

# Paul Hondola

+40745166873 | [paulhondola@gmail.com](mailto:paulhondola@gmail.com) | [linkedin.com/in/paulhondola](https://linkedin.com/in/paulhondola) | [github.com/paulhondola](https://github.com/paulhondola)

## EDUCATION

### Bachelor of Science in Computer Engineering

Oct. 2023 – Jun. 2027

*Universitatea Politehnica Timisoara*

- Relevant Coursework: Data Structures and Algorithms, Operating Systems, Computer Architecture, Distributed Systems, Networking, Web Development.

## EXPERIENCE

### Full Stack Developer Intern

Jul. 2025 – Oct. 2025

*Hibyte* | Typescript, Angular, NestJS, Payload CMS, Supabase

- Engineered 'GameBox', a scalable full-stack monorepo for game center management, integrating a customer-facing web app with a comprehensive admin CMS.
- Built a responsive frontend architecture using Angular, TypeScript, and SCSS, ensuring high performance and maintainability.
- Developed a robust backend system using NestJS and Payload CMS, leveraging Supabase for secure authentication and real-time database operations.

### Deepblue Maker - Underwater Robotics Summer Camp

Jul. 2025 – Aug. 2025

*Hangzhou Dianzi University, China*

- Co-engineered a remotely operated underwater vehicle (ROV) capable of environmental mapping and real-time object detection.
- Mechanical: Designed and 3D-printed a custom water-sealed chassis using SolidWorks; validated buoyancy and structural integrity through physical testing.
- Embedded Systems: Programmed an STM32 microcontroller for precise thruster propulsion control and hardware interfacing.
- Computer Vision: Deployed a YOLO-based object detection model on a Raspberry Pi 5 to identify sea life from live camera feeds.

### Malware Analyst Trainee

Apr. 2025 – Jun. 2025

*Bitdefender* | Java, Python, C, x86 Assembly, IDA Pro, jadx

- Conducted static and dynamic analysis of Windows and Android malware samples to identify command-and-control protocols and persistence mechanisms.
- Reverse engineered ransomware encryption routines using IDA Pro (x86 disassembly) and jadx (Java decompilation) to understand payload behavior.
- Investigated Android system vulnerabilities and exploitation techniques, analyzing APK structures and security models in sandboxed environments.

## TECHNICAL SKILLS

**Languages:** C, C++, Python, Java, TypeScript, x86/ARM Assembly, SQL, Bash

**High Performance Computing:** MPI, OpenMP, POSIX Threads, SIMD, Shared Memory, Distributed Systems

**Systems and Tools:** Linux, Docker, Git, Meson, Make, CMake, GDB, IDA Pro, Wireshark

**Web Development:** Angular, NestJS, Supabase, Payload CMS, REST APIs, HTML5/SCSS

**Data Analysis:** NumPy, Pandas, Matplotlib, PyTorch

**Spoken Languages:** English (C1/Advanced), Romanian (Native)

## PROJECTS

### Distributed Image Processing System | C, MPI, OpenMP, Meson, Python, Assembly

Dec. 2025 – Jan. 2026

- Implemented a high-performance hybrid parallel image processing engine utilizing MPI for distributed memory and OpenMP for shared memory parallelism.
- Engineered diverse parallel architectures including a standard multithreaded approach, distributed message passing, shared filesystem optimization and hybrid parallelism.
- Validated the efficiency of SIMD vectorization by inspecting generated assembly code across different compiler optimization levels, ensuring maximal hardware utilization.

- Optimized computationally intensive convolution kernels (Gaussian Blur, Ridge Detection) for processing large BMP images across cluster nodes.
- Developed an automated benchmarking suite with Python visualization to analyze strong scaling, speedup, and parallel efficiency.

#### **OpenMP Dense Matrix Multiplication** | *C, OpenMP, Meson, Python, Assembly* Oct. 2025 – Nov. 2025

- Developed a comprehensive benchmarking suite for dense matrix multiplication, implementing serial and parallel algorithms with OpenMP.
- Implemented loop permutations and cache blocking (tiling) strategies to optimize memory access patterns and minimize cache misses.
- Analyzed the impact of compiler optimization levels (-O0 vs -O3) by inspecting generated assembly code to verify SIMD vectorization usage.
- Achieved significant speedups by confirming auto-vectorization and optimizing critical loops for AVX/SSE/NEON instructions.
- Automated performance reporting with CSV exports and Python visualization (Pandas/Seaborn) to measure thread scaling and efficiency.

#### **Cross-Platform System Benchmark Suite** | *Python, NumPy, PyTorch, Docker, SQL* May. 2025 – Jun. 2025

- Architected a modular benchmarking framework to evaluate CPU, GPU, and memory performance across heterogeneous architectures (x86, ARM64/Apple Silicon).
- Implemented low-level microbenchmarks to quantify memory latency, bandwidth saturation, and cache hierarchy degradation using pointer chasing and vectorized NumPy operations.
- Integrated hardware-accelerated ML backends (PyTorch MPS for macOS, TensorFlow) to measure accelerator efficiency vs. CPU baselines.
- Engineered a Docker-based execution environment to ensure reproducibility of results across Windows, Linux (Asahi), and macOS systems.

#### **Treasure Hunt System** | *C, POSIX system calls, Clang, Make, Git* Mar. 2025 – May 2025

- Introduces an interactive shell-like CLI program to manage hunts and treasures via commands
- Uses logs to track user operations, with symlinked logs for centralized access
- Utilizes multi-process architecture and sigaction-based signal handling for inter-process communication
- Enables runtime features such as live monitoring, hunt and treasure inspection, and controlled shutdown of the monitor process

#### **SafetyMap - Community-driven Safety App** | *Java, Android, Google Maps API, Firebase* Nov. 2024 – Nov. 2024

- Interactive Map: Mark and view safety alerts using Google Maps
- User Alerts: Users can drop pins on the map to report issues such as thefts, road hazards, or other dangers
- Notifications: Real-time notifications for users approaching an area with a safety alert
- Community Trust System: Users can vote on the validity of alerts, contributing to a community trust score

#### **FPGA Video Transmission and Image Processing** | *Verilog, VHDL, Xilinx FPGA* Mar. 2024 – Jun. 2024

- Hardware based video transmission and image processing system, with camera input and display via VGA.
- Supports basic image processing and integrates with OpenCV for face recognition through UART.