www.elektormagazine.com

elektor MAG

design > share > earn

WK 51 * SINCE 1961 *
DEC 2022/JAN 2023
ELEKTORMAGAZINE.COM

Special edition guest-edited by ARDUINO

Declassified Bonus Edition!

Home Automation

Connectivity Simplified

Prototyping to Production

p. 4 Retro Gaming with Arduino

p. 14 A Controller for Spotify

Free Brand-New Bonus Articles Every Week!

Elektor's Arduino Training Board p. 10

Available in week 52

Bringing the Human [redacted] p. 24

Available in week 01 2023

Create Kinetic [redacted] p. 32

From Arduino and Elektor with Engineering Love

OUT NOW

A unique Elektor Magazine edition curated by guest editor Arduino!

DIY electronics projects, engineering insights, and more from Arduino and Elektor engineers

Packed with projects and tutorials

Dive into hot topics such as MicroPython, TinyML, and home automation with Arduino

£9.95 | €10.95 | $12.95
elektor MAG
design > share > earn
518
DEC. 2022 & JAN. 2023
ELEKTORMAGAZINE.COM
SINCE 1961
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 in Arduino
In this issue
Writing Arduino Sketches Just Got Better
Mozzi Arduino Library for Sound Synthesis
Introduction to TinyML
Connected Projects Simplified
MicroPython Enters the World of Arduino
and much more!
Taking over Elektor!
Building Future Skills with Arduino
Save the Planet with Home Automation
Getting to know Arduino with Fabio, Massimo, and David
ISSN 1757-0875

Get Started with the Portenta x8

Get it now! From your favorite newsstand or buy in the Elektor web stores!

Direct links in articles give you easy access to Arduino products and solutions

Get to know Arduino: Insights from Fabio, Massimo and David

More information

www.elektor.com/arduino-magazine

Elektor Magazine, English edition

Volume 48, BONUS EDITION
December 2022 & January 2023
ISSN 1757-0875 (UK / US / ROW distribution)
www.elektor.com / www.elektormagazine.com

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

Memberships: service@elektor.com
www.elektor.com/memberships

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

The Team:
International Editor-in-Chief: Jens Nickel
Content Director: C. J. Abate
International Editorial Staff:
Eric Bogers, Jan Buiting, Stuart Cording,
Rolf Gerstendorf, Alina Neacsu,
Dr Thomas Scherer, Brian Tristam Williams
Laboratory Staff: Mathias Claussen,
Ton Giesberts, Luc Lemmens, Clemens Valens
Graphic Design & Prepress: Harmen Heida,
Sylvia Sopamena, Patrick Wielders
Publisher: Erik Jansen

Declassified Bonus Edition!

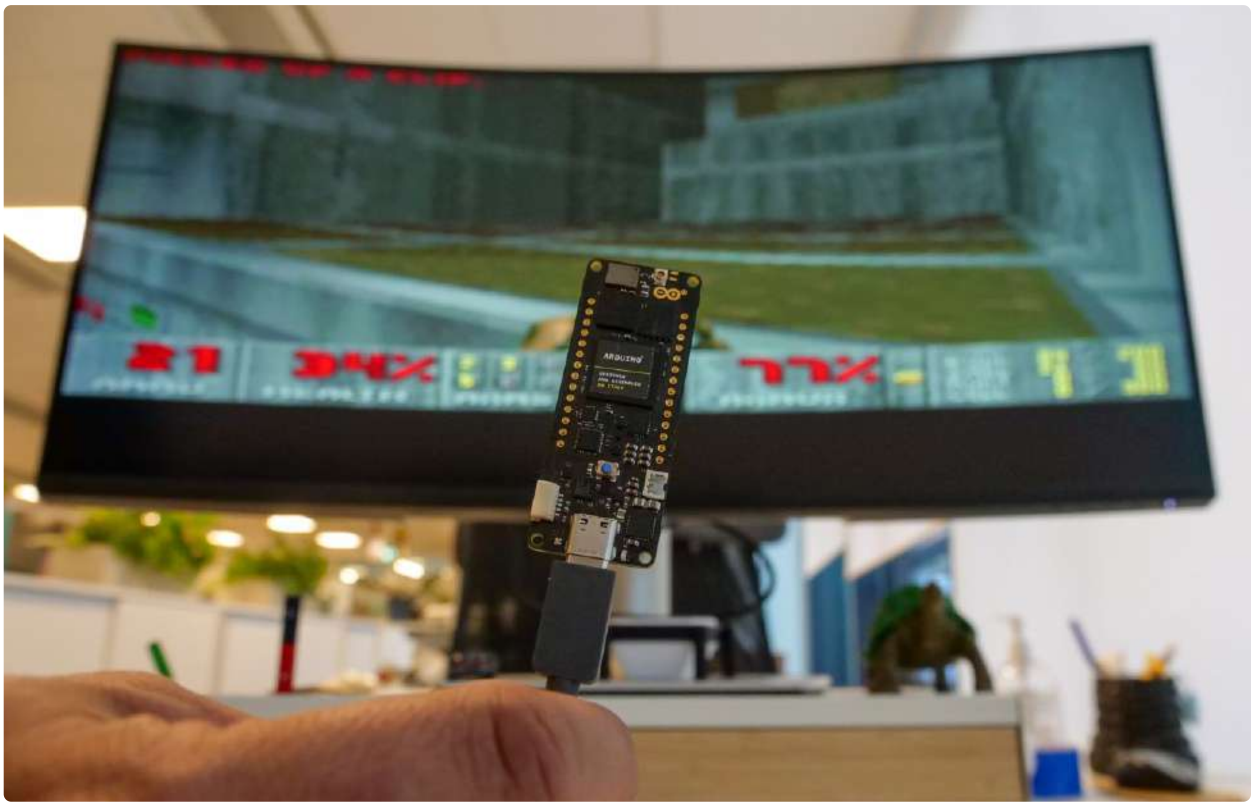
The creative collaboration between Elektor and Arduino did not end with the guest-edited edition of Elektor Mag that we published in early December 2022. We have more projects, technical insights, and informative articles for you to keep inspired for months to come. Over the course of four weeks, we are declassifying content in this edition until you have the complete bonus magazine in early January 2023. What a great way to kick off the new year!

Whether you are a pro engineer working on a new industrial product or a DIYer looking for a fun weekend Arduino-based project, you will find this extra edition of Elektor Mag informative and stimulating. We provide you with articles on a wide range of Arduino-related topics and projects, including retro gaming with Arduino, an Elektor Arduino training board, and a portable Arduino-based controller for Spotify.

As you read the projects and articles in this magazine, feel free to share your thoughts with us at elektormagazine.com, arduino.cc, and on social media. We look forward to your feedback. Enjoy!

C. J. Abate (Content Director, Elektor)

THIS EDITION



You can run Doom on a Portenta!

Running **Doom** on a **Portenta**

Retro Gaming with Arduino

By David Cuartielles (Arduino)

You can run Doom on an Arduino Portenta H7. Curious how it's done? Want to know why Arduino engineers ran the game on it in the first place? Martino Facchin, head of Arduino's firmware team, has the answers.

Doom is probably the most sold game in history, with sales of over 3.5 million copies. At a retail price of $50 (USD), the Doom developers, forming a small company called id Software, became millionaires overnight. When Doom was released in 1993, id Software had already been out with another well-known game for about a year, Wolfenstein 3D. Doom is a first-person shooter game, where the player is meant to navigate a 3D space and confront different enemies using different weapons and ammunition that can also be found all over the game's battlefield. Doom has been ported to all operating systems but also runs on bare metal on multiple systems. The source code is open, currently licensed under GPL. While it is not trivial to compile it, we have seen versions of Doom run on very small computers and also inside other programs. At some point, Microsoft Office's Excel 95 made a homage to Doom in the form of a playable easter egg to be found in the software's credits page.

The game has become a way to both check the performance of small computers and to display hacking wizardry. A few months ago, at DEF CON 22 in Las Vegas, the hacker known as @sickcodes on Twitter and GitHub demoed Doom on a John Deere tractor modded to include farm-related graphics. Arduino is no exception. And when the first prototype of the Portenta H7 — our most powerful board to that point in 2018 — came out, we used Doom to test the technical capabilities of the board. I recently invited

Martino Facchin, head of Arduino's firmware team, to tell us more about this story and how it was done.

David Cuartielles: Let's talk about the Arduino Portenta H7 running Doom. I have prepared a little summary of the history of Doom, how the guys at the id Software company made the game, how it sold like crazy, and how the makers became millionaires.

Martino Facchin: And then they published the source code — the most important thing.

Cuartielles: Exactly! They published the source code, but under which license did they publish it?

Facchin: I've got to check, but I think it is a license which is compatible with GPL, the Doom Source Code Licence. The important thing is that only the engine is open source. The assets are closed, and, in fact, you can only play the shareware version of Doom, so you cannot really play the full game unless you bought it.

A Note from David

I went to check this because I wanted to be sure about it, and in 1997, id Software published Doom under the above-mentioned licence, which was opening up the source for educational purposes. After an accident that happened to the maintainers of glDoom, which left the world without a copy of the openGL port Doom because of the non-distribution clause in the Doom licence, id Software agreed to modify the licence to GPL.

Cuartielles: The cool thing with this game is that people were making their own mods. I remember a version of Wolfenstein 3D with Star Wars characters.

Facchin: I don't remember that.

Cuartielles: I am that old. I remember for the both of us.

Facchin: Well, I recall playing Wolfenstein 3D when I was a child. But we didn't have an Internet connection, so we couldn't get all of these goodies from the modders' community.

Cuartielles: **Before we continue, let me ask you first. Could you introduce yourself?**

Facchin: [laughs] Martino Facchin, firmware engineer at Arduino.

Cuartielles: **What is your role in Arduino?**

Facchin: I am head of firmware — the main guy when you need some firmware help. Now, I have a wonderful team of colleagues that help me with this, because I started this team just by myself. The team keeps on growing. We try to also make the community grow with us by making everyone more aware of what we are doing.

Cuartielles: Besides making Doom run on a Portenta (something we will talk about more later), what is the thing you made at Arduino that makes you the most proud?

Facchin: I'd say that it's the PluggableUSB framework. When I just came to Arduino, having been there for six months, we had this big issue of people willing to add multiple functionalities to the USB port of the Arduino Due and the — by then — forthcoming Arduino Zero. Every time people were plugging an Arduino Leonardo into a computer, it was bringing up the keyboard drivers, the mouse drivers, etc., even if you were not using them. We had to build this thing on the fly, a USB descriptor that would help the users see just the things they wanted to use at the time. At once, we also allowed for other things to happen, things that people wanted to use such as USB MIDI and the like. For me, this was a huge development. I was pretty young at the time and had to interact with the community with the help of Matthijs Kooijman (read more about his work at www.stderr.nl) and Paul Stoffregen (creator of Teensy) to sort out the best possible strategy. And it worked. Every now and then, people come and say, "I used the MIDI library for this or that," or there is even a developer now making all of the layouts for all international keyboards built on top of the foundations of that code. I am very proud of that.

Cuartielles: **You said that you were "pretty young at the time." For how long have you been working for Arduino?**

Facchin: Six and a half years, now. Based at the Torino office.

Cuartielles: **That's quite some time. And how many people are involved with the Firmware team?**

Facchin: Six people. It might seem like a lot, but at the Firmware team, we maintain all of the products, while also performing other activities such as certification. On the other hand, we have

The obvious choice was to port Doom and try to instrument all the features that were needed for it to run.

separated Firmware from Tooling, which is a different team, dedicated to the Arduino CLI and other parts of the higher end software.

Cuartielles: **And all developers work against GitHub, correct?**

Facchin: Yes, everything is open sourced by the end of the development process.

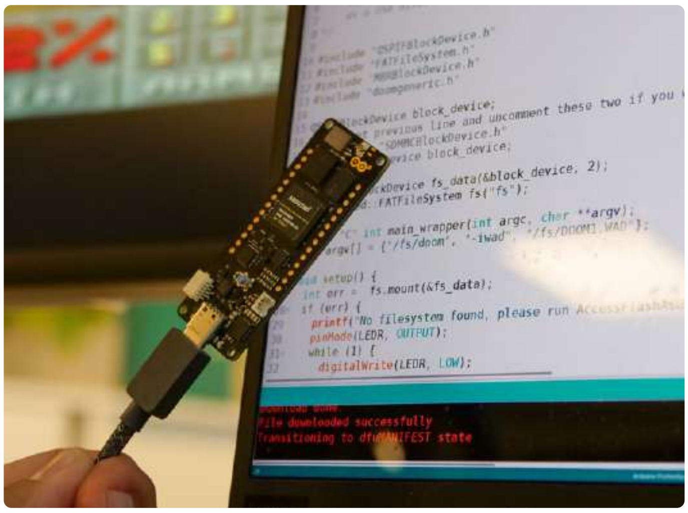
Cuartielles: **Great. Let's go back to talk about Doom. We know that it was a very successful game that sold over 3.5 million units, making their developers filthy rich. It was the first best-seller first-person shooter (FPS) game. The code was open sourced and was ported to all kinds of devices.**

Facchin: Phones, calculators, every operating system.

Cuartielles: **And Arduino's Portenta H7, a dual core processor board aimed at industrial environments. Who had the idea of running Doom on the Portenta H7?**

Facchin: It was more that we had just got the first prototype of the Portenta H7, a nice board with a lot of chips, and everything was still to be done. We had this chip from Analogix that you can typically find in other devices that converts MIPI signals into DisplayPort, which would allow it to address external monitors. We had no experience in the subsystems in the chip that should be handling this, and the existing examples were not helping either. At first, we managed to render some yellow boxes on the screen, then the Arduino logo with a few artefacts, but it was far from perfect. We were not really understanding why things were not working as expected, so I went for something with moving images that I could easily recognise.

The obvious choice was to port Doom and try to instrument all the features that were needed for it to run. It took very low effort to make it work on something without an operating system. It is not really the fork of Doom you would play, but it is very easy to port. You just need to change six or seven functions to adapt it to your hardware and you are ready to go. To begin with, we had no code to run the internal RAM or the external memory. I had to get those to work first and then fire up the simulator. From the beginning, we



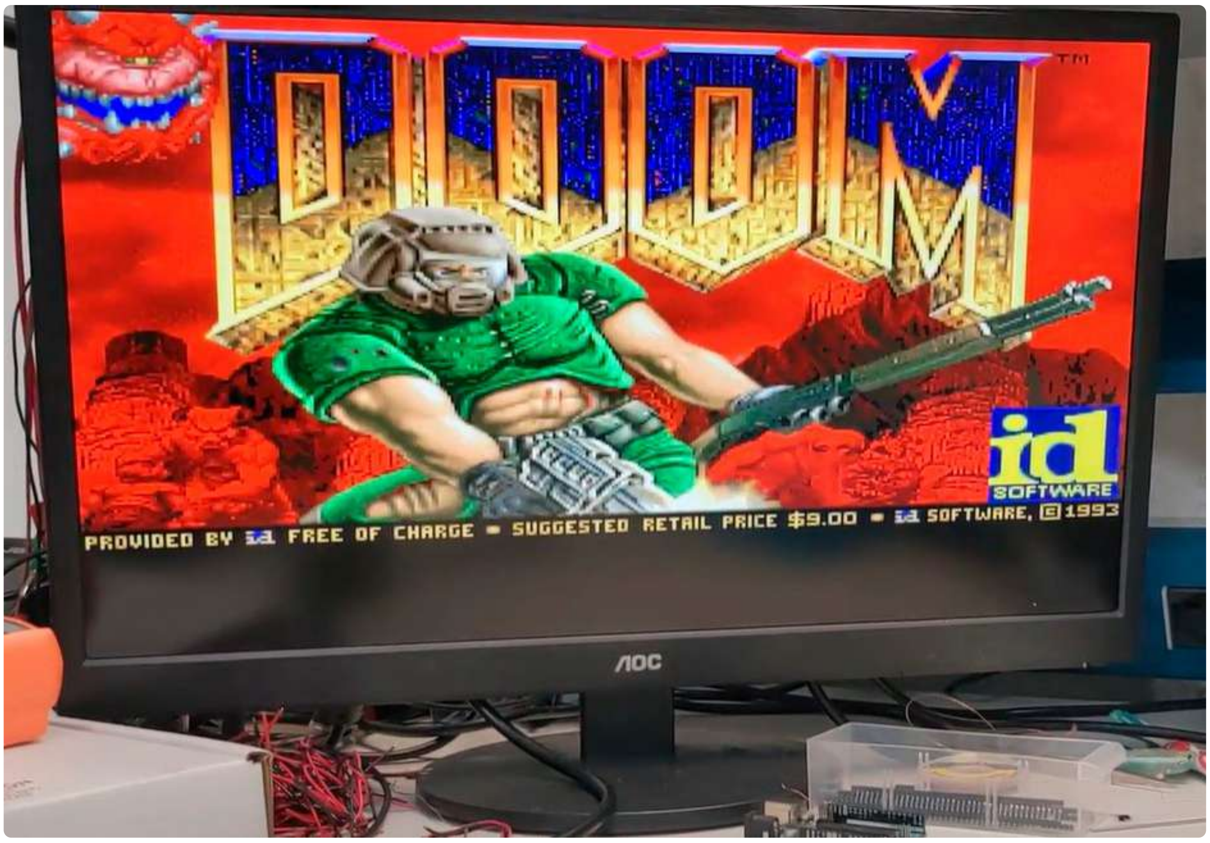
```
#include "QSPIFBlockDevice.h"
#include "FATFileSystem.h"
#include "MBRBlockDevice.h"
#include "doomgeneric.h"

...ockDevice block_device;
... previous line and uncomment these two if you
... "SDMMCBlockDevice.h"
...vice block_device;

...kDevice fs_data(&block_device, 2);
...: FATFileSystem fs("fs");

..."C" int main_wrapper(int argc, char **argv);
...argv[] = {"/fs/doom", "-iwad", "/fs/DOOM1.WAD"};

...setup() {
  int err =  fs.mount(&fs_data);
  if (err) {
    printf("No filesystem found, please run Access...
    pinMode(LEDR, OUTPUT);
    while (1) {
      digitalWrite(LEDR, LOW);
```

```
File downloaded successfully
Transitioning to dfuMANIFEST state
```

It is up to the reader to take over the project and add controls to make the game playable.

How to run Doom in your Portenta H7

1. Download the latest version of the Arduino IDE. We recommend Arduino 2.0 or more recent.
2. Download the Portenta core from the board manager.
3. Select the M7 core for your Portenta H7 board. All of the following steps have to be executed on that core.
4. Make sure the IDE has identified the port here the board is connected to.
5. There is an example at "Examples / Doom" where you can see the basic instructions. Before you install this, you need to run a couple of examples on your board.
6. [Optional] Upgrade your Portenta H7 bootloader to the latest version with "Examples / STM32H747_System / STM32H747_manageBootloader".
7. Format the external Flash using the example "Examples / STM32H747_System / QSPIFormat." After the installation, you will have to open the serial terminal and follow the instructions given to you in it.
8. Transform your board into a mass storage device as if it was a USB drive with "Examples / USB As Mass Storage / AccessFlashAsUSBDisk."
9. Open the Serial monitor and choose how the formatting of the board should proceed. Once it is done, your computer should register two new external drives connected to it.
10. Download DOOM1.WAD from the DoomWiki site at: https://doomwiki.org/wiki/DOOM1.WAD and copy it in the largest partition of the virtual Portenta drive.
11. Go back to the Doom.ino example we saw on step 5 and flash it on your board, remember: always on the M7 core. If you had problems seeing the programming port, just double click the reset button on your Portenta prior to uploading and make sure the serial port is properly recognized.
12. Disconnect the Portenta H7 from your computer and plug it to a USB-C hub as if it was a laptop. The hub will need to have the external power connected and an HDMI cable to send the video signal to a computer monitor.


DOOM
TM
id SOFTWARE
PROVIDED BY id FREE OF CHARGE - SUGGESTED RETAIL PRICE $9.00 - id SOFTWARE, © 1993
AOC

Doom is a classic!

didn't get video to work, I had to prepare a framebuffer, do some magic, and get some proper output over USB-C.

Cuartielles: **The Portenta H7 is a dual-core processor. It has an Arm Cortex M4 and an M7 inside the processor. Which one was the one running Doom?**

Facchin: Today we use the M7, but back then we ran it on the M4, because it was much simpler from an embedded programmer point of view. It is more like a typical microcontroller, without any special features. On the other hand, the M7 has this cache that you have to look into and invalidate at the right time when generating video. The first time I tried to run things on the M7, it was real fast, but also completely broken, and I could not see anything on the screen. The M4 was fast enough (25 frames per second) and pixel perfect.

Cuartielles: **25 fps is a lot faster than my first computer. But let's summarise this for the reader: You had Doom running in the slowest of the two processors, on a board which has Bluetooth, Wi-Fi as connectivity. Video is sent out via USB-C. There you can have a hub, mouse, keyboard, whatever. Which were the controls you implemented there?**

Facchin: Unfortunately, the project died there for us, because once we saw we could make video run, we started working with LVGL, which is a much more useful library for other developers to build applications on top of Portenta. LVGL is completely integrated with the USB hub, keyboard and mouse, so that you can build all needed interfaces for the professional context that many of Arduino's end users need.

Cuartielles: **Wouldn't it be cool to have a professional PLC based on Portenta H7 where you could be playing Doom on the Cortex M4 core while doing the serious work on the M7?**

Facchin: Absolutely!

Cuartielles: **Thanks, Martino. It was great to hear the story of Doom running on Arduino Portenta H7. We will share with our community the basic instructions on how to get it up and running. It is up to the reader to take over the project and add controls in order to make the game playable.**

220542-01

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.

Questions or Comments?

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

Related Products

- **Arduino Portenta H7**
  www.elektormagazine.com/arduino-portenta-h7

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!

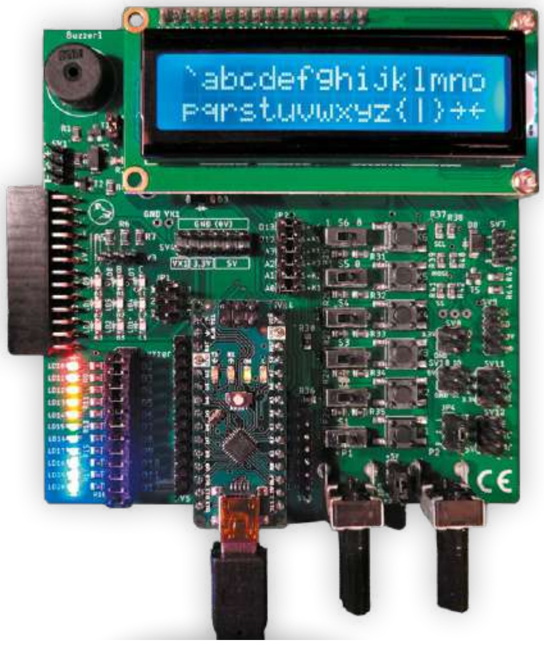
www.elektormagazine.com/labtalk-david

Crash Course Into the Arduino World

Development Board for the Arduino Nano

By Wolfgang Trampert (Germany)

Elektor remains true to its educational mission: here we present a brand new training board with an Arduino Nano at its heart. Together with a well-structured, hands-on training course, it provides an ideal platform to upgrade your skills and to explore the world of microcontrollers.



Editor’s Note
At the time of publication, the book associated with this kit is available in German only. Translations are planned for the near future. Once finished, the translated version will be available in the Elektor Store.

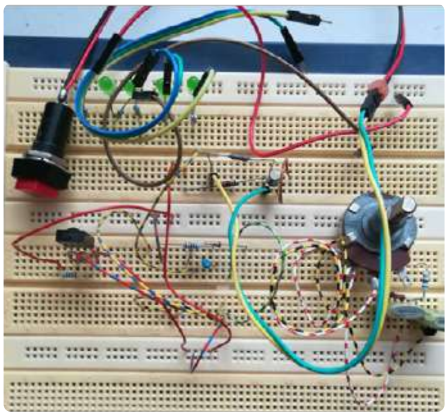


Figure 1: Typical plugboard circuit layout for an Arduino sketch.

You might say the “Arduino philosophy” is associated with a hardware-near design approach: in most cases, the Arduino software or sketch accesses components such as switches, pushbuttons, potentiometers, LEDs, LC displays, piezo buzzers, driver transistors etc via the microcontroller GPIOs. Other types of peripheral devices or electronic modules such as sensors, display or driver boards use various serial interfaces such as SPI, I^{2}C or 1-wire bus to communicate with the controller. In order to familiarize yourself with the world of microcontrollers and Arduino boards, you need to build new practice circuits and control them using an Arduino board.

The essence, however, of any project based on Arduino hardware is development of the software (sketch). The hardware is only a means to an end. You can of course connect up all the peripherals you need for a particular project using a prototyping plug board (**Figure 1**). This approach gives you maximum freedom to install peripherals and connect signals wherever you want. That’s not always a good idea, especially when you are just starting out. The layout can quickly grow in size so you end up with a rat’s nest of Dupont jumper leads. Often you’ll spend more time debugging the hardware and sorting out wiring errors than actually writing code. This only increases the frustration level, and correcting dumb errors won’t necessarily teach you anything useful.

The MCCAB Training Board

To get around these hurdles, we have developed the Elektor Arduino training board,

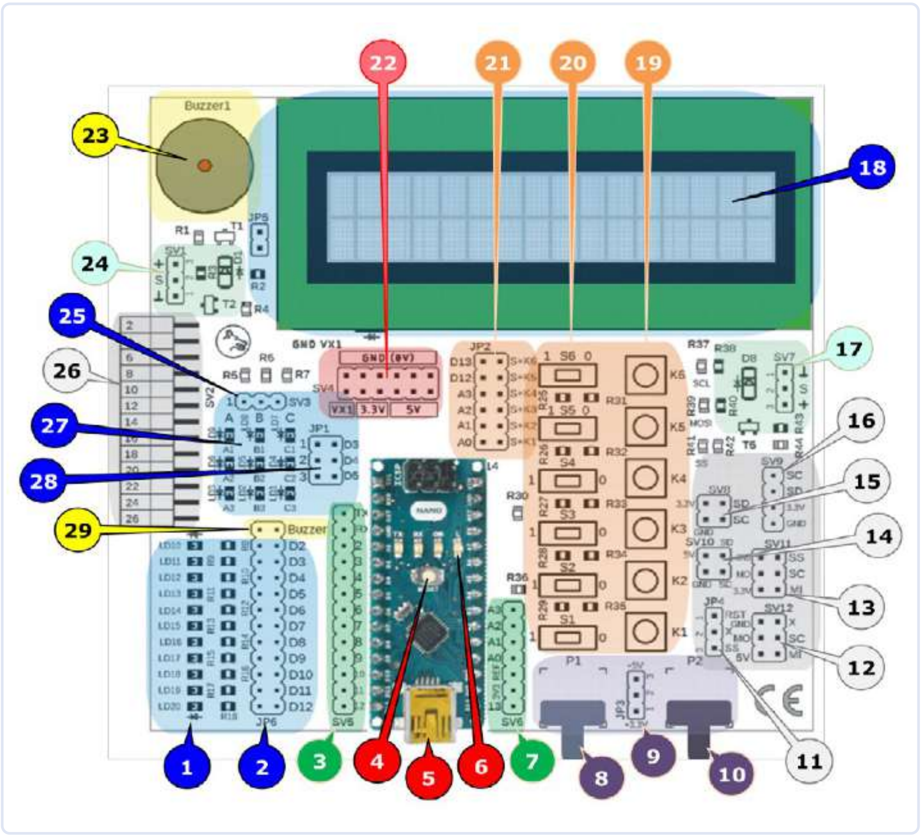


Figure 2: The MCCAB training board, Rev. 3.3.

also known as the *MCCAB Training Board* (**Figure 2**). At its heart is an Arduino Nano board which plugs onto the MCCAB training board. Alongside this are many of the basic peripheral devices you would generally need to build a new prototype for many applications such as a lab setup, test and experimental circuits, projects and exercises to support your studies and training and also hobby projects. The microcontroller GPIOs are all available on two pin header strips on the *MCCAB Training Board* which gives the board maximum deployment flexibility. Additional peripherals or external signals can be hooked up using Dupont jumper leads as required. You won't need to worry about incorrectly wiring the on-board peripherals and will spend less time rummaging through boxes of spare parts to find that elusive component you need to complete a circuit.

Additional circuits built on breadboards can also be easily connected using Dupont cables, since all GPIOs of the microcontroller on the Arduino Nano are connected to the two header strips SV5 and SV6 on the MCCAB training board (pointers 3 and 7 in **Figure 2**). Running down the left hand edge of the board you can also see the double row 26-pin right-angled pin header connector SV2 (pointer 26 in **Figure 2**) where an external expansion PCB can be plugged in.

This connector provides all the important GPIO signals from the microcontroller. External boards to implement functions such as an electronic component curve tracer, a lab power supply or a traffic light controller can be docked to the MCCAB training board here and be controlled by it. Information and results from a running sketch can be written to the on-board 2×16 character LCD (pointer 18), which interfaces via the board's I²C bus. Also on board is a 3×3 LED matrix (pointer 27).

The MCCAB training board is powered by a supply of Vcc = +5 V. This will usually be provided by the USB cable plugged into your PC which you need for creating and uploading the exercise sketch to the MCCAB. The MCCAB can also be powered from an external power pack.

In the Training Board schematic (**Figure 2**), all components associated with a specific board function are identified using a common background colour.

The *MCCAB_Lib* Library for use with the Training Boards

Software development involves using the Arduino IDE to write the program (or 'Sketch' in Arduino speak) which tells the microcontroller how to behave. The sketch is then compiled

The MCCAB training board controls and indicators

1	11 × LED (status indication of input/outputs D2 to D12)
2	Connector linking LEDs LD10 to LD20 with GPIOs D2 to D12
3	Microcontroller inputs and outputs
4	*RESET* button
5	Arduino NANO with mini USB socket
6	LED *L*, linked to GPIO D13
7	Microcontroller GPIOs
8	Potentiometer P1
9	Supply voltage to P1 and P2
10	Potentiometer P2
11	Signal at pin X of SV12
12	SPI-Interface 5 V (The signal at pin X is selected by JP4)
13	SPI-Interface 3.3 V
14	I²C-Interface 5 V
15	I²C-Interface 3.3 V
16	I²C-Interface 3.3 V
17	Switch output for external equipment
18	2×16 character LCD
19	6 × Pushbuttons K1 to K6
20	6 × slider switches S1 to S6
21	Pin header to link switches to microcontroller GPIOs
22	Supply voltage distributor
23	*Buzzer1*
24	Switch output for equipment
25	3×3 LED matrix columns
26	2×13 pin header strip to connect an external module
27	3×3 LED matrix (red)
28	Connections of 3x3 matrix rows to D3, D4 and D5
29	Jumper position links *Buzzer1* to GPIO D9

Table 1: Classes available in the MCCAB_Lib library.

Class	Usage
`KeySwitch`	Debounced status of switches S1 to S6 and pushbuttons K1 to K6
`Matrix`	Control of the 3×3 LED matrix.
`LED`	On / off / blink control of the 12 LEDs LD10 to LD20 and LED
`LedBlock`	Output a bit pattern on all 11 LEDs (LD10 to LD20*)*
`Sound`	Control of Buzzer1 and square wave signal generator.

and uploaded to the Arduino Nano's microcontroller on the training board via a mini USB cable.

The microcontroller GPIOs can be configured as usual using the Arduino function `pinMode()` and the value of signals to and from components on the training board can be read or controlled using `digitalRead()`, `digitalWrite()`, `analogRead()` etc.

However, a library called *MCCAB_Lib* [1] is available which supports the developer by providing additional commands to control the extensive hardware peripherals on the MCCAB training board. The library can be downloaded free of charge and integrated into your own sketch. This library makes handling the on board peripherals much easier.

The library *MCCAB_Lib* contains five classes for controlling the switches, LEDs and buzzer on the training board and can be easily included into the user's sketch as required. **Table 1** shows a list of the classes available.

Using this library means the user does not have to worry about defining time periods for switch debouncing, generating multiplex control signals for the 3×3 LED matrix and flashing the LEDs *LD10* to *LD20* or *L*, or even generating the buzzer tone frequencies. The library functions do this automatically in the background of the program flow, unnoticed by the user.

Listing 1 is a small example sketch to demonstrate use of the *MCCAB_Lib* library.

In line 15 of the sketch, the object variable `Led` is declared of the Class `LED` from the library *MCCAB_Lib*. The parameter `LED_PIN` passed in the declaration of the object variable `Led` is defined as a constant in line 13 indicating the pin to which the LED is connected. This pin is automatically configured as an output during instantiation.

The object variable `Key` of the class `KeySwitch` from the *MCCAB_Lib* library (declared in line 22) during execution monitors (in the background) the state of the switch input on

Listing 1.

```
/*
* Sketch which uses pushbutton K4 to toggle LED LD10 on and off using object variables in the
  "KeySwitch" und "LED" classes in the MCCAB_Lib library.
* To read the status of pushbutton K4 its necessary to insert a jumper to link position S+K4 (the switch
  connection) with A3 (GPIO A3 of the microcontroller)on double header strip JP2 of the MCCAB.
* Insert another jumper (in position D2 of the double header strip J6) to link LED LD10 with the
  microcontroller GPIO of the MCCAB.
*/

11 #include <MCCAB_Lib.h> // bind the MCCAB_Lib Library to the Sketch
12
13 #define LED_PIN   2      // the LED is connected to pin D2
14
15 LED Led(LED_PIN);       // Object-Variable
16
17 //function called by the object-variable "Key" when the switch is closed.
18 void switchTurnedOn() {
19   Led.toggle();         // toggle or flip the state of the LED
20 }
21
22 KeySwitch Key(SK4, ACTIVE_HIGH, switchTurnedOn, nullptr);  // Object-Variable
23
24 void setup() { }    // nothing to do here...
26 void loop() { }     // or here
```

pin SK4, which is passed to it as a parameter according to its declaration. It performs switch debouncing when the pushbutton K4 is pressed or released and calls the function `switchTurnedOn()` when the button is pressed. The `toggle()` method of the class `LED` from the *MCCAB_Lib* library is activated in the `switchTurnedOn()` function in line 19 to invert the current state of the LD10 light-emitting diode.

Since the connection pins for the switch and LED are automatically configured as input and output when the objects are declared nothing else needs to be done in the `setup()` function in line 24 in this sketch.

The `loop()` function in line 26 also does not contain any instructions because the only action to be performed in this sketch is to switch the LED state when the K4 button is pressed. This action is event-driven by the `KeySwitch` class by calling the `switchTurnedOn()` function.

Using these classes in the library *MCCAB_Lib* in more extensive sketches, the two functions `setup()` and `loop()` of the standard Arduino software model would not need be required to continually poll the state of peripheral components, thereby freeing them up for more important tasks.

12 Project Sketches and 46 Exercises

A detailed instruction manual for the MCCAB training board is available and can be downloaded from the website [1]. The MCCAB training board and the *MCCAB_Lib* library will also be described in detail in an upcoming book which will be available shortly (see editor's note).

The book explains in detail the hardware and software basics of a microcontroller system and introduces the programming language C, which is used to write Arduino sketches.

The book's principle focus is on practical exercises, so that "learning by doing" is the key concept used here to acquire the skills you will need when you go on to build your own projects. In a comprehensive practical section, there are 12 project sketches and 46 exercises, so that your knowledge builds as you work through the many examples. The exercises are structured in such a way that the reader is given a task that needs to be solved with the MCCAB training board using the knowledge gained from the theory section of the book. For each exercise there is then a detailed explanation and well-commented example solution that help solve problems.

220450-01

Questions or comments?

If you have technical questions, feel free to email the Elektor editorial team at editor@elektor.com.

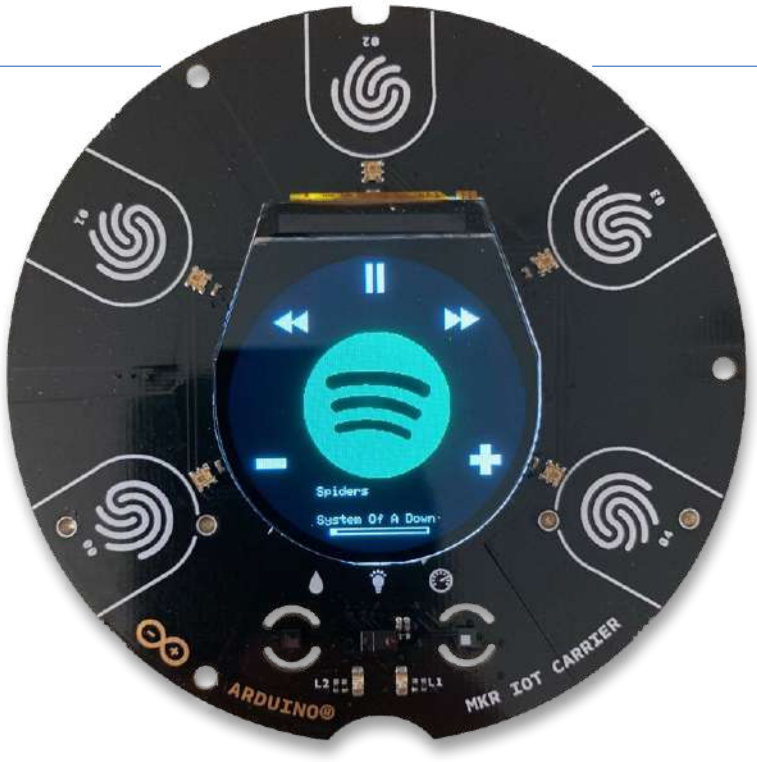
About the Author

Wolfgang Trampert has been developing and programming microcontroller systems since he finished his studies in electronics. His engineering business developed microcontroller based solutions to meet customer requirements. He has authored a number of specialist books and articles and conducts training courses on the subject of microcontrollers.

WEB LINK

[1] The MCCAB_Lib Library: http://www.elektor.de/20295

A Controller For Spotify



The Oplà IoT Kit Is (Almost) All You Need

By Altuğ Bakan (Turkey)

The Arduino Oplà IoT kit contains the MKR WiFi 1010 Maker Board and a carrier board that integrates relays, a round-shaped OLED display, capacitive touch buttons, and some sensors. Here we describe how to build a portable controller for the popular Spotify music player. Of course, some security is needed.

The Arduino MKR WiFi 1010 Maker Board is — thanks to its Wi-Fi capabilities — a perfect brain for your next IoT project. You are even better equipped with the Arduino Oplà IoT kit, which contains this Maker Board and a carrier board (**Figure 1**). The latter integrates relays, a round shaped OLED display and capacitive touch buttons.

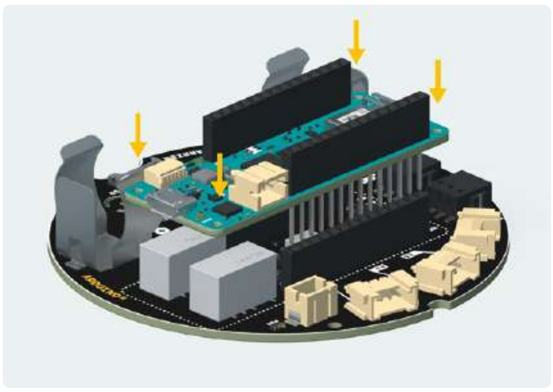


Figure 1: The MKR WiFi 1010 Maker Board is put on the carrier board, which integrates relays and other useful peripherals.

Also in the kit is a moisture and a PIR sensor (**Figure 2**). Projects such as home security alarms and automatic plant watering are therefore easy to implement.

The Wi-Fi feature also allows you to control programs running on your PC, if they have a network interface. The touch buttons, the display, the battery socket, and a housing make it easy to design a portable controller for different kinds of PC software — as an addition to a mouse and keyboard (**Figure 3**).

I am a fan of the music player Spotify, and so I used the Oplà kit to build my own wireless Spotify controller. You can press buttons to skip to the next and previous song, play/pause a song, and increase and decrease the volume. To do so, it goes without saying that the Spotify player must be started on your PC or smartphone.

Secure Communication

Spotify comes with an easy-to-handle programming interface to control your Spotify player via the network; however, you will need the Spotify Plus license for it. Of course, some security is needed. To use the Spotify web API, which is based on REST, you have to authenticate first at the Spotify Accounts Server with your Spotify login username and password. Once authenticated, your software has to send a Client ID and a Client Secret. The Spotify server will return an access token, which you have to send with each call of the Web API to control your Spotify player. This two-step authentication flow is based on the popular *OAuth2* process (see **Figure 4**).

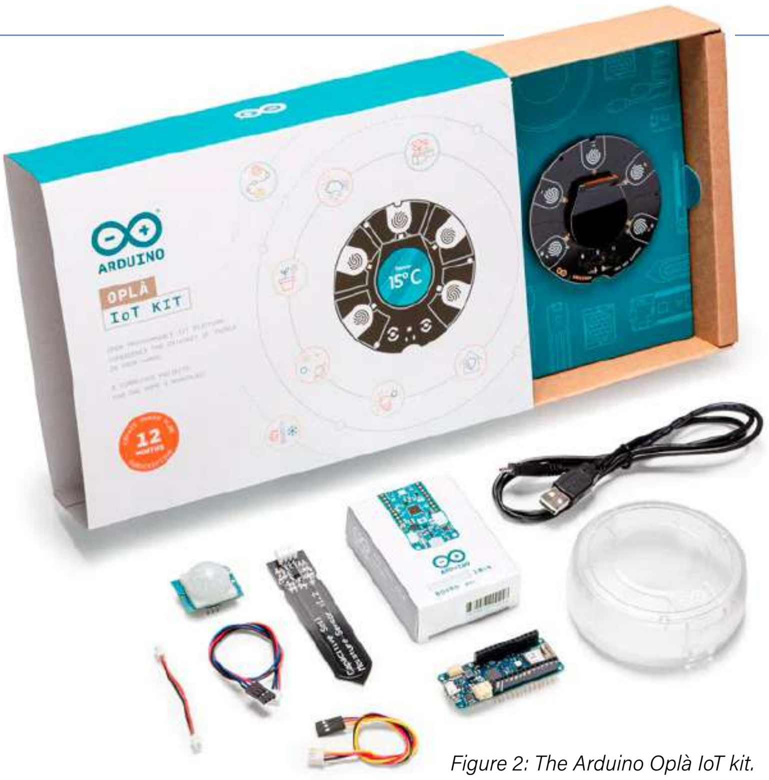
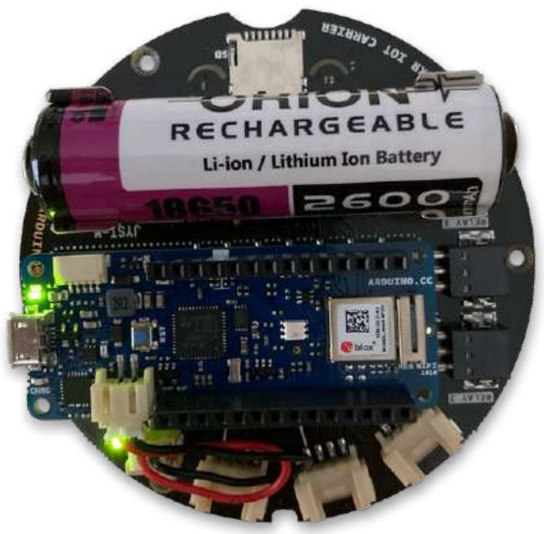


Figure 2: The Arduino Oplà IoT kit.



▲

Figure 3: The battery socket makes the Oplà IoT Kit portable.

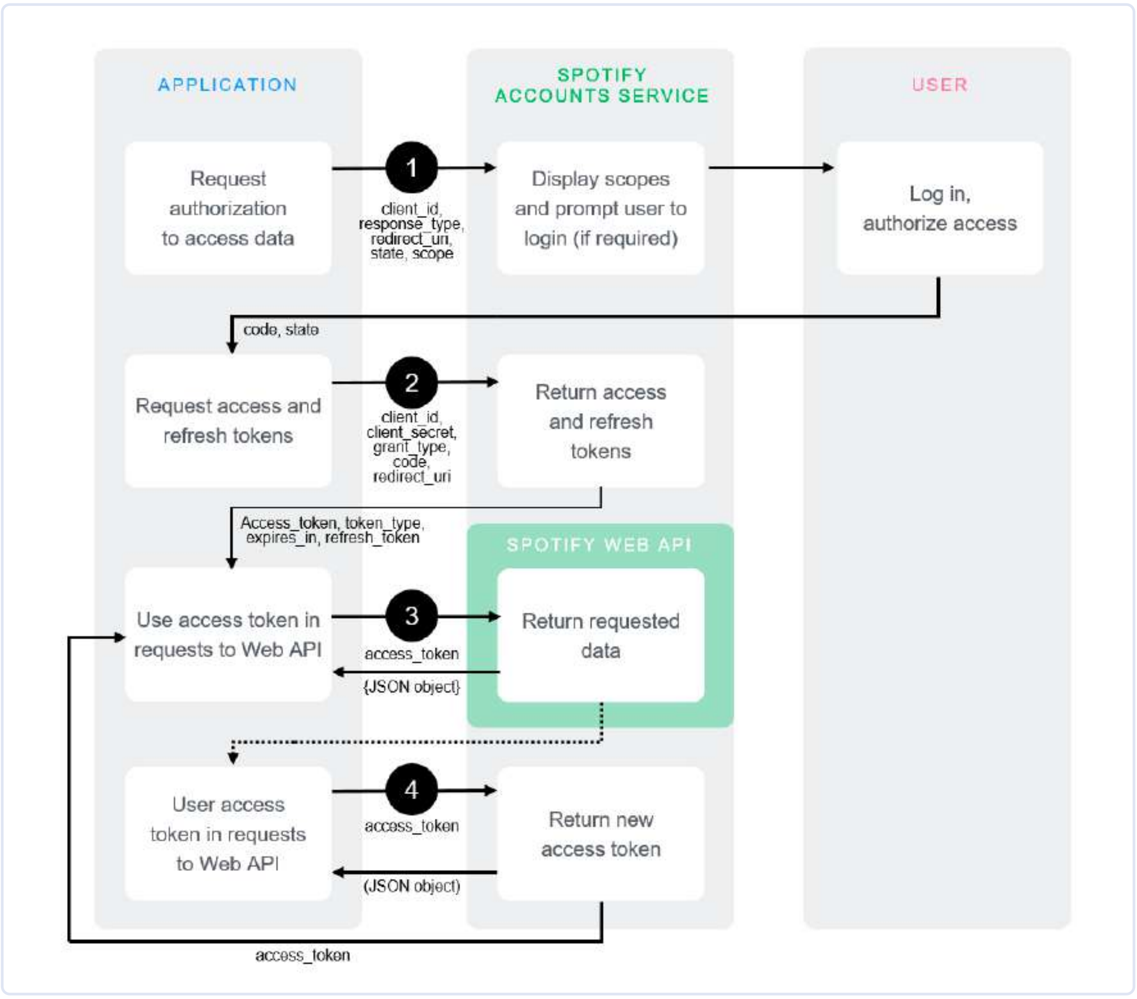
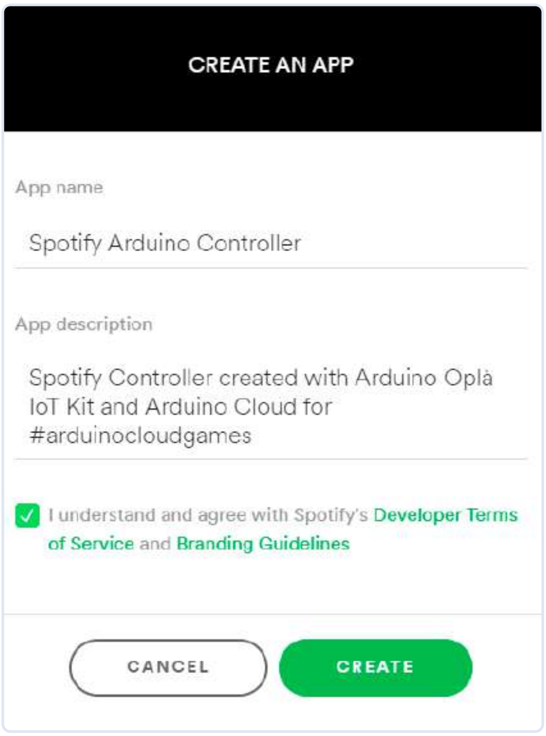

APPLICATION
SPOTIFY ACCOUNTS SERVICE
USER
Request authorization to access data
1
client_id, response_type, redirect_uri, state, scope
Display scopes and prompt user to login (if required)
Log in, authorize access
code, state
Request access and refresh tokens
2
client_id, client_secret, grant_type, code, redirect_uri
Return access and refresh tokens
Access_token, token_type, expires_in, refresh_token
SPOTIFY WEB API
Use access token in requests to Web API
3
access_token
{JSON object}
Return requested data
User access token in requests to Web API
4
access_token
{JSON object}
Return new access token
access_token

Figure 4: The multi-step-authentication flow is based on the popular OAuth 2.0 process.

◀


CREATE AN APP

App name

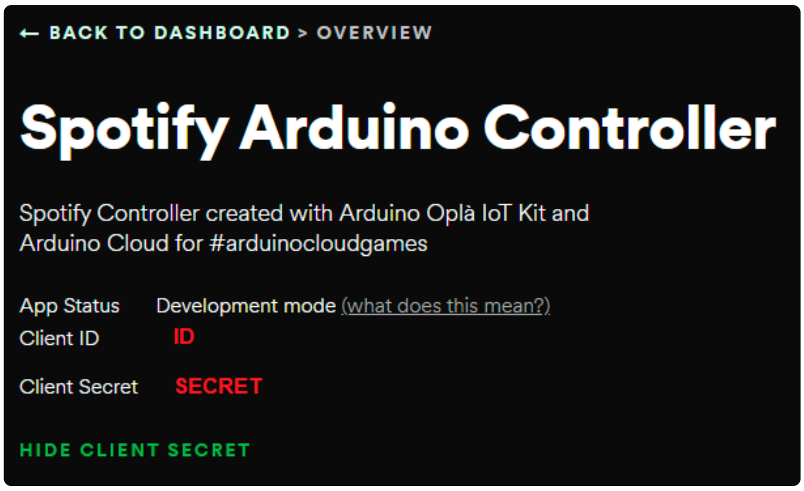
Spotify Arduino Controller

App description

Spotify Controller created with Arduino Oplà IoT Kit and Arduino Cloud for #arduinocloudgames

✓ I understand and agree with Spotify's Developer Terms of Service and Branding Guidelines

Figure 5: You have to create an "App" to get ...



Spotify Arduino Controller

Spotify Controller created with Arduino Oplà IoT Kit and Arduino Cloud for #arduinocloudgames

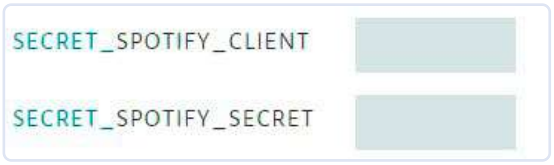
App Status Development mode (what does this mean?)
Client ID ID
Client Secret SECRET

Figure 6: ...your Client ID and Client Secret.

How do you get your Client ID and Secret? Just use the Spotify App Builder [1], which you can use to design your own PC software or mobile App to control Spotify (**Figure 5**). However, we don't do this here; we just want the credentials (see **Figure 6**). Client ID and Secret must be stored on our Arduino MKR Board. Of course, you could do this hardcoded in the Arduino sketch, but there is a more comfortable and more secure way to do so. The Arduino Web Editor [2] provides a *Secrets* tab, where you can set environmental variables to be later used in your code (**Figure 7**). Just enter the Spotify Client ID and the Secret as well as your Wi-Fi network name and password in the fields of the tab. If you compile and upload the software to the controller, your individual secret values will be also uploaded to be used by the project's code. In your sketch you have to replace the strings containing sensitive data by writing a `SECRET_xxx` expression — so, for example: `SECRET_SPOTIFY_CLIENT`.

Authentication

To start the OAuth2 flow, you have to authenticate at Spotify. When starting the Spotify controller described here, it will log in the specified (home) Wi-Fi network and show the IP address it got by the router on the OLED display. I wanted to give the user a chance to easily authenticate at Spotify, so I created the following approach. The Arduino Controller generates a


SECRET_SPOTIFY_CLIENT

SECRET_SPOTIFY_SECRET

Figure 7: Enter all private values in the *Secrets* Tab of the Arduino Web Editor, before compiling and uploading the code.

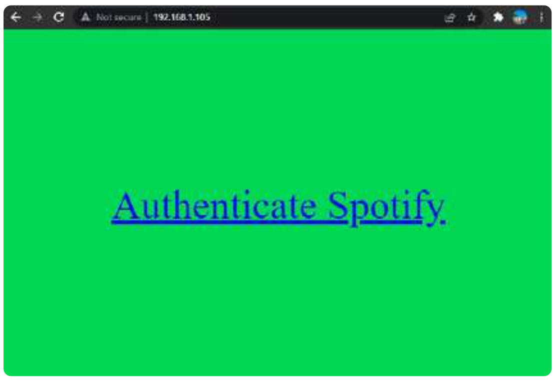


Figure 8: Webpage offered by the controller to log in at Spotify.

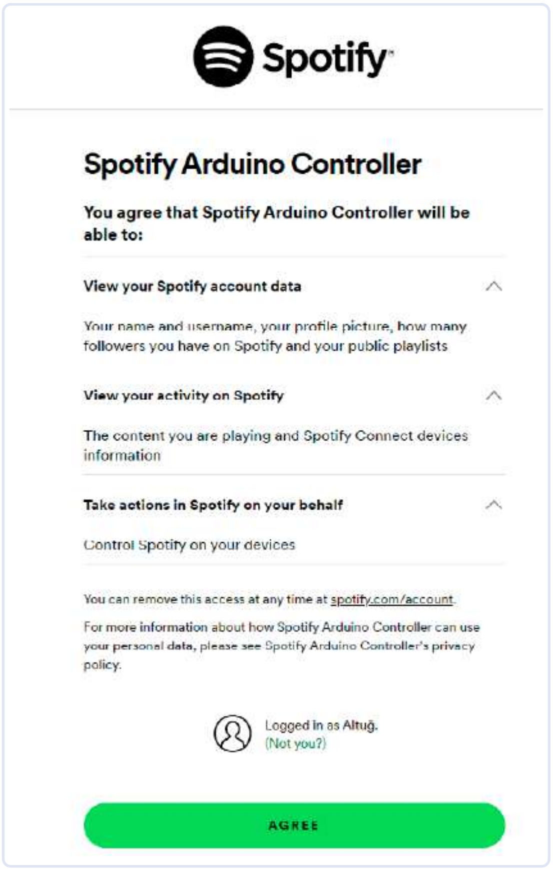
small webpage which will be shown in a web browser, when you enter the IP address of your controller there (**Figure 8**). This small webpage contains a weblink. (Refer to **Listing 1** to see how the webpage is generated in the Arduino code.) If pressed, the browser goes to the authentication page at Spotify, where you easily can log in. You will then be asked if you give the controller permission to control Spotify (**Figure 9**).

Please note: To get all this working, you also have to enter the IP address of the controller as a "*Redirect URI*" in the Spotify App editor (**Figure 10**).

From now on the Arduino Controller can get the API access token with sending the Client ID and Secret to Spotify (**Listing 2**). The access token must be regularly refreshed during operation. This is also done by a function in the sketch (**Listing 3**), which is called every 3000 seconds.

Listing 1: Webpage to authenticate at Spotify, offered by the Spotify controller.

```
String webpage = "<!DOCTYPE html>\n";
webpage += "<html><body>";
webpage += getStyle();
webpage += "<a href=\"https://accounts.spotify.com/authorize?client_id=";
webpage += SPOTIFY_CLIENT;
webpage += "&response_type=code&redirect_uri=http://";
webpage += ip_address;
webpage += "/redirect/&scope=user-read-playback-state user-modify-playback-state\">Authenticate Spotify</a>\n";
webpage += "</body></html>";
wifiClient.print(webpage);
```


Spotify

Spotify Arduino Controller

You agree that Spotify Arduino Controller will be able to:

View your Spotify account data

Your name and username, your profile picture, how many followers you have on Spotify and your public playlists

View your activity on Spotify

The content you are playing and Spotify Connect devices information

Take actions in Spotify on your behalf

Control Spotify on your devices

You can remove this access at any time at spotify.com/account.

For more information about how Spotify Arduino Controller can use your personal data, please see Spotify Arduino Controller's privacy policy.

Logged in as Altuğ.
(Not you?)

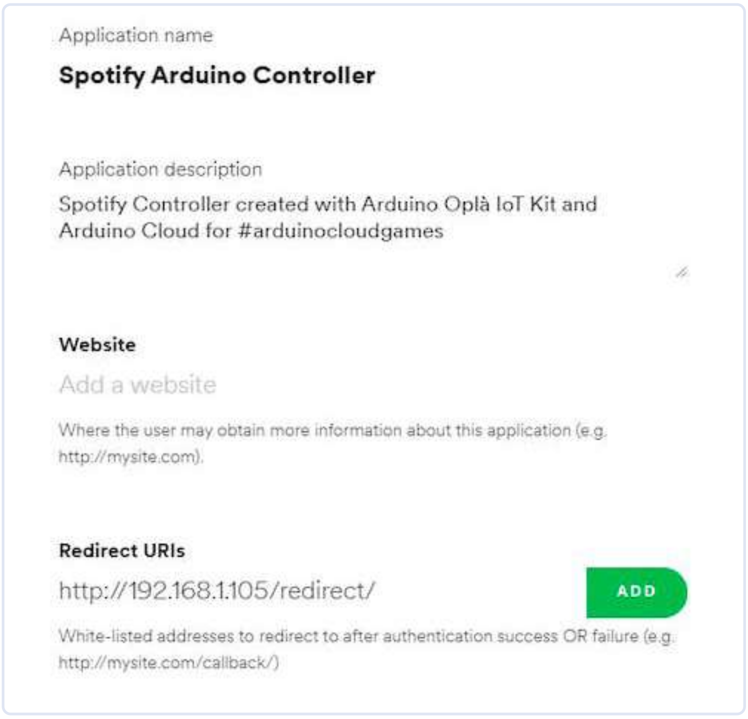
AGREE

Figure 9: If you are logged in at Spotify, you have to give the controller permission to act on your behalf.

◀

Figure 10: Redirect URL: The address of the controller in your home network.

▼


Application name

Spotify Arduino Controller

Application description

Spotify Controller created with Arduino Oplà IoT Kit and Arduino Cloud for #arduinocloudgames

Website

Add a website

Where the user may obtain more information about this application (e.g. http://mysite.com).

Redirect URIs

http://192.168.1.105/redirect/ ADD

White-listed addresses to redirect to after authentication success OR failure (e.g. http://mysite.com/callback/)

Listing 2: Function to get the token from Spotify for further use of the API.

```
// Get the user authorization token
 bool getAccessToken(String userCode) {
   String postData = "grant_type=authorization_code&code=" + userCode + "&redirect_uri="
                     "http://" + ip_address + "/redirect/";
   authClient.beginRequest();
   authClient.post("/api/token");
   authClient.sendHeader("Content-Type", "application/x-www-form-urlencoded");
   authClient.sendHeader("Content-Length", postData.length());
   authClient.sendBasicAuth(SPOTIFY_CLIENT, SPOTIFY_SECRET);
       // send the client id and secret for authentication
   authClient.beginBody();
   authClient.print(postData);
   authClient.endRequest();

  // If successful
   if (authClient.responseStatusCode() == 200) {
     lastTokenTime = millis();
     DynamicJsonDocument json(512);
     deserializeJson(json, authClient.responseBody());
     accessToken = json["access_token"].as<String>();
     refreshToken = json["refresh_token"].as<String>();
     return true;
   }
   return false;
 }
```

Listing 3: Function to refresh the token.

```
// Refresh the user authentication token
 void refreshAccessToken() {
   String postData = "grant_type=refresh_token&refresh_token=" + refreshToken;
   authClient.beginRequest();
   authClient.post("/api/token");
   authClient.sendHeader("Content-Type", "application/x-www-form-urlencoded");
   authClient.sendHeader("Content-Length", postData.length());
   authClient.sendBasicAuth(SPOTIFY_CLIENT, SPOTIFY_SECRET);
       // send the client id and secret for authentication
   authClient.beginBody();
   authClient.print(postData);
   authClient.endRequest();

  // If successful
   if (authClient.responseStatusCode() == 200) {
     lastTokenTime = millis();
     DynamicJsonDocument json(256);
     deserializeJson(json, authClient.responseBody());
     accessToken = json["access_token"].as<String>();
   }
 }
```

Listing 4: Example for using the API (next and previous song).

```
// Skip a song towards a given direction
 void skipSong(String direction) {
   apiClient.beginRequest();
   apiClient.post("/v1/me/player/" + direction);
   apiClient.sendHeader("Content-Length", 0);
   apiClient.sendHeader("Authorization", "Bearer " + accessToken);
   apiClient.endRequest();
 }
```

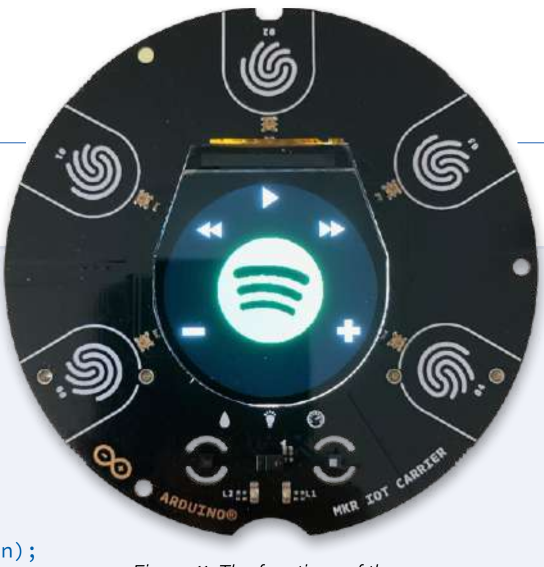


Figure 11: The functions of the buttons are shown on the display.

Operation

The rest of the code is less complex. The device will show the Spotify logo and the function of the buttons on the OLED display (**Figure 11**). If the user touches a button, the corresponding API function is called. Refer to **Listing 4** to see how this is done for skipping a song to the previous or next one.

There is also a function in the code which requests the status of the player from the Spotify API. The answer is a JSON string. I am using the *ArduinoJson.h* library and some of my own functions to process JSON strings more easily.

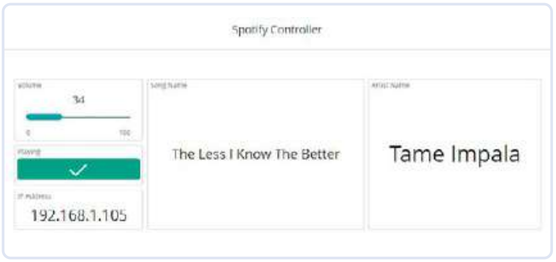
To get the status of the buttons, to control the LEDs, and to show graphics on the OLED, I am using the *Arduino_MKRIoTCarrier.h* library. You can dive into my code to get inspiration for your own projects you can do with the Oplà Kit. My software can be downloaded at [3].

Cloud Connection

I also set up a connection to the Arduino Cloud and created a dashboard that shows the current song and artist name next to the volume of the device (**Figure 12**). Of course you can create your own personal dashboard, with the data you want.

My project won the 3rd place award in the Arduino Cloud Games 2022!

220407-01


Spotify Controller

Volume 34 (0–100)

Playing ✓

IP Address 192.168.1.105

Song Name: The Less I Know The Better

Artist Name: Tame Impala

Figure 12: Current song and artist name are sent to the Arduino Cloud, where they are visible on your personal dashboard.

Questions or Comments?

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

About the Author

Altuğ Bakan has been working as an electronics engineer, mostly with embedded systems. He loves to use Arduino in his work for rapid prototyping and ease-of-use. His favorite electronics subjects are bare-metal embedded programming and Internet of Things (IoT).

Related Products

- **Arduino Oplà IoT Kit**
  www.elektormagazine.com/arduino-opla-iot-kit

WEB LINKS

[1] Spotify App Builder: https://developer.spotify.com/dashboard/
[2] Arduino Web Editor: https://create.arduino.cc/editor
[3] This Project on create.arduino.cc: https://create.arduino.cc/projecthub/Altug/opla-spotify-controller-6e7bc4

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

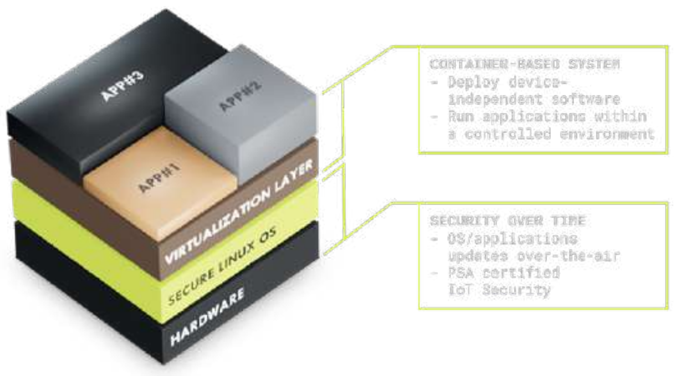
Contributed by NXP Semiconductors

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

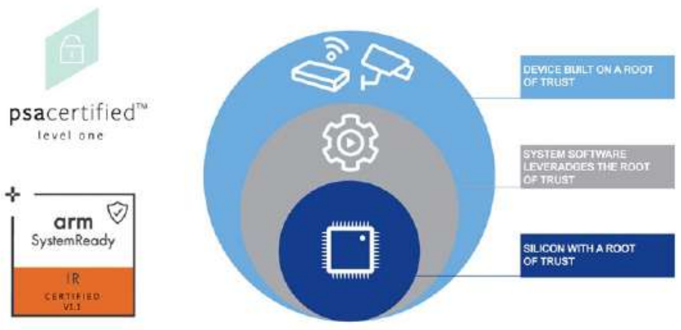
The container-based system integrates different layers of security starting from the hardware layer which includes NXP's Secure Element. It utilizes the cloud-based DevOps platform from Foundries.io [1] to reinvent the way embedded Linux solutions are built, tested, deployed and maintained. The Portenta 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.


APP#3
APP#2
APP#1
VIRTUALIZATION LAYER
SECURE LINUX OS
HARDWARE
CONTAINER-BASED SYSTEM
- Deploy device-independent software
- Run applications within a controlled environment
SECURITY OVER TIME
- OS/applications updates over-the-air
- PSA certified IoT Security

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-native 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.


psacertified™
level one
arm
SystemReady
IR
CERTIFIED
V1.1
DEVICE BUILT ON A ROOT OF TRUST
SYSTEM SOFTWARE LEVERAGES THE ROOT OF TRUST
SILICON WITH A ROOT OF TRUST

Platform Security Architecture.

EdgeLock SE050 – A Trust Anchor for IoT

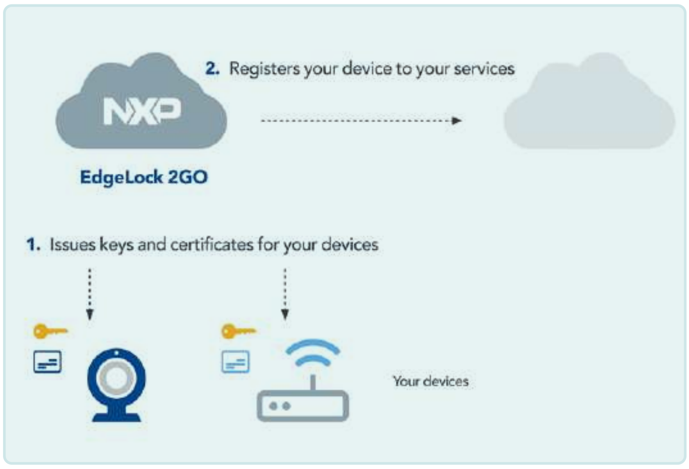
NXP's EdgeLock SE050 [3] is a discrete and tamper-resistant security hardware for protecting the identity of a device, including cryptographic keys and certificates. It's a standalone embedded secure element that is attached to the main processor over the I2C interface. The EdgeLock SE050 is certified 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.


2. Registers your device to your services
NXP
EdgeLock 2GO
1. Issues keys and certificates for your devices
Your devices

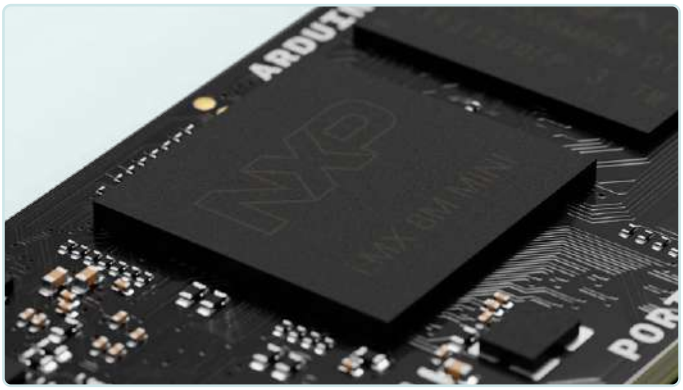
NXP Manages Device Credentials.

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.

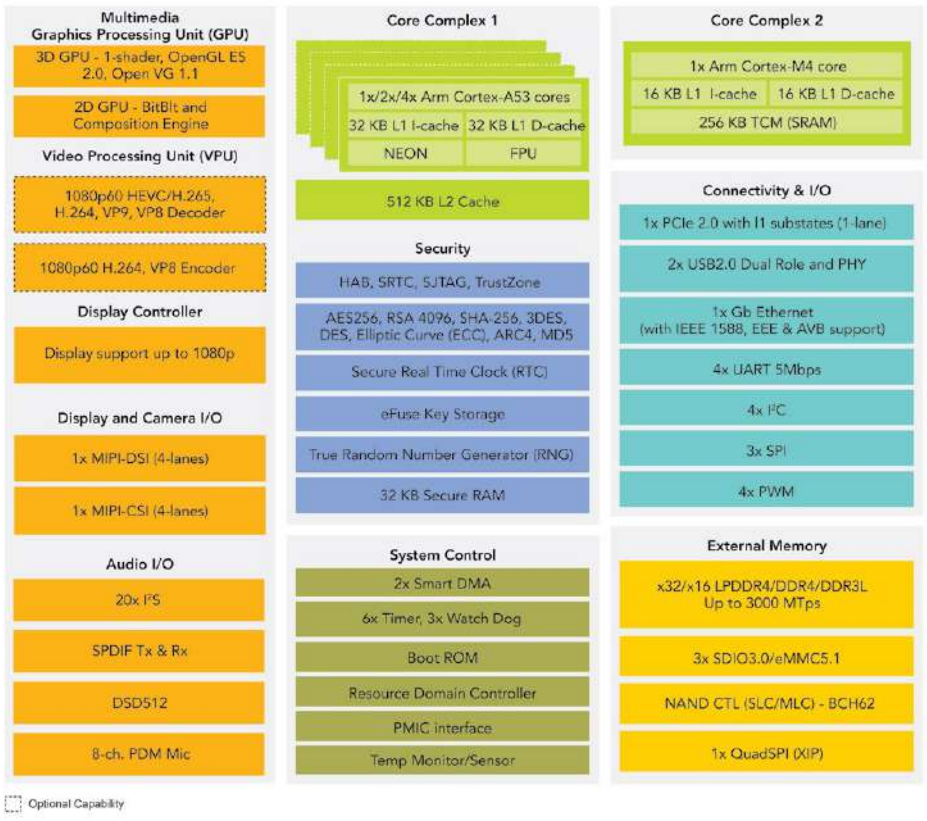
i.MX 8M Mini core options are optimized for ultra-low-power, even sub-Watt in specific applications, but offer the breadth of processing power necessary for consumer, audio, industrial, machine learning training and inferencing across a range of cloud providers. The i.MX 8M Mini SoC also packs-in hardware 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


Multimedia
Graphics Processing Unit (GPU)
3D GPU - 1-shader, OpenGL ES 2.0, Open VG 1.1
2D GPU - BitBlt and Composition Engine
Video Processing Unit (VPU)
1080p60 HEVC/H.265, H.264, VP9, VP8 Decoder
1080p60 H.264, VP8 Encoder
Display Controller
Display support up to 1080p
Display and Camera I/O
1x MIPI-DSI (4-lanes)
1x MIPI-CSI (4-lanes)
Audio I/O
20x I²S
SPDIF Tx & Rx
DSD512
8-ch. PDM Mic
Core Complex 1
1x/2x/4x Arm Cortex-A53 cores
32 KB L1 I-cache
32 KB L1 D-cache
NEON
FPU
512 KB L2 Cache
Security
HAB, SRTC, SJTAG, TrustZone
AES256, RSA 4096, SHA-256, 3DES, DES, Elliptic Curve (ECC), ARC4, MD5
Secure Real Time Clock (RTC)
eFuse Key Storage
True Random Number Generator (RNG)
32 KB Secure RAM
System Control
2x Smart DMA
6x Timer, 3x Watch Dog
Boot ROM
Resource Domain Controller
PMIC interface
Temp Monitor/Sensor
Core Complex 2
1x Arm Cortex-M4 core
16 KB L1 I-cache
16 KB L1 D-cache
256 KB TCM (SRAM)
Connectivity & I/O
1x PCIe 2.0 with l1 substates (1-lane)
2x USB2.0 Dual Role and PHY
1x Gb Ethernet (with IEEE 1588, EEE & AVB support)
4x UART 5Mbps
4x I²C
3x SPI
4x PWM
External Memory
x32/x16 LPDDR4/DDR4/DDR3L Up to 3000 MTps
3x SDIO3.0/eMMC5.1
NAND CTL (SLC/MLC) - BCH62
1x QuadSPI (XIP)
Optional Capability

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

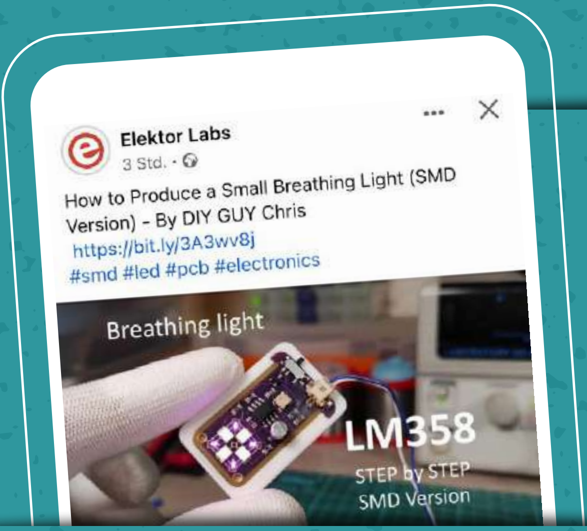
New *Arduino* or *Electronics* Project?

Share it with our community!

Elektor @Elektor · 4T
NEW 🤩🔈 Assembling the Elektor Fortissimo-100 Audio Power Amplifier
@Clemens_Elektor
bit.ly/3tORi8y
#amplifier #audio #power #electronics

Follow us on:

www.elektor.com/TW



Elektor Labs
3 Std. ·
How to Produce a Small Breathing Light (SMD Version) - By DIY GUY Chris
https://bit.ly/3A3wv8j
#smd #led #pcb #electronics

www.elektor.com/Intsta

www.elektor.com/FB

Declassified Bonus Edition!

Available in week 52

▲

Daria Ba
and her A
colleague

diffi
cost
imp

The
bec
pre
abi
is a
del
at t
las
tho
is s
co
wh
it s
in
pl
de
ar
si
bl
th

K
a
w

I
v
s
r
v
t

Declassified
Bonus Edition!

Available in week 52

Declassified Bonus Edition!

Available in week 52

Elektor
Webinars

FEB
2023
16:00 CET

MicroPython Enters the World of Arduino

with Stuart Cording & Sebastian Romero

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? Stuart Cording will speak with Sebastian Romero (Head of Content, Arduino) during our live webinar to find out more.

Join for free

www.elektor.com/webinar-MicroPython

× elektor

Declassified Bonus Edition!

Available in week 52

...mponents like lights, small motors, etc. An SBC

...quickly adopted by a new category of electronic designers that included enthu-

siast
foun
micr
2008
a lov
opm
Pi [
Bea
inte
cou
of F
and
pro

Pr

To
eta
de
su
an
in
cc
ac
sc
ar
ar

T
v
p
I
a
D
s
r

Declassified Bonus Edition!

Available in week 52

ARDUINO × elektor

its features, it would
their microcontroller and ancillary parts for use in initial

Declassified Bonus Edition!

Available in week 52

▲

Figure
Maxim

Figure 3
Logic Co
Shutters

▼

requirements of indus-
trial products.

On t
supp
suita
subj
for i
do r
thei

Wh
the
boa
Pro
tes
bu
are
for
at
be
tes
ar
wi

As
de
to
p
te
fo
o

Declassified Bonus Edition!

Available in week 52

ARDUINO × elektor

ns,
in

ne-
irs
ics

[7] Nordic Semiconductor

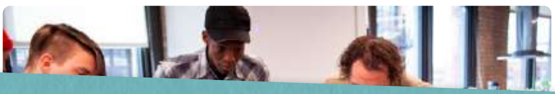
Declassified Bonus Edition!

Available in week 01

Figure 1
of a 'Cre
Flora' w

…from paper and other thin materials. Recently, ative symphony!

could create a gener-



The
of b
ing
con
mer
law
Niti
cov
tran
bor

Ea
the
Th
the
de

Yo
su

Declassified Bonus Edition!

Available in week 01

ARDUINO × elektor

for

Declassified Bonus Edition!

Available in week 01

ere to "sing" together by outputting audible

it to a set of samples
in the frequency domain (frequency vs. amplitude). This

math
and
get
Zerc
pull
Ada
the
look
mat

Mc

You
Fig
The
You
the
Wh
pe
cu

Be
ta

N
th
ur
th
p

T
tl
n
c
c
"
t
c

Declassified
Bonus Edition!

Available in week 01

ARDUINO × elektor

a

f
mory
cle.

the petals are still opening).

Declassified Bonus Edition!

Available in week 01

Fig

AR

Figure 9: L

SWD pins

Thes
— in

Bec
a pr
out

We
ins
Nit
to r
nu

De
wa
Le

C
W
W
nu
er
pl

Declassified Bonus Edition!

Available in week 01

er

CRT analog X-Y oscilloscopes. You can find him on Instagram at @ideo and @vondle_ synths.

Get your hands

Declassified Bonus Edition!

Available in week 01

A

p

The
Bra
Bra
arm
Ard
bled
task
mou
your
pane
the s
mult
from
Brac
new
totall

www.

Ardu

your projects from anywhere in the world.

www.elektor.com/19351

Declassified Bonus Edition!

Available in week 01

own UNO-powered synth!

www.elektor.com/20330

Arduino Sensor Kit Base

www.elektor.com/19944

tions (AI).

www.elektor.com/19936

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.