to see what the compiler was doing wrong (quite a lot!) and tuning my code to get the best results.

The game I chose to program was Space Invaders (**Figure 1**). It's an iconic game, and not too complex to program. The idea I had in my head was to use an Arduino to create a tiny Space Invaders machine that was as close as I could get to the original ("pixel perfect").

I found a website online that had a complete disassembly of the Space Invaders ROM and used that as a reference for the game logic. It was a lot of work to get all the details right, and I learned a lot about the game in the process. (It turns out there are some bits of old/dead code in the ROMs that give clues about the development.) The web site is www.computerarcheology.com/Arcade/SpaceInvaders/.

I can safely say that the result is as close to the original game as it's possible to get without running the original ROMs on an emulator, which can't be done on an 8-bit Arduino! All the logic is there. If you're an expert player, you can count your shots to get maximum points from the flying saucers, etc. The only real change from the original is that I made the invader bombs fall a little bit slower because it was hard to play the game on such a tiny screen.

Graphics

The original Space Invaders game has a display with 224×256 pixels. My first job was to find the smallest possible screen with that many pixels. (It has to be "pixel perfect" – remember!) After some searching on AliExpress, I found a suitable screen and went to work.

All the graphics for the game were created the "old fashioned" way with pencil and graph paper (**Figure 2**). The graphics are drawn as pixels, converted into hexadecimal numbers (I can still do that in my head!), and then typed them into the program. You can find them in the file graphics.h in the source code (www.thehighprotondiet.com).

The screen uses an SPI, and I was initially worried that this would be a bottleneck for a video game that runs at 60 fps (**Figure 3**). Space invaders doesn't update many pixels every frame though, so it turned out not to be a problem. After a few weekends, I had the graphics working. It only used about 10% of the Arduino's CPU to run the graphical part of the game. The Arduino's CPU is much more

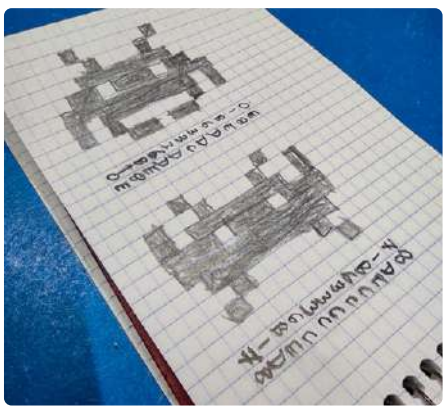


Figure 2: I created the graphics with a pencil and graph paper.

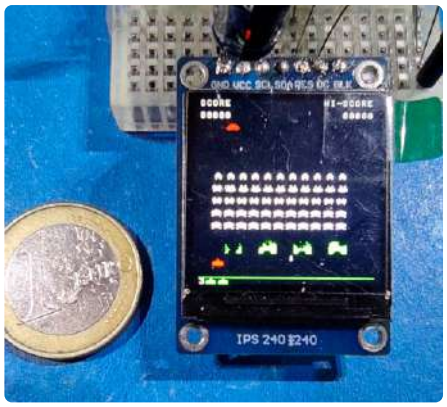


Figure 3: The screen is compact.

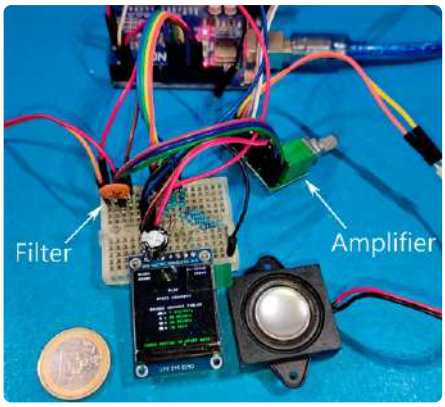
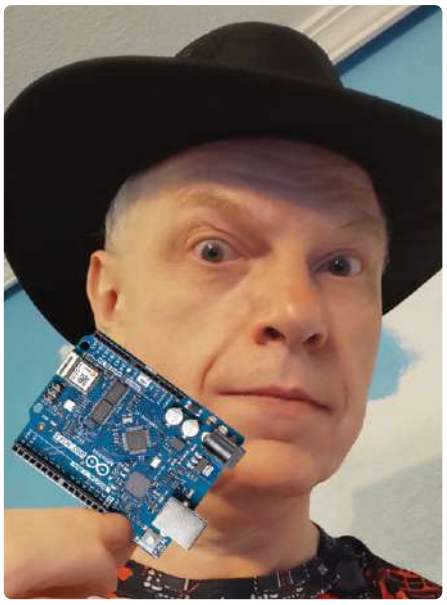


Figure 4: The design includes a simple low-pass filter, a little audio amplifier, and a speaker.



powerful than the CPUs found in a 1980s-era home computer. As it turned out, the sound needed much more CPU power than the graphics.

Sound

I found the sound samples for Space Invaders as part of the MAME emulator. The only problem was that they were 49 KB big, and the Arduino only had about 20 KB of memory left after the program and graphics were uploaded – they didn't fit! I ended up converting the sounds to 6-bit samples instead of 8-bits and packing them tightly in memory. There was one sound which simply wouldn't fit in memory (the flying saucer death sound), so I substituted the player death for that. I hope nobody will notice. I didn't plan it this way, but using 6-bit sounds has another huge benefit: Space Invaders has four sound channels, and you can add together four 6-bit numbers to get one 8-bit sound. This means that the sound mixing can be done optimally on an 8-bit CPU like the Arduino's ATmega328.

The final 8-bit sound is played back via a PWM signal generated by the ATmega328's hardware timer #2. The signal goes through a simple low-pass filter made with a resistor and capacitor and then into a little audio amplifier and a speaker I found on AliExpress (**Figure 4**). The result sounds great! The sound from the machine will definitely bring back a lot of memories to anybody who lived through that time.

Game Controls

For the controls, I found a really tiny joystick on AliExpress (**Figure 5**). The joystick was too small, and the movement was very stiff to use, so I 3D-printed a small extension for it. The end result looks really good and is much easier to use.

The Cabinet

I recently bought a laser cutter for my home workshop, so the cabinet is laser-cut MDF using a plan I found on the Internet (Figure 1). I modified it slightly to add graphics and cutouts for the screen and joystick. It makes a good base to let me figure out how to do the joysticks and internal electronics.

The Future

I'm not finished with this project, yet. The cabinet shown here is a very generic one, not a true Space Invaders machine. The original machine used an internal mirror system to overlay the invader graphics onto a colored background. Combined with clever internals, it gave a multilayered 3D effect.

I plan to reproduce that system soon! You can find more information and source code for this project on my website: www.thehighprotondiet.com.

220557-01

Figure 5: The tiny joystick.

About the Author

Colin Dooley grew up programming 8-bit home computers in the 1980s. His first real job was at a computer games company in Sheffield, England, in the mid-1980s during the time of the Sinclair Spectrums, Amstrads, and Commodore 64s that make up much of the "retro" scene today. According to Colin, "It's an honor for me to be featured in a special edition of Elektor magazine. Magazines like Elektor were one of the few resources we had for learning about computers and computer programming in the pre-Internet age."

Questions or Comments?

Do you have any questions or comments relating to this article? Feel free to contact the team at Elektor at editor@elektor.com.

Art with Arduino

Inspiring Insights from Artists and Designers

By David Cuartielles (Arduino)

Magic seems to happen when artists incorporate Arduinos in their work. Arduino co-founder David Cuartielles recently interviewed three forward-thinking artists about their passion for art and the use of Arduino solutions in several projects.

We first created Arduino to support our art and design students in the creation of digital devices and interactive installations. We believed (and still do) that people should be capable of programming and building their own smart artifacts by simply adding some components together and writing easy programs. The art and design community was in need for new simple tools that could address the craft of electronics, and Arduino became a great tool to support the creation of thousands of new art pieces and experiences. We ran many workshops for artists at museums, cultural centers, galleries, and faculties of arts at different universities. After the creation of the first Arduino Serial board in March 2005, we ran a workshop at the Centro Cultural Code Duque, in Madrid's Medialab, Spain. In 2006, barely one year after the creation of Arduino, we were invited to host workshops at Ars Electronica, the most important electronic arts festival in the world. The use of Arduino in arts expanded rapidly. As a consequence, in 2014, the Arduino Diecimila was added to MoMA permanent collection. [1] There are many stories of artists using Arduino in their work, which made it really hard to decide who could represent the Arduino-based electronics scene in a broad sense. I recently interviewed José Salatino, Kelly Heaton, and Jacob Remin and put their stories together in this special section devoted to art and technology. Here you will discover what painting robots, touch synths, and Buddha-enhanced control circuits have in common.

José Salatino

Painting with Arduino: José Salatino

José Salatino is a maker and painter living in Spain who has been experimenting with different technologies and techniques. We met up with him to talk about his painting robot and to get to know more about the specifics behind this technology.

David Cuartielles: Hey, José. Can you tell us something about yourself? Where are you based, and what are your interests?

José Salatino: Hello, David. Thank you for this opportunity to show my project and tell what my passions are. I have been living in Barcelona for a year, very close to the Sagrada Familia, and I am delighted to live here. It is a wonderful city wherever you look. In fact, the first time I showed my project was at the Barcelona Maker Faire in 2017. An unforgettable experience. One of my interests, as you mention in the introduction, is painting (**Figure 1**). I've been doing it as a hobby for a long time, and lately I'm trying to do it

in a more professional way. The other great interest is the "inventions," which I have been able to materialize thanks to the appearance of Arduino and the maker movement.

Cuartielles: The combination painter-maker is an interesting one. Can you tell us some more about how you got introduced to painting?

Salatino: My beginnings in painting were intuitive. I do not have an academic background in this field. One day, I began to paint the portrait of an Argentinian singer that I really like, and the result seemed acceptable to me, to the point that I framed it and delivered it to him personally. I liked this experience a lot and repeated it with several artists, including Joan Manuel Serrat and the famous Argentinian pianist Martha Argerich. And since then, I haven't stopped doing it.

With the arrival of Arduino, which gave me the possibility to design and build very sophisticated machines at an affordable cost, the idea of making a painting robot inevitably arose.

Cuartielles: What brought you to Arduino? We have met in the past, and we have seen your robot in action. Could you tell us what kind of operations it performs?

Salatino: Another of my passions has always been electronics. And from a very young age I tried to make circuits to automate things. But at that time the only possibility to do it were microprocessors, which were very expensive, difficult to obtain and very difficult to program and assemble. You had to have in-depth knowledge and a well set up laboratory to be able to reach a reasonable result. Arduino solved all these problems, and after many years of giving up trying to automate anything, Arduino brought it to my fingertips. Two days after buying my first Arduino UNO, I had already managed to build a differential robot that reacted to light and some other things. It was magical!

As for my painter robot, what he does is to paint a portrait with brushes and acrylic paints. Based on a digital image, and after a process carried out by several algorithms, I obtain the necessary information for the robot to carry out its work. That is: define the color to use, make the corresponding mixture, define the position and shape of the brushstroke, and finally apply it. There are other complementary tasks such as changing, cleaning, and drying the brushes.

Cuartielles: Does your robot have a name? You used it to paint portraits. Is there a reason why you went for these kind of works and not a different one?

Figure 1: Art created with Salatino's painting robot. (Source: José Salatino)

Salatino: In the Maker Faire that I have participated in, the official name has been: *Portrait Painter Robot Project* (**Figure 2**). As a painter, I have always preferred the portrait to another theme, and I think that this has defined what my robot likes, but I must also confess that there is also a scientific reason. The human brain has a specific capacity to recognize the human face, that is, if there are deficiencies in the portraits that my robot paints, the observer's brain will make the necessary corrections to perceive what he is seeing as the face of a person.

Cuartielles: There are other painting robots out there. As I understand it, they differ from one another in the kinds of painting styles they can perform, the way they mix colors, etc. Can you tell us about the main features of your design?

Salatino: It is true; there are many painting robots. In fact, I participated a few years ago in a contest for painting robots (RobotArt 2018) obtaining an honorable tenth place, competing with groups from famous universities and scientists with access to great technical and economic resources. Without a doubt, what makes my robot different from others is the ability

Figure 2: Portrait Painter Robot Project. (Source: José Salatino)

Figure 3: Salatino and Cuartielles discuss the robot and a painting. (Source: José Salatino)

to automatically mix colors. In fact, it was only after solving this issue that I was encouraged to make my first painting robot.

Using the three primary colors, the three secondary colors, black, white, and four shades of gray, it is able to generate more than 150 different shades, with which any color image can be reproduced, closely matching the original image.

Cuartielles: You got very engaged in the maker community and got to travel to different Maker Faires. As a matter of fact, we met at Maker Faire Rome (Figure 3). How was that experience? How did people receive your robot?

Salatino: Yes, I have been able to participate in several Maker Faires, and the experience has always been incredible. I have several blue ribbons given by the organizers in recognition of my projects, and they are my most precious trophies. I have also had the acceptance of the public who have expressed their opinions to me from their various perspectives.

At a Maker Faire, you meet a wide variety of audiences. There are many makers who are attracted by the purely technical aspect or to see how you solved a particular problem. There is also a lot of public that is not connected with the maker movement and wants to see what it is about, and simply tells you if they like it or not, or if they are struck by something in particular that generally has nothing to do with the technical issue. But there is also a group of people who want to know the motivations for someone to do something like that, and with those people I like to talk. Finally, my project is fully immersed in a very conflictive field that is the relationship between art and technology, and I have also had very interesting conversations about this.

Cuartielles: Going deeper into the technology behind your robot. How is it powered? Which board are you using? What are the other parts you used?

Salatino: I have built two versions of the painter robot. The first one, the one you saw in Rome, was based on the electronics of a 3D printer (Arduino Mega 2560 R3 + Ramps 1.4 and + 4 NEMA 17 stepper motors + Marlin Firmware). Then I built a second version with the same concept, but I got NEMA 23 motors with built-in drivers and had to make some changes to the electronics by replacing the Pololu drivers with a PCB that allowed me to connect the Ramps 1.4 to the new drivers.

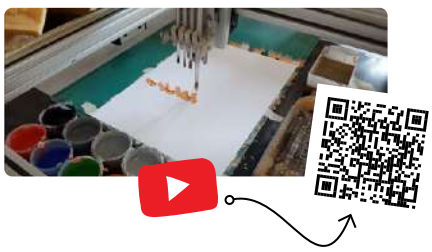
As for the mechanics, the first version was built with aluminum profiles, mechanized with great effort with the few tools I have. The second version is based on industrial linear movements mounted on a wooden frame that I designed in such a way that it could be disassembled to be able to transport it.

Cuartielles: Finally, can you tell us where people can reach out to you? Do you have a website or social media accounts where people could find more about your projects?

Salatino: Yes, if you google "Jose Salatino," you will find several entries about my activity as a painter and as a maker. I recently set up a website to show my paintings in a more professional way. The address is: www.josesalatino.com. You can see my most important maker projects at this address: www.hackaday.io/josesalatino. I currently live in a flat where I can't set up a workshop to work, so I'm not doing many things, just playing with my machines. (Well, I also work eight hours a day in a factory to pay my bills.)

I want to take this opportunity to say that I would love to participate in new projects (the crazier the better), in which art and technology interact. There is still a lot to do, and I would like to be part of it. You can contact me at through email (jvsalatino@gmail.com), Instagram (@josevicentesalatino), Twitter (@jvsalatino), and YouTube (jvsalatino).

Watch the Portrait Painter Robot in action!



Arduino and *The Tree of Life*: Kelly Heaton

Kelly Heaton is a well-known artist to *Elektor* readers. She made it to the cover of Elektor's Summer Circuits 2022 issue, and we had the chance to get to know extensively about her initiation to electronics and her current art practice. There is one piece of hers that features an Arduino board. At first sight, *The Tree of Life* is a central controller for a larger installation including multiple of Kelly's famous birds. We reached out to learn more about this piece and how it works.

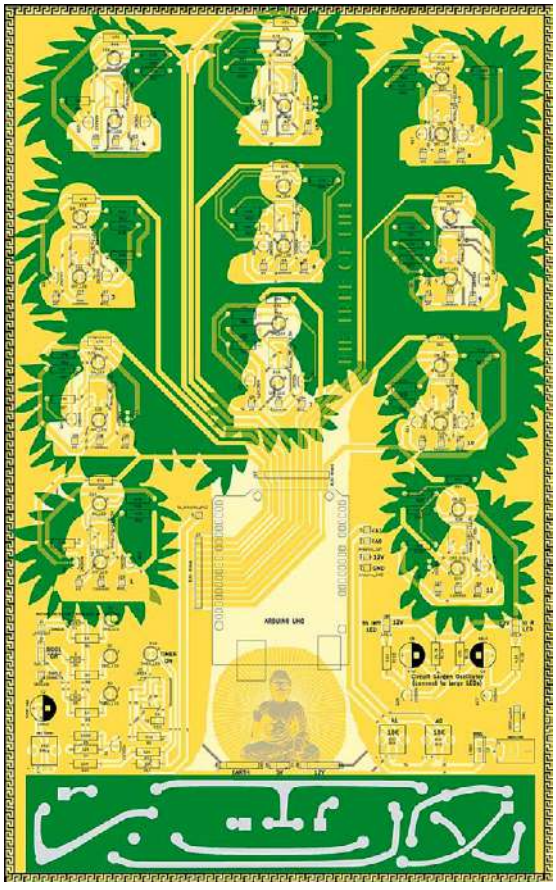
Kelly Heaton

David Cuartielles: Hey, Kelly. Thanks for making yourself available for this interview. Can you tell us briefly about yourself for those readers who haven't yet read our previous interview with you? [2]

Kelly Heaton: Thanks for inviting me! I build artistic and philosophical circuits to visualize the flow of energy in our conscious universe. Analog electrical engineering is key to my practice because it's the closest I can get to understanding and sculpting electricity as a raw creative medium. Most people think of electrical engineering as logical and pragmatic, but for me, it's a way to ponder the energetic nature of existence.

Cuartielles: Your work includes a whole ecosystem of different boards, models of components, clothing ... I would like, however, to focus on *The Tree of Life*, a thrilling PCB featuring an image of Buddha on it, a series of relays, an Arduino UNO R3, and connectors for sensors (Figure 4). Can you tell us about the concept behind it? How does it interplay with the rest of your work?

Heaton: *The Tree of Life* is an ancient archetype of interconnectedness. It is one of my favorite symbols and a recurring theme in my work. In Western mythology, the sacred world tree connects all of creation in a powerful symbol of ecological wellness. In Tibetan



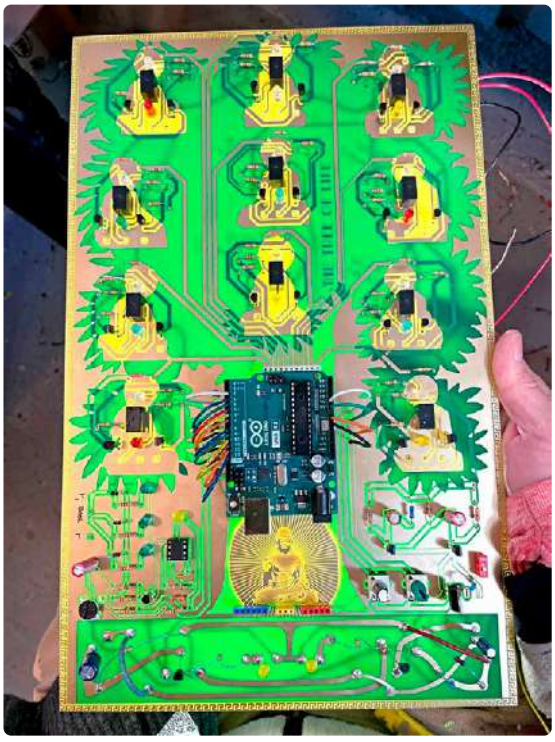


Figure 4: On the left, the design file for the The Tree of Life in the Circuit Garden, 2022. On the right, the PCB with an Arduino Uno integrated. (Source: Kelly Heaton)

Figure 5: The Tree of Life PCB on display in the Circuit Garden. (Source: Kelly Heaton)

Buddhism, Gautama Buddha achieved enlightenment beneath the Bodhi Tree, and the related Refuge Tree is a lineage diagram of divine teachers and spiritual aspirants. I combined these themes in an aesthetic circuit with electronic, natural, and spiritual energies. As part of my Circuit Garden (2022), *The Tree of Life* is presented inside of a sculptural integrated circuit as the soul of the ubiquitous "black box" microprocessor. In my design, I used an Arduino UNO to programmatically orchestrate a soundscape of my birdsong and cricket generators. I'm not a skilled Arduino coder, so I posted a request for help on your [the Arduino] forum and member John Wasser generously provided me with a pseudorandom sequencing routine. The Arduino community is amazing! I am so grateful when people share their knowledge. Technical complexity can be a frustrating barrier to artistic expression because it's hard to cultivate right and left brain intelligence simultaneously. The pursuit of consilience is why *The Tree of Life* is such a resonant symbol for my art.

Cuartielles: There is the obvious question that I have to ask. In your work you focus on analog technologies, but Arduino is basically a digital microcontroller used to schedule and execute tasks in sequence. What brought you to Arduino and why did you decide to use an Arduino UNO at the core of *The Tree of Life*?

Heaton: I focus on analog electronics for three reasons: (1) digital simulates life, whereas analog *is* life-like; (2) I want to understand the fundamental nature of electricity as a creative medium; and (3) analog electrical engineering holds tremendous potential for advancing human knowledge. That said, many practical tasks are easier with an off-the-shelf digital tool such as the Arduino. I had already built a pseudorandom analog sequencer in 2018 as part of my project, *Hacking Nature's Musicians*, so I knew the complications. I was on a deadline for my *Circuit Garden* installation and needed a sequencer that would function in a public space for three months. Plus, making electronic songbirds, crickets, butterflies, and plush circuit sculptures was keeping me plenty busy and challenged. Experimental analog electronics make great art, but they are not always practical, robust, reproducible, or quick to develop. I decided to use an Arduino for the sequencer because sometimes you just need to solve a problem. I chose my particular Arduino board because the specs worked, and I really like the name "UNO." We are all connected in one (uno) cosmic circuit.

Cuartielles: Browsing through the online documentation of this piece, I can see that there are four different parts in the PCB. One is the one dedicated to the control. That's where the Arduino is placed. Then there is a smaller IC, which I assume is a 555. There are a couple of potentiometers used as analog inputs to the Arduino. And finally, there is the bottom part of the PCB, which we could call the roots of the tree. Is that a fair depiction of the board? How do the different parts interact with each other?

Heaton: Understanding the function is easier if you see my 21-foot-long *Circuit Garden* (**Figure 5**) because that's what it controls. Basically, the Tree of Life orchestrates when my printed circuit birds and crickets are audible, like a Mother Nature puppeteer for an electronic garden chorus. Two remotely positioned motion detectors are used to trigger a 555 monostable, aka one-shot timer, with a time delay that is set by a potentiometer. When motion is detected, the 555 generates an active signal for somewhere between five to thirty seconds. This signal triggers the Arduino to run a pseudorandom sequencing routine for as long as the monostable remains activated. The Arduino UNO switches its output pins on and off for varying periods of time that can be adjusted by another potentiometer. Because Arduino can only sink a small amount

of current, I amplify the output signals and use them to trigger relay switches that you see in the eleven bodhisattvas of the tree canopy. The relay switches control the audibility of my remote animal circuits vis-a-vis long wires that run beneath the installation. When motion is no longer detected, the monostable eventually shuts off, and the Arduino goes quiet. At the bottom of the *Tree of Life* board, what you are calling the roots, there is an astable multivibrator with the same visual composition as my large *Circuit Garden* installation. These oscillating "roots" are a microcosm of the garden, a symbol for the secrets of the universe, and a reminder that everything is ultimately reducible to vibrating energy. My *Tree of Life* embodies the hermetic principles "as above so below," and "as within, so without."

Cuartielles: The top copper layer is a piece of art in itself. It is a tree where you can see the silhouette of eleven bodhisattvas on the leaves of a tree. The Arduino is fixed to the board above a meditating Buddha. You played with copper, substrate, and solder-mask layers to achieve three colors. Did you draw everything on a PCB design software? If so, which one did you use?

Heaton: Unfortunately, there is no single software with the cross-disciplinary tools to make art like this. It's a challenge with brain twisters and constraints, both in the design and manufacturing (but these challenges are also what make the art exciting because it's innovative). Here's how I do it in a nutshell: I lay out traces in KiCad using my schematic and a vector diagram of the composition that I draw elsewhere and import as a reference layer. When the functional circuit is finished, I export my production layers for editing in Adobe Illustrator and Photoshop. This is where things get beautiful and scary at the same time. It's not easy to keep the functional circuit intact using programs that don't know about electrical connectivity. I always think it will be easier than it is, but my brain goes into a different mode when I am working with imagery, and I tend to snip wires that I shouldn't. Anyway, with fingers crossed that my circuit still works (if it ever did in the first place, lol), I export my graphical layers as bitmap files and convert them to Gerber for production. I almost always need a second or third run to get my boards right because the errors can be electronic or visual or (often) both. For example, I unwittingly deleted most of the ground plane on my first *Tree of Life* board and had to rebuild it painstakingly by hand. I have inverted my solder-mask layer more than once, and board manufacturers will typically remove all silkscreen printed on bare metal unless you clearly tell them otherwise. There are many pitfalls to navigate. I wish there were a great PCB tool and manufacturing pipeline for artists, or at least some way to check my electrical nets after I modify them in another program. I would also like to have plug-ins for algorithmic modification of my nets, like space-filling curves and other topological transformations such as copper optimization. Last but not least, I want to meet some of the factory technicians who fabricate my boards because, especially as someone who does my own screen printing, I am astonished by their level of craftsmanship. Most people take printed circuit boards for granted, but I see highly skilled artistry with the potential for much greater expression. Circuit board printmaking remains highly experimental as a fine art form, but it is destined to become widely known. For now, it's really cool to electrify my artistic tableau and be involved in the invention of this new genre.

Cuartielles: A key question that I like to bring up when talking to artists working with open-source technologies is about the possibility of reproducing their work given that all parts are available. How do you feel about that? Have you thought about how to license your artworks?

Heaton: I make art out of curiosity, passion, and a desire to share with others. I publish many of my schematics in the public domain because I am indebted to everyone who has made their knowledge freely available to me, and my schematics reveal the magic and mystery that I want to discuss. I am honored when people build my circuits and make them their own. I don't want money or intellectual property to be a barrier to anyone's artistic expression. That said, we all need to earn a living. The works of art that I personally make by hand and sign are unique, rare artifacts and are priced accordingly. I am currently designing editions of my art that can be manufactured in larger quantities (not by me) and sold at a more accessible price. I am absolutely open to partnerships and licensing models, assuming they align with my ethos and artistic vision.

Cuartielles: Finally, can you tell us where people can reach out to you? Do you have a website or social media accounts where people could find more about your projects?

Heaton: Follow me on Instagram or Twitter @kelly_heaton. You can send me an email through my website www.kellyheatonstudio.com/contact.

Jacob Remin (Photo by Lotte Løvholm).

The Artist-Engineer: Jacob Remin

Technology is not just about bits and atoms. Contemporary technological jobs require having a sense of the community, being ready to collaborate with many different people, and having an open mind to crazy new ideas. No two careers are the same, but there's no doubt that some are more intricate than others. To bring an example of a consistent yet thrilling career in technology, I reached out to Jacob Remin, a Danish artist and engineer.

I first met Jacob in Istanbul in 2002 when a group of Scandinavian artists and designers I was part of landed in the city for a series of cultural exchanges with art galleries and universities. Back then, Jacob was a friend of a friend who, over the years, would become a student, a gallerist, a curator, but first and foremost, a respected artist and creative technologist. In this interview we will discuss projects, digital creativity, and even music videos … all powered by Arduino.

David Cuartielles: When you first came to the School of Arts and Communication at Malmö University (where I teach) in the late 2000s, you made a very funny project that tried to get us to think about our relationship to TV. Can you tell us more about it?

Jacob Remin: It was called *Workout TV*, a project I developed together with Martin Aggerbeck. We were Design Engineering students at the Danish Technical University but decided to take this class at Malmö University because our school had no courses in Arduino. The course required creating a critical design concept, something to make you reflect upon your own existence. We decided to make a humorist take on the concept "couch potato" and TV culture. For this we hacked a remote control to a TV so that it would automatically change channels if it detected that you had not been doing any exercise for a while. It featured an Arduino and a motion sensor, and if you didn't move while in front of the TV, it would randomly shift to a different channel. To explain what was going on we made a satire of a TV shop commercial explaining the concept. The project was later exhibited at Half Machine festival in Copenhagen.

Cuartielles: The Danish Technical University was not the last design school you attended?

Remin: First, I should mention that I was not at the DTU because I wanted to learn how to become an engineer in the traditional sense. I was interested in the possibilities that technology would offer for art production. I had been working as an artist before, doing video installations, film, music, making live performances, etc. The teachers at DTU were very kind to support my interests and let me take courses at other institutions to support our education.

Cuartielles: And then you moved to the Copenhagen Institute of Interaction Design to take a one year Master's course in applied technology to the design world. Can you tell us how it was there?

Remin: CIID is the opposite of engineering: everything moves fast, you have to go wild with your ideas, and follow a process of trial and error. You will break stuff and rebuild it through fast iteration cycles. I am using these learnings a lot in my own practice today. There, I had the chance to dive completely into electronics, and started making my own circuits. You introduced me to Eagle (the PCB CAD software), and I made my first printed circuit boards. I was impressed with how fast you could come close to "real products" by pairing custom PCB designs with laser cut enclosures and so I really fell down a hole here exploring these mediums.

Cuartielles: Let's talk a bit about that. At CIID, for your final thesis project, you designed a small synth using the ATmega328. What brought you to do that?

Remin: I have always done music. Back then, I was strongly influenced by 8-bit music and the existing portable music platforms. I liked the sound aesthetics of the scene, the portability, and the creative constraints of the technology. Microcontrollers were a perfect match for making a chip-tune sound platform. I wanted to build a musical instrument that could be performed on live by musicians in a typical jam situation. I called it CFO (Cheap, Fat, and Open), and it was a kind of 8-bit groove box. I was (and still am!) inspired by the work of Critter & Guitari, and they were really sweet to send me the source code to one of their instruments for me to study how it was made and make my own.

Cuartielles: Did this thesis project have a continuation? Did you jump into the business of making small synths? There is a market for these small, ultra-portable, music instruments with a character.

Remin: By the time I was done with my thesis, I was not interested in going into manufacturing and selling music instruments. I was much more interested in making art. However, the CFO kept on coming back to me. Somehow the project refused to die. When I was invited through my art career to exhibit at places, it became common that I would also make workshops where participants would assembly their own synthesizer. At some point my good friend Jacob Bak took over the project and gave it a second life by updating the synth engine to run on Teensy, later David Gauthier would join too, and we would all join forces under the moniker Vsionhairies. More recently, and thanks to the collaboration of Dennis P Paul, professor at the Art Academy of Bremen, we ported the synth to run on our own ARM based architecture. We run courses on generative music using code which is sparking the interest in making new hardware around it, once more.

Cuartielles: If I got this right, you met all three at CIID, right?

Remin: That's true for Dennis and David, but I had met Jacob before during my time at DTU. We studied the same place for years but never met on campus; first time we spoke was on a bus to a techno festival in Sweden. Jacob later became an intern in my company to learn more about sound synthesis and PCB design by reworking the CFO from the ground up.

Cuartielles: Tell us more about that. How does it work for an artist in your discipline? How do you make your money?

Remin: I run a single-person company; it is a very common thing here in Denmark. I have been the CEO of my own company for years [laughs]. It allows me to freelance while keeping everything lean. I am not interested in growing as a company. In the art world money comes and goes in irregular flows, so being responsible for a lot of other peoples paychecks on a month-to-month basis would be super stressful for me.

Cuartielles: This means that you must have quite a constant influx of projects. Is everything you do related to electronics and music?

Remin: Yes and no. I do quite a bit of social engineering as well. For instance, I got the possibility of opening an art space below my apartment in Nørrebro, Copenhagen. We called it Mikrogalleriet, because it was tiny. Among other things, we used it to build a community of 8-bit music makers: 8bit klubben. We had some great art shows too. Tristan Perich produced his first edition of "Interval Studies" there. I participated in forming other groups such as Science Friction, Click Festival, KKT and Centre for Cyber Wellness. A lot of good things happened from being part of these spaces, but working in groups almost always means cutting down on my personal production. Lately I have redirected my efforts towards the production of artworks once more.

Cuartielles: Have you completely abandoned curation to focus on hands-on activities then?

Remin: Almost. But rather than splitting the two, I try to integrate them in my arts practice. I see a value in helping other people and doing community work, where I can leverage some privileges for other people. Bringing others into the scene will benefit the world in a broader sense. There are enough white men making synths and electronic art; therefore, I try to balance this through collaborations. This is very present in my work, and I think a lot about this. The question of how to be an artist sustained by an art career and still support the community.

Cuartielles: Then, I guess that some of your work exists also because of these experiences you mention where you get to collaborate and co-create things. Your latest large-size installation, called Skyen, seems to me like the result of having an open dialogue with different people (Figure 7). It consists of over 300 Arduino-compatible boards hanging from the ceiling in front of a fab lab in Denmark. How did you come to produce this piece?

Figure 6: So many PCBs. (Source: J. Remin, Photo by Jonas Normann)

▲

Figure 7: The Skyen installation. (Source: Jonas Norman)

Remin: Very much so! Skyen is a permanent commission for Spinderihallerne culture centre in Vejle, Denmark. And as such, it has gone through a long development and funding period before its realization in April 2022. This has not been an easy process, but one of the reasons it has been successful is exactly because I developed a very good relationship with both the core team of the fab lab and Eva Sommer Hansen of Spinderihallerne.

Cuartielles: Can you tell me more about the work?

Remin: In the last years, I have been exploring datacentres and global computational infrastructures, what we often refer to as "the cloud". My suggestion for Spinderihallerne was to build an abstract data centre, made by modular PCBs connected by stardard network cables. I envisioned a huge computer hanging in the air processing data. Each one of the modules is essentially an Arduino-compatible board with 14 LEDs and network ports running both power and serial over standard ethernet cable. Some of the modules are running a fan by means of a MOSFET. On top of that, there is some power circuitry to inject 12 V at every 15 modules, so that there is not too much voltage drop going through all of that cable.

All of the 300 modules run the same exact same code, the brain of the installation is an Arduino Mega, which acts as a sort of conductor changing overall programs, intensities and rhythms. The Arduino Mega is connected to a Raspberry Pi, which acts as a Wi-Fi hotspot offering access to a curated selection of text and media. The hotspot also allows people to get access to a terminal offering full control over the installation, but only if you are willing to engage with a command line interface. I insisted on command line interaction, as this is an important point about who holds real power over datacentres and the difference between admins and everyday users.

Spinderihallerne is a big culture house with a fab lab, museum, artist studios, startup zone, and other community services. They wanted an art manifestation that highlighted what they were doing in the house which, at the end, is more about connecting the community and activating the citizens than the specific technologies involved. As a deliberate gesture, local users of the culture house were invited to help assemble the piece. And now that it is finished, it is not a passive sculpture, it is a room for exploration. So far we have hosted talks, concerts, hackathons and even morning yoga under Skyen.

Cuartielles: Skyen is open-source, right?

Remin: My electronics and code always is. I think it is great that my work can live on in many different forms. At some point, I met some hackers in China that had made a version of my synth. I was so happy to see that, it looked amazing! For the installation, I coded everything, designed the boards, built the installation, and open-sourced it. I don't expect anyone to build that installation again, but I could expect people building things on top. When sharing the work, your design world transcends the object. A lot of artists and craftsmen get to collaborate and exchange things with others. All of my work relies heavily upon works by others, the tools, the libraries, the materials. Giving my code and circuits back is the least I can do.

Cuartielles: But what about identity? Would you think that people would be able to distinguish your work from that of others? If you share it, will people eventually pose as you? (This is a tricky question that I always bring up when discussing with people in the art world.)

Remin: I am not so worried about people separating my work from the work of others. Art is all about context. Art can be found objects and sampled media. Art can be very personal. I think the image of the solitary artist genius is an idea of yesterday, I much rather prefer to be generous and engage with the world.

Cuartielles: And what are you busy with now? What is your latest work all about? I heard you're also using an Arduino in it, even if just at the production level.

Remin: I am working on a video piece for an exhibition called *Caring Futures*, which will be shown at Galleri Sølvbjerget in Stavanger, Norway. The main theme is how healthcare is bound to technology and the internet and how the capitalist agenda of these technologies is putting the future of healthcare in jeopardy. I really wanted to produce a piece which carried forward a vision of the future which was positive, but I found it very hard given the development of things in the last 40 years or so. I decided I needed to go back to the roots of cybernetics, back to that inherent positivity of the late 60ies, and so I took as a point of departure Richard Brautigan's 1967 poem, *All Watched over by Machines of Loving Grace*, which also lends its name to the piece. In June we held a workshop with 10 healthcare professionals in Stavanger, in the workshop we produced the initial lyrics to a song that I then wrote music for. I brought it to Simon Littauer a music producer here in Denmark and we made it into a great pop song. I decided to take this further and produce a music video with a giant silicone-based toad that has an Arduino inside to control its internal glow and other effects. The techno toad sits in my hand while I walk in a forest, singing this song of cybernetic harmony. Together, we become part human, part animal, part network. Arduino allowed me to build several technology props for the video: the toad, my LED glasses. These props are neither perfect nor long lasting, but they completely define the project. That is, I wear those glasses during the whole video. I would not have been able to do this otherwise.

220425-01

Questions or Comments?

Do you have any questions or comments relating to this article? Feel free to contact Elektor at editor@elektor.com.

About the Author

David Cuartielles co-founded Arduino. He holds a PhD in Interaction Design and an MSc in Telecommunications Engineering, and he teaches at Malmö University.

Related Products

Looking for the main items mentioned in this article? Arduino and Elektor have you covered!

- **Arduino Mega 2560 R3**
  www.elektormagazine.com/arduino-2560
- **Arduino UNO**
  www.elektormagazine.com/arduino-uno

WEB LINKS

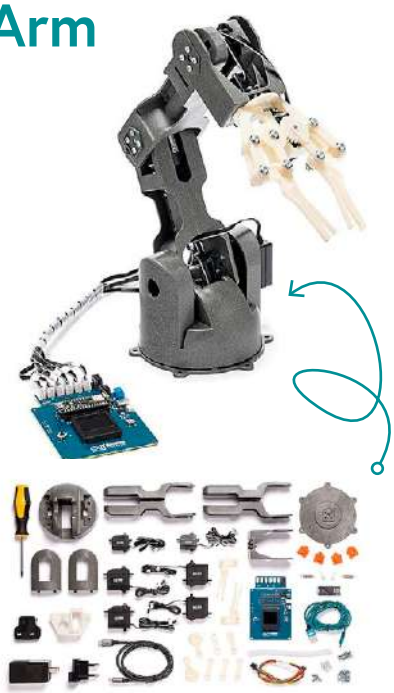
[1] P. Antonelli, "Welcoming New Humble Masterpieces into MoMA's Collection," INSIDE/OUT, MoMA, November 5, 2014: https://mo.ma/3SoiY2c

[2] C. Abate, "Making Art with Electricity: A Q&A With Kelly Heaton," ElektorMagazine.com, August 2022: https://www.elektormagazine.com/q&a-heaton

Get your hands on new Arduino Hardware!

There is nothing that excites us more than getting our hands on new hardware, and so this collaboration with Arduino has been a treat! Want to experience the real deal yourself? Elektor has stocked up the stores to accommodate all products that are featured in this edition!

Arduino Braccio++ RP2040 powered Robot Arm



The next evolution of the Tinkerkit Braccio robot is called Arduino Braccio ++ , a brand new robotic arm designed for advanced users. Arduino Braccio ++ can be assembled in several ways for multiple tasks, such as moving objects, mounting a camera and tracking your movements, or attaching a solar panel and tracking the movement of the sun. Arduino Braccio ++ offers a multitude of expansive possibilities from the very outset, including a new Braccio Carrier with LCD screen, new RS-485 servo motors, and a totally enhanced experience.

www.elektor.com/20174

PID-based Practical Digital Control with Raspberry Pi and Arduino Uno

www.elektor.com/20274

Arduino Pro Nicla Vision

Nicla Vision combines a powerful STM32H747AII6 Dual ARM Cortex-M7/-M4 IC processor with a 2 MP color camera that supports TinyML, as well as a smart 6-axis motion sensor, integrated microphone and distance sensor.

www.elektor.com/20152

Arduino Pro Portenta H7

Portenta H7 allows you to build your next smart project. Ever wanted an automated house? Or a smart garden? Well, now it's easy with the Arduino IoT Cloud compatible boards. It means: you can connect devices, visualize data, control and share your projects from anywhere in the world.

www.elektor.com/19351

Arduino Pro Portenta X8

Portenta X8 is a powerful, industrial-grade SOM with Linux OS preloaded onboard, capable of running device-independent software thanks to its modular container architecture. It's basically two industrial products in one combining Arduino's availability of libraries/skills with container-based Linux distribution.

www.elektor.com/20270

Arduino Pro Nicla Sense ME

A new standard for intelligent sensing solutions.

www.elektor.com/20327

Portenta Vision Shield (Ethernet)

The Portenta Vision Shield brings industry-rated features to your Portenta. Professional computer vision, directional audio detection, Ethernet, and JTAG for Arduino Portenta.
www.elektor.com/19511

Portenta Vision Shield LoRa®

This Portenta hardware add-on will let you run embedded computer vision applications, connect wirelessly via LoRa® to the Arduino Cloud or your own infrastructure.
www.elektor.com/20332

Portenta Breakout

Portenta Breakout board is designed to help hardware engineers and makers to prototype and help test devices connections and capacity within the Portenta family boards.
www.elektor.com/20341

Arduino Pro Portenta Max Carrier

Easily prototype your Portenta applications. Deploy in zero time. Max Carrier transforms Portenta modules into single-board computers or reference designs that enable edge AI for high-performance industrial, building automation and robotics applications.

www.elektor.com/20271

Arduino Uno Rev3

The classic high-performance, low-power AVR® microcontroller. The Uno is the best board to get started with electronics and coding. The Uno is simply the most robust board to enable you to start tinkering with the Arduino platform.

www.elektor.com/15877

Arduino Make-Your-Uno Kit

A new kit including a **DIY through-hole UNO** with all components to build up your UNO and make your own UNO-powered synth!

www.elektor.com/20330

Arduino Ethernet Shield 2

www.elektor.com/19941

Arduino Sensor Kit Base

www.elektor.com/19944

Arduino Nano

The Arduino Nano is a small, complete, and breadboard-friendly board based on the ATmega328 packed in the smallest available form factor of 18 × 45 mm!
www.elektor.com/17002

Arduino Nano RP2040 Connect

The Arduino Nano RP2040 Connect is an RP2040-based Arduino board equipped with Wi-Fi, Bluetooth, a microphone and a six-axis smart motion sensor with AI capabilities.
www.elektor.com/19754

Arduino Nano 33 BLE Sense

Bring the power of AI to your pocket with the more powerful nRF52840 processor and a series of embedded sensors and the possibility of running Edge Computing applications (AI).
www.elektor.com/19936

The Future of Arduino

By David Cuartielles (Co-founder, Arduino)
and Fabio Violante (CEO, Arduino)

As we advance in the third decade of the 21st century, more and more of the original paradigms of ubiquitous and pervasive computing are becoming real. IoT in all of its forms (at home or at work) is becoming a reality thanks to the contributions of emerging technologies such as tinyML, long-distance low-power radio systems, and ultra-low-power microcontrollers. The question is no longer whether AI will be part of our lives, but how much will we let these kinds of technologies take over tasks that we are currently doing in other ways. There is a clear benefit in having machines that are always on, and always connected, but we have to look at making such things in a socially-responsible and environmentally-sustainable manner.

This is why we keep on looking at the creation of transparent technologies. Devices, software frameworks, and services should be attentive to end users' needs, such as connectivity or processing power, while offering clear guidelines on how to deal with sensitive aspects such as privacy, micro-transactions, or secure over-the-air updates. In the last years, we have made a move towards increasing the technical capabilities of Arduino boards and created a whole new family of boards shaping to our professional line. They represent how we understand the future of microcontroller boards where systems will become more complex and integrate more components — such as cameras, digital microphones, or environmental sensors — by default. On the software front, the industrial Arduino computers can already be updated over the air using secure bootloaders. What we can expect from the future is seeing these devices running in your own private clouds, operating your additive manufacturing machines in the form of fully-reprogrammable PLCs, executing complex ROS operations in the robots in your factory, or running a robust Linux-based industrial computer monitoring your workshop.

On the educational front, there is a high demand from families, governments, and school districts for more capable technologies. Terms such as Python and AI have become a must in most of the educational curricula in the countries where Arduino operates. We can only expect to see how processors that were used until recently for industrial applications will end up being part of more powerful, AI-enhanced, educational tools. Edge computation and on-device training are secure ways to introduce the latest technologies in the classroom. At the same time, we will see how new workflows will allow for a zero-coding approach to AI to emerge and will make digital technology into a malleable construction material for new everyday artefacts. Teachers will be able to focus more and more on applications as new programming environments will simplify how students learn about digital technology. Modular Arduino tools, connected to phones, tablets, and laptops will become the mobile lab of the future. Students will do fieldwork using their self-made portable laboratories and store data in their mobile devices for later analysis.

In higher education and academic research, Arduino will continue to play a relevant role by offering reliable, low-cost boards with simple APIs. This will help researchers build their own tools and experimental machines for a fraction of the cost of a commercial one.

Makers will continue to be Arduino's main source of inspiration. We will see new boards with newer processors and capable of industry-like performance. The Arduino IDE 2.0 will first get contributions in the form of translations to slowly transition into the production of new plugins to enhance its functionality. It is only a matter of time until people start making their own dashboards to represent information from the boards and launch them from the IDE, automate the programming of full fleets of devices at the click of a mouse, or monitor the bus of a complex machine with dozens of sensors and actuators hanging from it.

The connected operation of Arduino devices will be handled by the Arduino Cloud. Operations such as over-the-air software updates, drag-and-drop dashboards to control devices, or templates to run off-the-shelf machines are already working from the Arduino servers. We can imagine that the future will bring ways to pipe information flows to cloud-based AIs to detect patterns and establish modes of operation. It is only a matter of time until the whole edge intelligence of their devices will be uploaded to a user's profile to be downloaded later to a replica of the devices somewhere else. This will allow for hot-swapping devices from a fleet without stopping their operation.

Arduino was born to support a community — the one of artists and designers — in getting access to embedded technology. Over time, the Arduino platform has reached other communities of practice. Engineers, researchers, teachers, and makers — all part of the Arduino family — are called to shape the future of the platform with us. Join our open repositories and let's make things happen!

220541-01