

NORBERT MORAWSKI, M. SC.

IT Master of Science / Cloud engineer

 norbert@mrwski.eu  795-723-579  MrJake222  mrwski.eu  Kraków

EXPERIENCE

Embedded Software Intern

Nordic Semiconductor

 Jul 2025 - Sep 2025  Kraków

Letter of Recommendation available below.

- Worked on both older nRF52 and the newest nRF54 chips,
- Assisted in developing Zephyr-based tests,
- Helped to maintain the embedded `nxp` drivers,
- Debugged firmware, written Python automation scripts,
- C programming with a professional review process and CI,
- Linux workstation with bash scripting,
- Active part of an Agile team.

Content Editor

Tutor Maurycy Gast

 Oct 2021 - Nov 2022  Remote

- Edited IT-oriented textbook for Official Polish Government e-Learning Platform,
- Revised texts for publishing, leader of the QA team,
- Provided professional expertise in C++, Java, and Python,
- Responsible for training new employees,
- Fully remote workflow.

Java Developer

Jotafan Andrzej Zagórski

 Sep 2018 (internship)  Kraków

- Soldered & tested hardware modules,
- Developed Android application – Modbus Bluetooth controller.

PUBLICATIONS

Estimation of Distribution Algorithms with Overlapped Subpopulations

DOI: 10.1007/978-981-96-6005-6_13

 April 2025  AGH

- Developed a solver in Python for the new algorithm,
- Overlapped subpopulations gave 8% better results.

LANGUAGES

English C1 (ACERT)



Polish Native



For personal projects see the second page.

SKILLS

- Cloud basics from **AWS Foundations & Architecture** courses,
- Skilled in **Git/GitHub** workflows, **Make/CMake** build systems as well as **West** and **Ninja**,
- Hands-on debugging using professional tooling such as **GDB**.
- Fluent in **Linux** environment (daily-driver, home + work),
- Exposure to **Jenkins** and GitHub actions integration, experience in **expanding** the pipeline,
- Agile development practices using **Jira**, **Confluence** and code reviews.
- Analytical problem-solving, communication, and adaptability.

Additional Skills

- Experience in **embedded C** programming, on various IoT platforms, both in bare metal and **RTOS** environments,
- Familiar with Bluetooth Low Energy (**BLE**) protocols,
- Familiarity with **VSCode** and **CLion** IDEs,
- Proficient in **Python** and **Bash** scripting for test automation and tooling,
- Experienced in creating technical documentation with **LaTeX** and **Markdown**,
- Practical skills including soldering, PCB design in **KiCad**, and 3D printing for prototyping.

EDUCATION

M.Sc. in Computer Science

AGH University

 2024 - 2025

Thesis: Enhancing signal processing for on-device machine learning with auto-generated RISC-V cores

B.Sc. in Computer Science

AGH University

 2020 - 2024

Thesis: Receiver of Internet radio stations

Upper Secondary School

Zespół Szkół Łączności w Krakowie

 2016 - 2020

IT Technician

PROJECTS

Full portfolio with broader descriptions, GitHub links and photos available at mrwski.eu/projects.
An export of the portfolio is also attached to this document.

DHCP, DNS manager

C++ automation script

- Built a service management tool with IPv6 support,
 - Automatically generates configs for DHCP/DNS/firewall daemons from YAML,
 - Reduced manual setup by automating network infrastructure.
-

Home Server with Automation

Linux, Docker, nginx, Authelia, MariaDB

- Set up a cloud-like infrastructure for hosting services and applications,
 - Implemented secure authentication and reverse proxying,
 - Deployed databases and web applications using Docker.
-

RGB LEDs 4.0

ESP8266 Wi-Fi LED controller / C++

- Wireless lighting control designed for home automation,
 - Product based on shortcomings of previous versions,
 - PCB professionally manufactured by JLCPCB,
 - Currently 6 deployed modules around the house,
 - Created a web app for modules' management and control.
-

Master thesis: RISC-V survey & enhancement

iCE40 FPGA / Verilog, C

- Deployed several RISC-V cores (including PicoRV32, DarkRiscV, and Hazard3) to FPGA,
 - Measured Dhrystone scores, the best core was selected for further modification,
 - Profiled a TinyML framework for the bottleneck,
 - Designed a hardware-software interface, modified the algorithm to use it,
 - Achieved performance was 5x with reduced power consumption,
 - Got involved in open-source development of the `pico-ice-sdk`,
-

Engineering thesis: Pico Radio

RP2040 Internet radio / C++, KiCad

- Li-ion power, with a built-in charger and voltage sensing,
 - Built-in stereo speakers with 2x3W amplifier,
 - Designed schematic, PCB and 3D-printed enclosure,
 - Multicore architecture, 2nd core used for MP3 decoding,
 - FreeRTOS, debugged with OpenOCD, developed with CLion.
-

Letter of Recommendation

To Whom It May Concern,

I am pleased to recommend Norbert Morawski for the position of Firmware Engineer. During his internship, I was consistently impressed by his skills, work ethic, and character.

Norbert excels in technical knowledge, particularly in embedded systems and the C programming language. He was tasked with a major rework of Nordic's driver library and performed exceptionally well. His involvement in architectural discussions demonstrated an additional strength, which was highly unexpected from a student. He proved to be very familiar with all tools used in software development for embedded workflows. Additionally, he quickly adapted to Nordic's codebase (including Zephyr RTOS) with minimal introduction.

Beyond his professional excellence, Norbert's workplace conduct has been impeccable. His dedication and proactive approach were greatly appreciated. He exceeded my expectations during his internship, and I am confident that he will excel in his new role as well.

Please feel free to contact me at marcin.szymczyk@nordicsemi.no or +48 509 542 543 for further information.

Sincerely,
Marcin Szymczyk
Staff R&D Engineer
Nordic Semiconductor

Projects

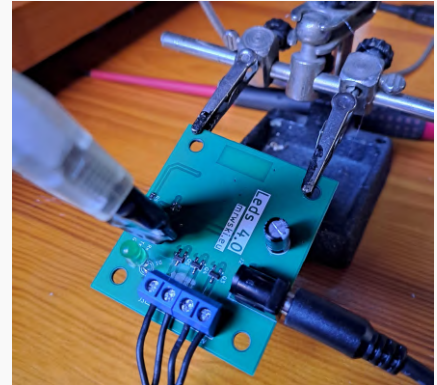
RGB LEDs 4.0

GitHub

This is a rework of my previous attempts at making RGB lighting controller. Bluetooth-based version 1 and 2 were difficult to control from PC. RS485 version 3 didn't survive room rearrangement (not enough cabling). This version 4 is based on ESP8266, controlled via Wi-Fi, and integrated with Ethernet wired network to provide easy controlling from all over the house.

On separate host, I've developed and deployed Linux/Nginx/MariaDB/PHP web-app for creating module groups and profiles. It supports user authentication via [Authelia](#) (each user gets to create it's own modules, lists and profiles).

This version is also the first one to have it's PCB professionally manufactured (KiCad/JLCPCB) and 3D printed enclosure. Each RGB strip got it's own controller brick, so it was unfeasible to make each board/case by hand.



My first professionally manufactured PCB!

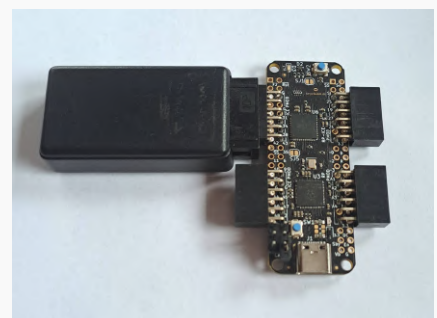
RISC-V processor survey on iCE40 FPGA (2024)

GitHub

PDF

I worked on surveying a couple of implementations of the RISC-V processor on low-power iCE40 FPGA with open-source toolchain. I've tested [PicoRV32](#) and [VexRiscV](#), reaching 10.14 and 12.61 DMIPS respectively. I will be writing my master thesis about ways to expand one of these cores with custom DSP blocks.

Pico-ice proved to be a versatile and useful board. It features a RP2040 chip, which I'm already familiar with (eng. thesis). The chip can be programmed to act as multitude of different USB devices. I love exploiting this hybrid approach. The board can, for example, simulate multiple UART interfaces. Separate USB-TTL adapters are not required. However the firmware is still work-in-progress. I got voluntarily involved in it's development. A couple of pull request were created (some approved) for both [pico-ice-sdk](#) and [pico-ice_default](#)



pico-ice + logic analyzer testbed

[firmware.](#)

Pico Radio (2023 - 2024)

GitHub

PDF

My engineering thesis. I made internet radio from scratch using Raspberry Pi Pico W (RP2040, dual-core Cortex M0+, Wi-Fi).

Tech stack: C++, [pico-sdk](#), [helixmp3](#), [freertos](#), [fatfs + sdlib](#), [littlefs](#).

Other libraries were written by hand. I made a couple of pull requests during this project: [1](#), [2](#), [3](#), [4](#), [5](#). Some even got approved.

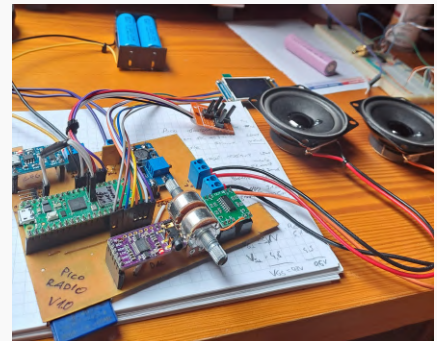
Features:

- Playback of stereo radio stations (MP3 format),
- search functionality based on [radio-browser.info](#),
- favourite station list (saved to flash),
- local MP3/WAV files support (SD card),
- concise user interface planned in advance.

Project uses:

- DMA, for raw audio data transfer,
- PIO blocks, for I²S support,
- two cores of the MCU, one used exclusively for MP3 decoding.

Project was aiming quite high, but all planned objectives were met. Hardware-wise, a homemade PCB was fabricated and I designed the enclosure and 3D printed it.



Final prototype



Finished product in enclosure

DHCP, DNS manager (2023)

GitHub

I developed a C++ program to generate configs for services managing network hosts. This being [DHCP](#), [DNS](#) and [nftables](#). See readme on GitHub for more details.

Features:

- YAML config file, multiple subnets,
- A, AAAA, CNAME, MX, NS, PTR records,
- port forwarding,
- IPv6.

```
File: eth0.yaml
1  hosts:
2    net-isp-router-gateway:
3      name: gateway
4      ipv4: .0.1
5
6    net-ddmgr-dpi:
7      name: dpi
8      ipv4: .0.4
9
10   lan-bpi-nas:
11     name: bpi
12     mac: 02:53:ca:xx:xx:xx
13     ipv4: .1.1
14     aname:
15       - '@'
16       - ns1
17     cname:
18       - mail
19       - svnc
```

It proved useful to manage a small home network. I can assign addresses to hosts and then use easy-to-remember DNS names without having to manually write all those config files.

```
19 - sync
20 - vpn
21 mx:
22   for: '@'
23   pri: 10
```

Example config file

6502-based computer on FPGA (2023)

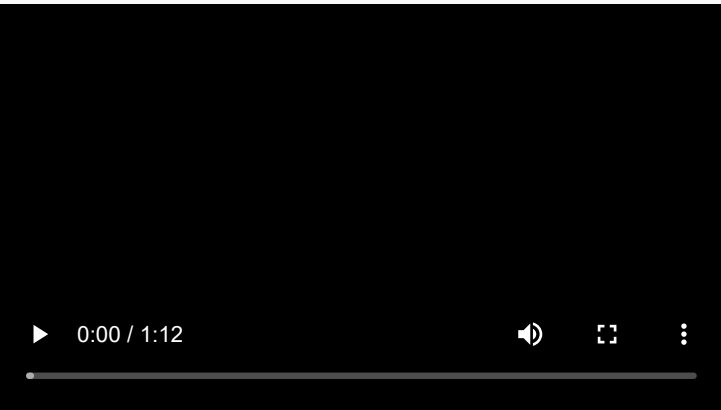
- GitHub 6502
- GitHub UART
- PDF

The main trouble I challenged myself to overcome was to implement 6502 core in Verilog by myself. Also, I created debug UART interface and testing infrastructure for it in Python.

Features:

- Working 6502 core written in Verilog,
- Debug UART interface with memory and register access,
- PS/2 keyboard interface

I also ported [Wozmon](#) to my hardware.



Heating unit web app (2022)

- GitHub

ESP8266-based web app for managing central heating unit. Written using Arduino ecosystem (needed to be functional ASAP). Uses closed protocol to communicate with the unit, decoded with the help of [elektroda](#) folks.

Features:

- reading current temperatures,
- setting target temperatures,

Temperature

Kocioł CO

28°C

Aktualna

55°C

Zadana

Woda użytkowa

29°C

Aktualna

50°C

Zadana

Zawór podłogowy

- turning on a circulation pump via relay,
- start/stop the heating,
- responsive design for mobile devices.

Not a big fan of web development, but I like to setup a simple page from time to time. This one was written using HTML, CSS, and vanilla JS. Server side uses C++ to deliver JSONs to the frontend, which acquires it with async fetches.

Bete/geza is a codename for a bigger project that sadly never came to life. I was aiming for a mobile app someday, the API was written to future-proof the design. Currently, only the web frontend is available.

[First version](#) of this used MicroPython, but had nasty memory related issues. Decided to rewrite it in C++ instead.

29°C Aktualna 30°C Zadana

[Temperatures preview](#)

Adaptive Huffman coding in C++ (2022)

GitHub

Advanced C++ uni final project. It implements adaptive Vitter algorithm (no need to store dictionary nor read the file twice). Effective for small and large files ([results](#)). Unit test infrastructure in place. See github readme for more details.

Features:

- Own BitArray implementation (extensible array of bits implemented as a template over numeric types),
- Own LinkList implementation as a template,
- Own progress printer (simple ASCII one),
- [CLI11](#) command line options parser,
- [Google Test](#) for testing BitArray class.

6502-based computer (2020)

PDF

Asm code

Inspired by [Ben Eater's](#) videos, I decided to make my own version of the 6502-based computer. 65C02 was a single-chip sourced from WDC. I also used 65C51 from them as UART interface. All other logic was done via TTL logic chips. I designed a VGA graphics card and a PS/2 mouse interface. Then, programmed it in assembly using [vasm](#).

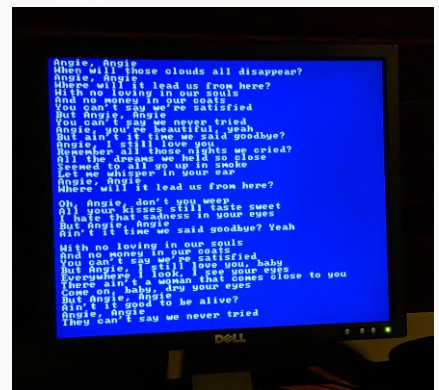
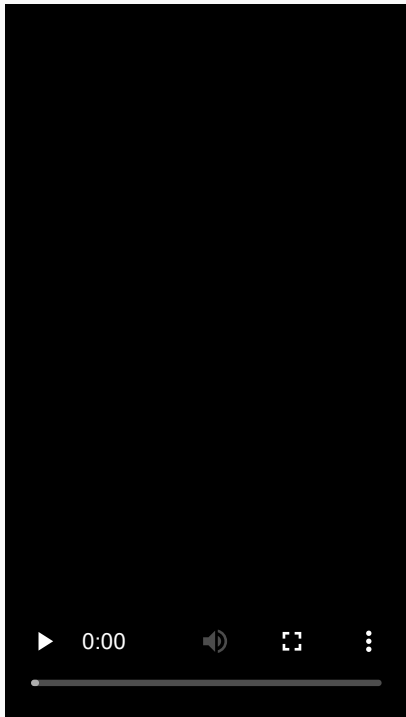


[Computer with monitor and keyboard](#)

Got a real grasp of pointers by using indirect addressing modes. Computer supports loading code via UART bootloader. Description was made 2 years later for Digital Systems uni course.

Features:

- 32KiB ROM, 16KiB RAM, rest for MMIO
- 50x37 character mode monochrome graphics (VGA connector)
- PS/2 keyboard controller with extended and break codes support,
- basic shell parsing with argument support,
- custom 16-bit floating point (addition, printf)



Text mode test (8x8 IBM font)

AUNIS (2018 - 2021)

GitHub

All You Need Is Stargate.

[Minecraft mod](#). First such a big project, released to the public, downloaded almost 60K times. Written in Java. It required some OpenGL magic and trig math to get all this to render properly within a block game. It once made a thriving community, that is now abandoned due to a shift of interests and lack of time. But thanks to the magic of open source, it is maintained and updated via community as [Just Stargate Mod](#). There is also a simple [web app](#) for recipe calculations.



Stargate opens!

RGB leds (2016 - 2019)

GitHub links

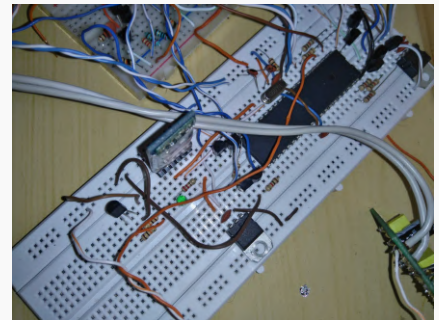
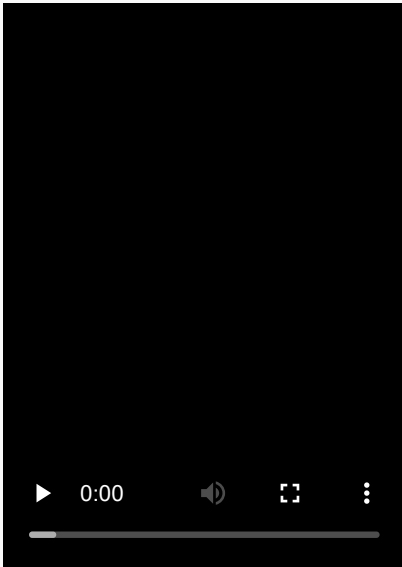
[Server](#)

[React Native](#)

[AVR](#)

[Web front](#)

The project started as an idea to control RGB strips from AVR microcontrollers. Later, Bluetooth joined the picture. The first version was absolutely unreliable. I learned the hard way why breadboards aren't any good for permanent projects. Later, it got a reboot based on soldered perfboards. Works to this day! Further, the project was redesigned with a custom PCB (made in KiCad) and RS485 interface connected to simple hub (Node.js) and controller via web or React Native app.



Breadboard version



Soldered perfboard version



Custom PCB + RS485 version

Passion from childhood (2000 - still going)