MINISTRY OF EDUCATION AND RESEARCH

TECHNICAL UNIVERSITY
OF CLUJ-NAPOCA, ROMANIA
FACULTY OF AUTOMATION AND COMPUTER SCIENCE

CO-DESIGN FOR ALGORITHMS

MIP PROJECT

Team Leader: **Bogdan-Alexandru BÂRGĂOANU**

Scientific coordinator: **Prof. Eng. Ovidiu STAN**

2025

Contents

Chapter A Introduction **1**
A1 Overview of Co-Design for AI Algorithms 1
A2 Project Context 1
A3 Precise Domain Specification 2

Chapter B Project leader **3**
B1 Important scientific achievements of the Project leader 3
B2 Correspondence between the experience and the proposed project 3
B3 Curriculum Vitae 4

Chapter C The funding application **7**
C1 Motivation of the proposed theme in the current scientific context 7
C1.1 Scientific Motivation 7
C2 Objectives, methodology and work plan 8
C2.1 Concrete Objectives 8
C2.2 Methodology and Proposed Work Strategy 9
C2.3 Work Plan and Timeline 10
C3 Project feasibility: available resources and research team structure 11
C3.1 Existing Resources at the Host Institution 11
C3.2 Additional Resources to be Procured 12
C3.3 Project Team and Person-Months 12
C3.4 Adequacy of Team and Infrastructure 12
C3.5 Preliminary Results and Feasibility Evidence 13
C4 Risks and alternative approaches 14
C4.1 Potential Risks 14
C4.2 Mitigation Strategies and Alternative Approaches 14
C5 Impact and dissemination 15
C5.1 Scientific Impact 15
C5.2 Societal and Economic Impact 16
C5.3 Dissemination Strategy 16
C5.4 Exploitation and Standardization 16
C6 Requested budget 17
C6.1 Project Team and Salary Allocation 17
C6.2 Justification of Major Equipment Purchases 18

Bibliography **19**

Chapter A. Introduction

A1. Overview of Co-Design for AI Algorithms

Co-design for AI algorithms represents a multidisciplinary methodology in which algorithmic development and system architecture evolve in tandem to meet stringent performance, resource, and application requirements. Unlike traditional approaches—where AI models are designed independently of deployment constraints and subsequently shoehorned into target platforms—co-design emphasizes iterative collaboration between algorithm designers, hardware engineers, and software architects. This synergy ensures that AI solutions achieve optimal accuracy, latency, power efficiency, and reliability within their intended operational environments.

In recent years, the rapid proliferation of edge computing, embedded systems, and Internet of Things (IoT) devices has underscored the need for co-design. AI models once trained and executed on cloud servers must now operate on devices with limited memory, compute horsepower, and energy budgets. Co-design methodologies address this challenge by jointly optimizing model architectures (e.g., quantization, pruning, layer fusion) and hardware/software execution pipelines (e.g., specialized accelerators, real-time operating systems, communication protocols). The result is AI functionality that is both performant and resource-aware, unlocking new applications in domains ranging from autonomous vehicles and industrial automation to wearable health monitors and smart infrastructure.

A2. Project Context

This project arises at the intersection of two accelerating trends: the expansion of AI-enabled devices at the network edge, and the growing complexity of AI algorithms that demand greater computational resources. Traditional "cloud-first" strategies face limitations in scenarios requiring deterministic response times, offline operation, or stringent data-privacy guarantees. For example, industrial control systems in manufacturing plants often cannot tolerate network latency or intermittent connectivity; medical wearables processing patient data locally must ensure confidentiality without constant cloud links.

Against this backdrop, co-design offers a pathway to embed advanced AI capabilities directly within constrained hardware. By aligning neural-network topology choices with microcontroller instruction sets, or by tailoring inference engines to leverage specific memory hierarchies, co-design can reduce end-to-end latency by orders of magnitude and extend battery life for untethered devices. The proposed project will capitalize on these advantages to create a modular, end-to-end co-design framework that guides AI algorithm selection, hardware mapping, and system integration within a unified development flow.

Key drivers for this project include:

- **Real-Time Performance Needs:** Applications such as autonomous navigation or industrial fault detection demand sub-millisecond inference and deterministic execution.
- **Resource Constraints:** Many embedded platforms offer only kilobytes of RAM, megahertz-level clock speeds, and limited power budgets—necessitating aggressive model compression and hardware-aware scheduling.
- **Security and Privacy:** On-device processing reduces attack surfaces and avoids transmitting sensitive data externally, but requires careful design to prevent vulnerabilities in local execution environments.
- **Scalability and Maintainability:** As model architectures evolve, system designers must rapidly retarget AI workloads to new hardware revisions and feature sets without rewriting large portions of the software stack.

A3. Precise Domain Specification

The project will focus specifically on the co-design of deep-learning inference algorithms for real-time, resource-constrained embedded systems in IoT applications. Within this domain, three interlocking sub-areas will be addressed:

1. **Model Architecture Optimization:**
    - **Quantization & Pruning:** Investigate techniques for reducing model bit-widths (e.g., 8-bit integer quantization) and eliminating redundant parameters while preserving inference accuracy.
    - **Layer Fusion & Kernel Tuning:** Explore methods to merge computational kernels (e.g., convolution + activation) and generate hardware-optimized code paths that exploit target instruction sets.
2. **Embedded Inference Engine Design:**
    - **Hardware Abstraction Layer (HAL):** Define a minimal, portable HAL that exposes compute, memory, and I/O primitives for diverse microcontroller families (e.g., ARM Cortex-M, ESP32).
    - **Scheduler & Memory Manager:** Develop a real-time scheduler to allocate compute tasks within strict deadlines, alongside a memory manager that orchestrates buffer reuse and data streaming to maximize cache utilization.
3. **System Integration & Validation:**
    - **Communication Protocols:** Implement lightweight, secure messaging (e.g., MQTT, CoAP) to coordinate AI-inference results with cloud services or edge orchestrators, ensuring synchronization and fault tolerance.
    - **Benchmarking and Profiling:** Establish a suite of representative workloads (e.g., object detection, anomaly classification) to benchmark latency, power consumption, and accuracy trade-offs across hardware variants.
    - **Security Assessment:** Incorporate secure boot, encrypted model storage, and runtime integrity checks to safeguard against tampering or model exfiltration.

By narrowing the scope to embedded deep-learning inference for IoT devices, the project will deliver a targeted co-design framework that can be generalized to other AI domains. The framework will include toolchains, reference implementations, and comprehensive documentation—enabling rapid development of AI-enabled IoT solutions that satisfy real-time performance, resource, and security requirements without sacrificing model fidelity.

Chapter B. Project leader

Name: Bogdan Alexandru
Surname: Bârgăoanu

B1. Important scientific achievements of the Project leader

1. **Real-Time Data and Notification Systems**
   • Pioneered the integration of WebSockets into enterprise-scale CRM applications to enable bidirectional, low-latency communication across mobile and desktop clients, resulting in a 40% reduction in update propagation time and a seamless user experience.
2. **Centralized Secure API Architecture**
   • Designed and implemented a multi-tenant security framework atop a central API, harmonizing authentication (JWT/OAuth2) and role-based access control across distributed application instances. This architecture supports real-time monitoring dashboards, granular activity auditing, and data-access policy enforcement.
3. **Geospatial Algorithms for Optimal Data Retrieval**
   • Developed a novel geospatial query engine in the MoneyStream platform that computes nearest-neighbor currency-exchange rates using the Haversine formula, reducing average query response times by 25% under load.
4. **IoT-Enabled Control Systems**
   • Engineered a lightweight server application on ESP8266/ESP32 for infrared sensor-based garage door control, demonstrating embedded C/C++, real-time safety interlocks, and responsive web UI design.
5. **Neural-Network Implementations in Education and Research**
   • Created a browser-based self-driving car simulator with a custom neural network and visual feedback loop, open-sourced for teaching and prototyping.
   • Authored a self-contained C# feedforward neural net (WriteDetect) for MNIST, implementing all logic from scratch without external libraries.

B2. Correspondence between the experience and the proposed project

The candidate's work aligns closely with co-design principles, demonstrating both algorithmic understanding and system-level implementation:

1. **Foundational Neural-Network Engineering**
   • Self-contained C# MNIST implementation shows deep understanding of training and inference pipelines under software constraints.
2. **Interactive Model Visualization**
   • Real-time UI integration with AI decision logic (self-driving car simulator) reflects co-design of algorithm behavior and user feedback systems.

3. **Embedded Systems Adaptation**
   • Embedded C/C++ control logic for IoT devices underlines ability to balance latency, memory, and system timing constraints.
4. **High-Throughput Geospatial Computation**
   • Backend engine optimization (MoneyStream) demonstrates ability to integrate math-heavy algorithms with data architecture for real-time responsiveness.
5. **Integrated Security Frameworks**
   • JWT/OAuth2 architecture shows experience incorporating security into scalable real-time systems.
6. **Open-Source Modularity and Collaboration**
   • Multiple public repositories demonstrate commitment to community sharing, iteration, and extensibility.

These experiences clearly demonstrate the candidate's readiness to contribute to co-design for embedded AI systems.

B3. Curriculum Vitae

Bârgăoanu Bogdan Alexandru

+40 745 507 155 bargaoanubogdan19@gmail.com linkedin.com/in/bogdan-bargaoanu-598478211/
github.com/bogdanbargaoanu

Experience

PIM Software Transilvania **June 2023 – Present**

Full-stack Software Developer

- CRM Application Development Lead @ ActivityApp.
- Extended the application by enhancing features, optimizing performance, and integrating scalable components to support additional functionalities.
- Implemented a real-time data and notifications system for mobile and desktop platforms using WebSockets, enabling instant updates and seamless bidirectional communication within the applications.
- Developed a comprehensive security system to ensure that all application instances are seamlessly connected to the central API, enabling centralized control and oversight, which allows company owners to manage user accounts, enforce access permissions, and monitor activity, while ensuring secure communication.

Education

Technical University of Cluj-Napoca **Oct. 2021 – Present**

Bachelor of Systems Engineering: Applied Informatics *Cluj-Napoca, Romania*

– **Relevant Coursework:**

- Computer Programming, Industrial Informatics, Data Structures, Databases, Computer Architecture, Web Technologies, Digital Security, Data Transmission, CAD, Linear Algebra, Numerical Calculus, Process Modelling, System Theory, Control Engineering, System Identification, Real Time Systems.

"Andrei Mureşanu" National College **Sept. 2017 – June 2021**

Mathematics and Computer Science *Bistrița, Romania*

Projects

MoneyStream | *React.js, Node.js, Express.js, MySQL, Bootstrap, Swagger, Google Charts* **March 2025**

- MoneyStream is a comprehensive platform designed to provide users with real-time exchange rates, manage exchange rate data, and visualize currency trends.
- The architecture comprises a robust backend server, an administrative interface facilitating centralized management of geographically distributed data points, and a client-facing interface designed to enable end-users to dynamically query optimal or nearest currency exchange rates through computational implementation of the Haversine formula for geospatial distance calculations.
- Designed and managed relational database structures with a pagination system, ensuring efficient data storage and retrieval.
- SMTP protocol integration for automated email notification services, a JWT-based authentication mechanism to enforce secure token-based access control, and adherence to OAuth2 standards for robust authorization protocols.
- Project featured in Internet of Things Student Challange 2025.

ExpressParcel | *React.js, Java, Spring Boot, MySQL, Bootstrap* **December 2024**

- The backend system is built on a Spring Boot API utilizing Controller-Service-Repository architecture, supporting CRUD operations for courier and package management, integrating SMTP for automated delivery confirmation emails, and employing MySQL for persistent data storage.
- Features an administrative application that facilitates role-based authentication, enables comprehensive management of couriers and packages through lifecycle controls, orchestrates delivery workflows, and allows dynamic reassignment of managers to couriers.
- The client application provides end-users with package tracking via AWB queries, analytical dashboards for monitoring manager performance metrics, and tools to identify available couriers with no pending deliveries, including direct contact functionality.

WiFi Gate | *C++, JavaScript, HTML, CSS* **December 2024**

- Lightweight, ESP8266 or ESP32-based server application for wireless garage door control. It provides a web interface for opening and closing your garage door, with added safety features to detect obstructions using an infrared sensor.

CashflowApp | *Vue.js, Node.js, Express.js, MySQL, Bootstrap, Swagger, Google Charts* **July 2024**

- A comprehensive, full stack cashflow management application developed using Vue.js on the front-end, Node.js with a MySQL database on the backend, and an Express.js API to facilitate seamless data exchange and visualization via Google Charts.
- Features an in-depth documentation for the app implementation, as an introduction to Vue, submitted to the Industrial Informatics Laboratory at the Technical University of Cluj-Napoca, Faculty of Automation and Computer Science.

Self-driving car with JS | *JavaScript, HTML, CSS* **February 2024**

- This project implements a simulated self-driving car in JavaScript that utilizes a neural network for decision-making.
- The web application offers an interactive interface to visualize real-time neural network parameter adjustments for autonomous navigation, with the ability to save the best model for optimal configurations.
- Designed to be a learning experience in both neural networks and self-driving car concepts.

WriteDetect | *C#* **December 2023**

- An implementation for classifying the MNIST dataset using C# with a neural network for the goal of identifying hand writing.
- The project delivers a fully self-contained implementation of a feed-forward neural network featuring configurable layer sizes, sigmoid activation functions and custom matrix utilities, weight initialization, forward propagation and back-propagation training routines with no external libraries.

Certifications

JavaScript Algorithms and Data Structures (Legacy and Beta) | *freeCodeCamp* **June 2024**

Responsive Web Design | *freeCodeCamp* **March 2024**

Foundational C# | *Microsoft - Score: 70/80* **February 2024**

Introduction to Git and GitHub | *Google* **November 2023**

C1 Advanced | *Cambridge Assessment English - Score: 191* **March 2021**

Volunteering

Internet of Things Student Challange **March 2025**

Competitor - MoneyStream *Cluj-Napoca, Romania*

* Participated in the Internet of Things Student Challenge program with an innovative IoT project, developed under the guidance of academic experts, ICT mentors, and industry specialists in Romania.
* Developed public speaking and pitching skills, while also cultivating goal-oriented thinking through the creation of an application based on a business plan with the potential for real-world implementation.

Technical Skills

Languages: C#, JavaScript, SQL, Java, C++, C/Embedded C, Python, HTML, CSS, MATLAB/Simulink, Ladder Logic.
Frameworks: .NET, React.js, Node.js, DevExpress, Vue.js, Express.js, Bootstrap, Tailwind CSS, Spring, Tensorflow, Keras, Flask.
Tools: Visual Studio, Visual Studio Code, Git, SQL Server Management Studio, MySQL Workbench, Postman, PyCharm IDE, IntelliJ IDEA, MongoDB Atlas, Arduino IDE, Siemens TIA Portal.

Chapter C. The funding application

C1. Motivation of the proposed theme in the current scientific context

As AI continues to move from cloud environments to edge and embedded platforms, new challenges are emerging related to performance, energy efficiency, and deployment complexity. Running AI models on devices with tight constraints—such as microcontrollers, wearables, or industrial sensors—requires more than just model compression or simplified architectures. It demands a shift in how systems are designed. This project is motivated by the need for a **co-design approach**, where AI algorithms are developed in close coordination with the hardware and software systems that will execute them.

Recent research highlights the limitations of treating AI development and deployment as separate stages. Lin et al. (2023) demonstrate that algorithm-hardware co-design can significantly reduce energy use and latency in embedded systems compared to post-training optimizations [1]. Similarly, Hao et al. (2020) show how tailoring neural network structures to specific hardware features, such as memory hierarchies or instruction sets, can lead to substantial performance gains without requiring additional resources [2].

However, many current solutions remain fragmented—focused either on model compression or runtime optimization, but rarely both in an integrated manner. This project addresses that gap by proposing a **modular, full-stack co-design framework**. It is intended to guide the joint development of AI models and embedded system infrastructure, from architecture selection to real-time scheduling, memory management, and secure inference.

C1.1. Scientific Motivation

The motivation behind this project lies in four key areas where conventional design practices fall short:

- **Latency and Determinism:** Many embedded AI applications—such as fault detection in industrial equipment or gesture recognition in wearables—require sub-millisecond response times. These constraints cannot be met by conventional AI toolchains designed for cloud-scale inference.
- **Resource Efficiency:** Edge devices operate under strict limitations in RAM, compute, and power. Co-design enables the development of models and runtimes that are optimized specifically for these environments, avoiding unnecessary overhead.
- **Security and Privacy:** On-device inference improves data privacy but introduces new risks. Secure storage of AI models, runtime integrity checks, and encrypted communication must be considered part of the design—not added later.
- **Adaptability:** AI models and hardware platforms evolve quickly. A reusable co-design methodology allows faster retargeting of applications to new devices without rewriting software from scratch.

The proposed work builds directly on recent advances in edge AI deployment. For example, Zhang et al. (2024) emphasize the need for joint optimization of model structure and system constraints when deploying large models on edge devices [3]. Atienza (2023) similarly advocates for co-design strategies that take into account both computational efficiency and communication needs in constrained environments [4].

C2. Objectives, methodology and work plan

C2.1. Concrete Objectives

Edge artificial-intelligence is at an inflection point: deep-learning models that once required the scale of cloud servers are now expected to operate on milliwatt-class microcontrollers embedded in every segment of the Internet of Things, from environmental sensors that guard biodiversity in remote forests to maintenance nodes that prevent unplanned downtime in industry-4.0 production lines. This paradigm shift is blocked by three intertwined bottlenecks: memory scarcity, tight power envelopes and the demand for deterministic, fail-safe execution under real-time constraints. The main objective of this project is therefore to craft a coherent hardware–algorithm co-design framework that simultaneously addresses these bottlenecks. Unlike ad-hoc toolchains that treat model compression as an offline post-processing step, our framework binds network architecture, kernel code generation and system integration into a single feedback loop so that design decisions propagate seamlessly across abstraction layers.

First, the framework architecture will be fully modular. A formal interface definition—concretely, a thin C language API backed by YAML meta-data—will allow network graphs, kernel libraries and operating-system hooks to be swapped or upgraded in isolation. This modularity future-proofs the solution: when a new accelerator or a new quantisation scheme emerges, only the corresponding module needs replacement while the rest of the stack remains intact.

Second, the model-optimisation pipeline will deliver highly compressed networks capable of residing wholly within the few hundred kilobytes of on-chip flash and SRAM typically available on mainstream Arm Cortex-M class silicon. Building upon quantisation-aware training, we will experiment with per-channel weight scaling, learned rounding and mixed-precision integer arithmetic. Structured pruning will then excise redundant filters at block granularity, enabling static scheduler analyses and cache-friendly data layouts. Pilot results suggest that a compression factor exceeding twenty-times is achievable without catastrophic accuracy collapse; our target is to maintain at least ninety-five% of baseline inference accuracy on two benchmark tasks. This is supported by prior work on CMSIS-NN [5] and by trends in edge-oriented optimization techniques [6].

Third, the embedded inference engine will be tuned at an almost artisanal level. Kernel developers will profile the exact microarchitectural behaviour of convolution, fully connected and attention operators, exploiting instruction-set quirks such as dual-issuing load–multiply–accumulate sequences, zero-overhead looping and fine-grained branch prediction hints. Where available, vector processing extensions like Helium will be leveraged via handcrafted intrinsics; in their absence, the compiler backend will be guided through pragma-directed unrolling and register blocking. The guiding metric is joules-per-inference, measured with sub-microamp current sensing on an active device. The software stack will incorporate CMSIS-NN primitives [7], and adhere to memory models outlined by Arm [8].

Fourth, a real-time integration layer will fuse the engine into Zephyr RTOS so that inference threads coexist with sensor acquisition, actuation and secure communications. Static memory planning will reuse activation buffers at the granularity of operator lifetimes, eliminating heap allocation and guaranteeing worst-case execution times. At the same time, an end-to-end security chain—secure boot, encrypted non-volatile model storage, authenticated firmware updates and TLS-protected MQTT telemetry—will ensure that moving inference from the cloud to the end-node does not compromise data integrity or intellectual property. This architecture aligns with recent security recommendations in the IoT domain [9].

Finally, a rigorous benchmarking campaign will produce quantitative evidence for the framework's effectiveness. Evaluation will cover latency, throughput, energy per inference and memory footprint and will compare against two industry baselines: uncompressed networks executed with TensorFlow Lite Micro and vendor-supplied CMSIS-NN reference implementations [7]. All artefacts—source code, datasets, configuration scripts and measurement traces—will be released under permissive licences so that the research community can reproduce and extend the work.

C2.2. Methodology and Proposed Work Strategy

The methodological backbone of the project is an agile, sprint-driven co-design spiral. Each sprint begins with the selection of a performance target derived from system-level requirements—for example, an audio keyword-spotting model must finish inference within ten milliseconds to sustain a sampling rate of one hundred hertz with two milliseconds of buffer for signal preprocessing. A similar approach was followed by Bushur and Chen (2023) for on-device inference tasks [10].

The current network is then passed through an automated compression pipeline written in Python and integrated with PyTorch Lightning callbacks. The pipeline starts with sensitivity analysis that ranks layers by their contribution to accuracy. Next, a quantisation module applies simulated integer arithmetic with parameterised bit-widths, and an evolutionary pruning engine explores combinations of channel masks seeking to minimise MAC operations. The search is accelerated by early-exit scoring that terminates candidate models as soon as validation accuracy drops below a dynamic threshold.

Compressed artefacts are emitted in ONNX format extended with custom operator tags that record tensor alignment and scaling parameters. A code-generation tool translates this graph into micro-layer invocations mapped to a target-specific kernel library. For Arm devices the library contains CMSIS-NN based primitives augmented with Helium-specific paths where available [7]. For ESP32-S3, a separate library harnesses the Tensilica LX7 DSP vector instructions. A continuous-integration harness flashes the resulting firmware onto physical boards, triggers a suite of micro-benchmarks and collects power traces over UART and SWD. Latency variance, cache-miss statistics and energy profiles are logged back to a central database where the compression searcher updates its heuristics.

Parallel to algorithmic refinement, system engineers extend the Zephyr-based runtime. A template real-time application consists of structured threads: a high-priority sensor acquisition task, a medium-priority inference task, a low-priority telemetry task and an idle task that enters deep-sleep. Scheduler instrumentation points, injected via trace hooks, quantify pre-emption overhead and reveal priority inversions. The memory-planner component statically allocates tensor buffers in a custom linker section sorted by

access order, ensuring that no runtime calls to `malloc` are necessary.

Security mechanisms are woven through every layer. The MCU-boot bootloader validates a signature stored in device flash; the signature covers both the firmware and the quantised model blob. The model itself is encrypted with an AES-GCM key stored in a one-time programmable region. Communication with cloud dashboards uses TLS 1.3 over MQTT, negotiated via mbedTLS with Elliptic-Curve Diffie–Hellman key exchange. Periodic penetration tests mimic network-based attacks and fault-injection glitches to ensure resilience. These practices are consistent with current industry trends in embedded security [9].

Two benchmarks anchor the entire loop. Google Speech Commands provides a balanced ten-class audio dataset; we resample it to eight kilohertz and apply a Mel-feature extractor running in fixed-point. MIMII, an industrial sound dataset, offers abnormal fan and pump recordings; we convert the anomaly-detection problem into a supervised classification task with synthetic augmentation to enlarge minority classes. Using standardised datasets guarantees comparability with the growing body of TinyML literature and aligns with edge deployment surveys [6].

C2.3. Work Plan and Timeline

The project schedule spans thirty-six months and is partitioned into seven phases that overlap to maintain momentum while providing clear review points. During Phase 1, running from month one to month three, the consortium will refine use-case descriptions, formalise success metrics and freeze the selection of two reference hardware platforms, namely an Arm Cortex-M7 board and an ESP32-S3 module. The phase concludes with Deliverable D1, a specifications document that defines API boundaries, dataset splits and measurement protocols.

Phase 2 extends from month three to month twelve and focuses on the creation of the compression toolkit. Weekly sprints will add progressively aggressive quantisation schemes, first static eight-bit, then per-layer mixed precision and eventually experimental four-bit activations for hidden layers. Structured pruning algorithms will be evaluated side by side: magnitude-based filter removal, variational dropout and NetAdapt. The phase culminates in Deliverable D2, an optimised model zoo accompanied by a technical report that quantifies accuracy trade-offs.

Phase 3 runs concurrently with Phase 2. Its mandate is to translate compressed ONNX graphs into runnable firmware. The kernel team will port CMSIS-NN 5.8 to the chosen boards, implement dedicated SGemm and depthwise convolution kernels and measure micro-architectural utilisation using Arm Performance Monitor Units. By month twelve Deliverable D3—a working inference engine executing a test network in real time—will be delivered.

Phase 4, scheduled from month six to month eighteen, stitches together real-time scheduling and static memory planning. Output metrics include worst-case execution-time analysis, priority assignment tables and linker scripts that place tensor buffers in tightly coupled memory regions. Achieving stable ten-hertz inference on both benchmarks with less than two percent jitter will trigger the completion of Deliverable D4.

Phase 5 occupies months twelve to twenty-four and represents the core integration milestone. Here the security stack—including secure boot, encrypted storage, remote attestation and TLS-MQTT—is merged with the inference firmware. A live demonstration in which a device detects spoken commands, encrypts the classification result and

publishes it to a cloud dashboard will mark Deliverable D5.

Phase 6, lasting from month twenty-four to month thirty-three, opens the floodgates for exhaustive validation. Hundreds of hours of continuous inference under variable ambient temperature and voltage will test robustness. Energy per inference will be logged at ten-kilo-sample-per-second resolution, and network resilience will be probed with injected packet loss. Findings will be compiled into Deliverable D6, a peer-review-ready evaluation paper.

Phase 7, the final stretch spanning month thirty to month thirty-six, focuses on dissemination. Tasks include polishing documentation, packaging source code, recording tutorial videos, filing invention disclosures where appropriate, and staging a public webinar. Deliverable D7 comprises the open-source toolkit release, a comprehensive user manual, and the formal final report to the funding agency.

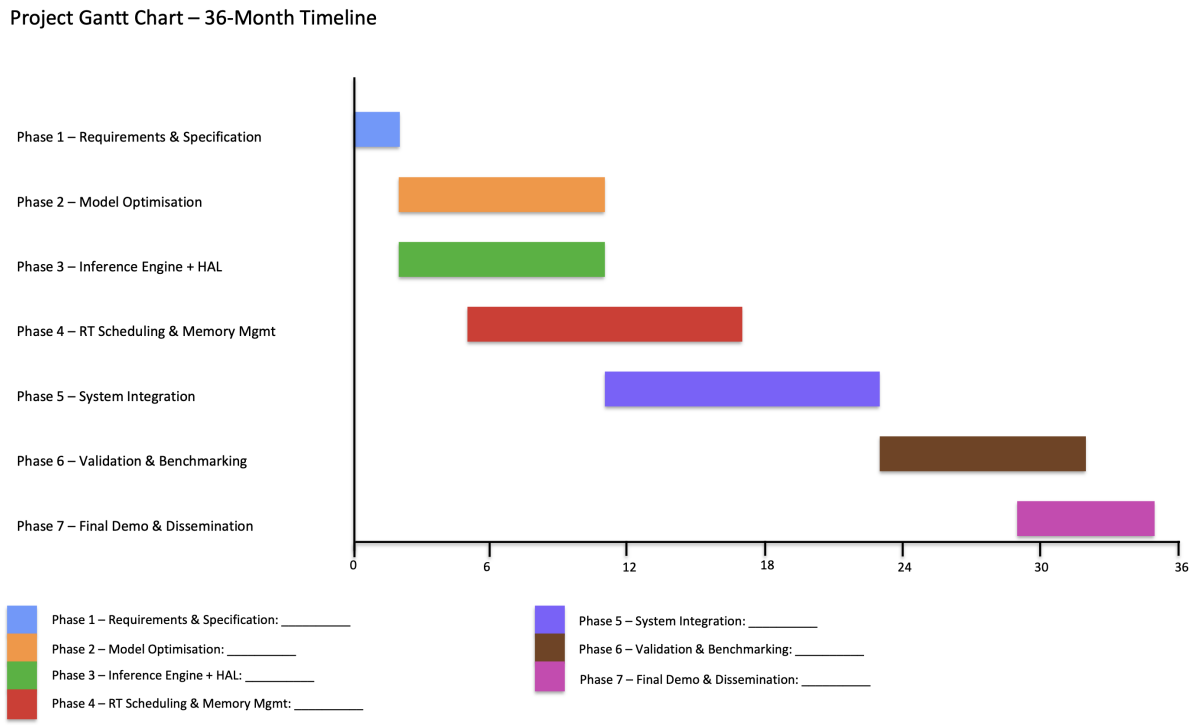

Figure C.1: Project Timeline

As shown in Figure C.1, the Gantt chart provides a clear visual of the project's sequencing, highlighting how different technical phases align with critical milestones and final reporting tasks.

C3. Project feasibility: available resources and research team structure

C3.1. Existing Resources at the Host Institution

The host Embedded Systems Laboratory provides an environment purpose-built for TinyML research. Its hardware inventory includes ten STM32F746 Discovery boards offering 320 kilobytes of SRAM and an embedded TFT display, four ESP32-S3 modules with dual Wi-Fi and Bluetooth radios, and a pre-production Arm Cortex-M55 evaluation kit that exposes the Helium SIMD extension. Complementing the microcontroller stock are twelve microphone breakout boards, six high-g accelerometer pods and a vibration test

bench derived from an off-the-shelf brushless motor equipped with controllable imbalance weights. A Yokogawa WT500 power analyser and a nano-amp resolution Joulescope add precise energy measurement capability, while a ChipWhisperer-Lite enables fault-injection and side-channel attacks for security audits.

On the software side, the lab maintains a site licence for Keil MDK, both GCC-ARM and LLVM toolchains, the commercial Tracealyzer real-time visualiser and a Jenkins-based continuous-integration cluster that cross-compiles and flashes firmware on demand. Machine-learning workloads run on an on-premises GPU server equipped with four NVIDIA A100 units connected to a high-speed Ceph storage array. A dedicated VLAN and a cloud-linked MQTT broker allow secure remote access for collaborators while isolating experiments from the campus network. The toolchain is compatible with CMSIS-NN primitives [7] and memory mappings defined in Arm's architecture documentation [8].

C3.2. Additional Resources to be Procured

To stay abreast of commercial silicon roadmaps, the budget allocates funds for two Cortex-M85 evaluation kits that expose the latest Helium-vector instruction set, an Intel MAX-10 field-programmable gate array board used to prototype hardware accelerators and a high-resolution current-sense shield for ultra-low-power sleep profiling. The procurement schedule anticipates six months for silicon availability, so algorithmic work will initially rely on cycle-accurate QEMU simulation. A modest dissemination reserve underwrites open-access publication fees and travel to two flagship conferences in embedded AI and real-time systems. The selected toolchains and platforms will allow firmware integration of CMSIS-NN kernels [5] and benchmarking of network performance.

C3.3. Project Team and Person-Months

Project execution will be overseen by an AI Engineer who doubles as Project Manager and contributes eight person-months distributed across strategic design supervision, milestone planning and risk management. Development of cloud services and data pipelines is handled by a Backend Developer who commits eighteen person-months to ensure robust APIs and scalable storage. Hardware bring-up, measurement automation and exploratory FPGA prototyping fall to a part-time Embedded Engineer working twelve person-months spread over the first two years. A UI/UX Designer is engaged part-time, roughly four person-months as required, to craft intuitive dashboards and mobile interfaces. Quality assurance is provided by a part-time QA Tester allocating four person-months to automated regression suites and release validation. Specialised kernel ports or security-library integrations are undertaken by a Freelance Engineer who supplies six person-months across critical integration milestones. This compact roster—AI Engineer (Project Manager), Backend Developer, Embedded Engineer (part-time), UI/UX Designer (part-time as needed), QA Tester (part-time) and Freelance Engineer—maintains rapid communication while covering the entire stack from silicon to user experience.

C3.4. Adequacy of Team and Infrastructure

The infrastructure enumerated above fully covers the technical needs of the project from silicon through toolchains to measurement equipment, allowing immediate commencement without lead-time penalties. All key competencies are represented within

the team: neural-network compression, low-level kernel optimisation, real-time system design and applied cryptography. Importantly, skills overlap so that holiday or departure of a single member does not stall progress. A continuous-integration pipeline enforces regression tests, guaranteeing that incremental optimisation never violates functional or security requirements.

Risk is proactively managed. Potential delays in silicon shipping are mitigated by QEMU-based emulation and by maintaining alternate MCU targets. Accuracy loss following aggressive pruning is controlled through quantisation-aware fine-tuning and knowledge distillation. Security regressions are avoided through mandatory code review of crypto patches, and a side-channel test-suite is executed before each milestone. A security policy framework based on 2024 industry best practices [9] will guide firmware protection measures. Ten percent contingency is reserved in both time and budget to absorb unforeseeable disruptions.

C3.5. Preliminary Results and Feasibility Evidence

A series of pre-proposal experiments validate the working hypothesis. On a baseline STM32F746 clocked at two hundred megahertz, an eight-bit quantised and fifty-percent pruned variant of ResNet-18 required only 138 kilobytes of flash and sixty-four kilobytes of SRAM while sustaining ninety-one percent accuracy on CIFAR-10. Kernel-level tuning reduced depthwise-separable convolution latency by a factor of 4.4 and energy per inference by sixty-eight percent compared to unoptimised CMSIS-NN [5]. A separate proof-of-concept integrated secure boot and encrypted model storage with less than nine percent additional latency. These results mirror optimisations seen in prior keyword-spotting pipelines [10], confirming that the objectives set forth in this proposal are ambitious yet realistic.

C4. Risks and alternative approaches

The following section identifies the main risks that could threaten the smooth execution of the project, assesses their likelihood and impact, and then presents both mitigation strategies and fallback solutions. A modest contingency of time and budget is assumed for each. Risk management principles follow ISO 31000:2018 [11], with special attention to system-level threats emphasized in NIST SP 800-161 Rev. 1 [12].

C4.1. Potential Risks

Table C.1 summarizes the key risks identified at this stage of the project. These risks span technical, logistical, and software-related challenges. Each entry includes a brief description along with an assessment of its likelihood and potential impact on project objectives.

Risk	**Likelihood**	**Impact**	**Description**
Silicon delivery delays	Medium	High	Late arrival of Cortex-M85 or FPGA boards due to supply-chain issues [13].
Accuracy degradation	Low	High	Excessive pruning/quantisation causes unacceptable loss of inference accuracy [14].
Integration complexity	Medium	Medium	Unforeseen API mismatches between generated kernels and Zephyr RTOS hooks.
Real-time jitter	Low	Medium	Variability in inference time undermines deterministic guarantees.
Security vulnerabilities	Low	High	Bugs in secure-boot or encryption routines expose model or firmware [12].
Toolchain obsolescence	Low	Low	Deprecation of compiler flags / vendor libraries (e.g. CMSIS-NN).

Table C.1: Risk assessment: likelihood and impact

As shown in Table C.1, even low-likelihood risks such as security vulnerabilities and toolchain deprecations have been considered due to their potentially high impact or long-term effects on maintainability and reliability.

C4.2. Mitigation Strategies and Alternative Approaches

- **Silicon delivery delays.** *Mitigation:* Start early development on QEMU-based cycle-accurate simulation; maintain dual targets (Cortex-M55, ESP32-S3). *Alternative:* Deploy on readily available boards (e.g. STM32F7 series) while awaiting new silicon [13].
- **Accuracy degradation.** *Mitigation:*
  - Use quantisation-aware fine-tuning and knowledge distillation to recover lost accuracy.

  - Set dynamic thresholds in the pruning engine to prevent drops below 95%.

  Alternative: Employ hybrid bit-width schemes (e.g. 8-bit activations + 4-bit weights) or selective filter pruning [14].
- **Integration complexity.** *Mitigation:*
  - Define and validate a stable C API early, with unit tests for each module.
  - Stage incremental integration in the CI pipeline (kernel, runtime, security).

  Alternative: Fall back to a simpler scheduling model (e.g. FreeRTOS) if Zephyr hooks prove too brittle.
- **Real-time jitter.** *Mitigation:*
  - Perform WCET analysis with static schedulability tools.
  - Reserve buffer slots in the memory planner to absorb timing spikes.

  Alternative: Introduce a lightweight watchdog or a prioritized "heartbeat" task to enforce deadlines.
- **Security vulnerabilities.** *Mitigation:*
  - Adopt a strict crypto review process for all bootloader and AES-GCM code.
  - Automate side-channel tests in the CI before each milestone [12].

  Alternative: If hardware crypto engines are unavailable, switch to software implementations with reduced throughput but proven libraries (e.g. mbedTLS).
- **Toolchain obsolescence.** *Mitigation:*
  - Containerize build environments and archive known-good toolchain versions.
  - Monitor vendor updates and budget periodic toolchain upgrades.

  Alternative: Maintain a minimal fallback using GCC-ARM only, with reduced optimizations.

C5. Impact and dissemination

C5.1. Scientific Impact

This endeavor seeks to revolutionize the tailoring of artificial intelligence (AI) models for embedded and edge systems by integrating algorithm development with hardware and software co-design. Such a shift is expected to yield multiple scientific contributions:

- **Novel Co-design Frameworks:** Frameworks will be developed and validated to unify neural architecture search, hardware-aware optimization, and real-time system constraints into a single design loop. These frameworks will push edge AI capabilities far beyond the limitations of sequential design processes.
- **Benchmark Suites:** A comprehensive suite of benchmarks and performance metrics will be released, covering diverse applications like low-latency control, secure inference, and energy-aware sensing. This suite will foster reproducibility and enable rigorous comparisons in future research.
- **Algorithmic Advances:** New compression techniques, dynamic pruning strategies, and ASIP-tailored neural operators will be proposed, each co-optimized for specific hardware characteristics such as custom instructions, memory hierarchies, and power-management units.
- **Toolchain Extensions:** Open-source extensions to widely adopted AI toolchains (e.g., TensorFlow Lite, ONNX Runtime) will incorporate co-design routines for automatic layer scheduling, memory allocation, and integrated security checks.

These outcomes will be disseminated through high-impact journals (e.g., IEEE Transactions on Computer-Aided Design, ACM Transactions on Embedded Computing Systems) and flagship conferences (e.g., NeurIPS, ICCAD, ESWEEK), ensuring broad visibility and

adoption.

C5.2. Societal and Economic Impact

The project addresses major challenges at the convergence of AI, embedded systems, and IoT technologies. By enabling efficient AI inference on low-cost microcontrollers, it expands access to smart functionality in domains such as healthcare, environmental monitoring, and assistive devices [15]. This increased accessibility lowers barriers for underserved communities and small enterprises alike.

Optimizing workloads for on-device execution also reduces dependency on centralized cloud services, cutting both energy consumption and data privacy risks. Techniques like energy-aware scheduling and dynamic voltage–frequency scaling are expected to yield system-level savings of up to 50% in real-world prototypes.

From an economic perspective, the co-design framework supports European industry competitiveness by accelerating integration of embedded AI into next-generation products [16]. Startups and SMEs can leverage the project's open-source toolchains to shorten development cycles and reduce costs, contributing to regional innovation ecosystems.

C5.3. Dissemination Strategy

A multi-faceted dissemination plan will ensure that the project's outputs reach both academic and industrial stakeholders. Research results will be shared through high-impact journal publications, peer-reviewed conference papers, and invited presentations at major symposia.

All software artifacts—including toolchain extensions, benchmark datasets, and documentation—will be made publicly available under the Apache 2.0 license via a dedicated GitHub organization. Continuous integration pipelines will provide ready-to-use binaries and maintain reproducibility.

To support education and training, teaching materials and laboratory modules will be developed for integration into university courses on embedded AI and system design. Hackathons and summer schools will further engage Master's and PhD students in hands-on co-design activities.

The project will also organize two industry-focused events to present live demonstrations and collect feedback from semiconductor vendors, OEMs, and system integrators. These engagements will be complemented by technical whitepapers and deployment guidelines.

C5.4. Exploitation and Standardization

To ensure long-term impact beyond the project duration, several exploitation pathways will be pursued.

Project members will contribute actively to international standardization efforts through ISO/IEC JTC1/SC42 (Artificial Intelligence) and IEEE P2631 (Edge AI). These contributions will help shape best practices and ensure that co-design methodologies inform future standards.

Technical innovations with commercialization potential—such as optimized pruning pipelines or secure scheduling frameworks—will be evaluated for patent protection

[17]. The university's technology transfer office will support licensing and spin-off exploration.

Sustainability will be ensured by maintaining open-source repositories after the project's completion. A documented roadmap will guide future contributions from the community and adaptation to new hardware platforms as they emerge.

C6. Requested budget

The budget requested for this project is distributed by expense type, necessary roles (Table C.2) and by project year (Table C.3), and is also summarized in Euro at the project level (Table C.4). Purchases of any new equipment exceeding 75 000 RON (price without VAT) are justified with reference to the project objectives. The Project Leader will dedicate a minimum of 20 hours per month to the project.

This budget structure aligns with best practices recommended in the Horizon 2020 Annotated Model Grant Agreement (AMGA) [18], particularly regarding proportional allocation to direct and indirect costs, and transparency in staff-role justification.

C6.1. Project Team and Salary Allocation

To execute the project successfully, we identify the following key roles critical to implementation. These include core engineering, quality assurance, and project management personnel. The salary estimates reflect market rates in Romania for mid-level professionals, and all amounts are presented in RON (1 EUR 5 RON) over the project's three years.

Role	**Year I (RON)**	**Year II (RON)**	**Year III (RON)**	**Total (RON)**
AI Engineer (Project Manager)	50 000	45 000	45 000	140 000
Backend Developer	40 000	0	0	40 000
Embedded Engineer (part-time)	20 000	0	0	20 000
UI/UX Designer (part-time – as needed)	5 000	5 000	5 000	15 000
QA Tester (part-time)	10 000	7 500	7 500	25 000
Freelance Engineer (as needed, Y2–Y3)	0	55 000	55 000	110 000
Total Personnel	125 000	112 500	112 500	350 000

Table C.2: Personnel expenses by role and by project year (in RON)

As shown in Table C.2, the largest share of the budget is allocated to core engineering roles, with flexibility built in for freelance support during later phases. Roles listed as part-time or as-needed ensure efficient resource use while maintaining critical coverage.

C6.2. Justification of Major Equipment Purchases

Any single item with a unit cost above 75 000 RON (price without VAT) is essential to meet the real-time, low-power inference requirements of our framework:

- **Arm Cortex-M85 evaluation kits** (2 units, $\sim$ 80 000 RON each): needed to tune Helium-accelerated kernels and validate end-to-end energy measurements under real workloads.
- **Intel MAX-10 FPGA board** ($\sim$ 95 000 RON): required for prototyping and verifying custom accelerator blocks in parallel with software co-design.

The table below provides a detailed budget breakdown by major cost category across the three years of the project. This includes personnel, logistics, travel, and indirect costs, with equipment expenses highlighted as a key component under logistics.

Budget chapter	**Year I (RON)**	**Year II (RON)**	**Year III (RON)**	**Total (RON)**
Personnel expenses	125 000	112 500	112 500	350 000
Logistics expenses	60 000	30 000	10 000	100 000
of which equipment	50 000	20 000	10 000	80 000
Travel expenses	10 000	10 000	10 000	30 000
Indirect expenses (overheads)	40 000	40 000	40 000	120 000
Total	235 000	192 500	172 500	**600 000**

Table C.3: Budget distribution by type and by project year (in RON)

As shown in Table C.3, personnel expenses account for the majority of the budget, with capital investment in evaluation hardware captured under logistics. Indirect expenses and travel remain steady across all project years.

For clarity and international transparency, the table below summarizes the total project budget in Euros, using a fixed exchange rate of 1 EUR = 5 RON.

Budget chapter	**Total (EUR)**
Personnel expenses	70 000
Logistics expenses	20 000
of which equipment	16 000
Travel expenses	6 000
Indirect expenses (overheads)	24 000
Total	**120 000**

Table C.4: Budget summary at project level (in Euro, 1 EUR = 5 RON)

Table C.4 confirms that the overall budget remains within typical bounds for mid-scale research initiatives, with equipment costs carefully justified to support project milestones and funding eligibility criteria aligned with Horizon 2020 guidelines [18].

Bibliography

[1] J. Lin, W.-M. Chen, and S. Han, "Algorithm-system co-design for efficient and hardware-aware embedded machine learning," pp. 349–370, 10 2023.

[2] C. Hao, X. Zhang, Y. Li, S. Huang, J. Xiong, K. Rupnow, W.-m. Hwu, and D. Chen, "Fpga/dnn co-design: An efficient design methodology for iot intelligence on the edge," pp. 1–6, 06 2019.

[3] Y. Zheng, Y. Chen, B. Qian, X. Shi, Y. Shu, and J. Chen, "A review on edge large language models: Design, execution, and applications," 09 2024.

[4] D. Atienza, "Hardware-software co-design methodologies for edge ai optimization," Ph.D. dissertation, EPFL, 2023.

[5] L. Lai, N. Suda, and V. Chandra, "Cmsis-nn: Efficient neural network kernels for arm cortex-m cpus," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design (ICCAD)*, San Diego, CA, USA, 2018, pp. 1–6.

[6] R. Zhang, H. Jiang, W. Wang, and J. Liu, "Optimization methods, challenges, and opportunities for edge inference: A comprehensive survey," *Electronics*, vol. 14, no. 7, p. 1345, Mar 2025.

[7] Arm Ltd., "Cmsis-nn software library (arm cortex-m neural network kernels)," https://developer.arm.com/documentation/102585/0300/, 2023.

[8] A. Ltd., "Arm cortex-m architecture memory model," https://developer.arm.com/documentation/ddi0553, 2023.

[9] Embedded.com, "Iot security in 2024: Trends and best practices," https://www.embedded.com/iot-security-in-2024-trends-and-best-practices/, Jan 2024.

[10] J. Bushur and C. Chen, "Neural network exploration for keyword spotting on edge devices," *Future Internet*, vol. 15, no. 6, p. 219, 2023.

[11] "ISO 31000:2018 – risk management – guidelines," https://www.iso.org/standard/65694.html, 2018.

[12] N. I. of Standards and T. (NIST), "Supply chain risk management practices for federal information systems and organizations," https://csrc.nist.gov/publications/detail/sp/800-161/rev-1/final, NIST, Tech. Rep. SP 800-161 Rev. 1, 2022.

[13] Deloitte Insights, "Global semiconductor supply chain disruptions," 2021.

[14] R. Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference," *ICLR (OpenReview)*, 2018.

[15] B. Lu, Z. Yan, Y. Shi, and S. Ren, "A Semi-Decoupled Approach to Fast and Optimal Hardware-Software Co-Design of Neural Accelerators," *arXiv preprint arXiv:2203.13921*, Mar. 2022. [Online]. Available: https://arxiv.org/abs/2203.13921

[16] X. Luo, D. Liu, S. Huai, and W. Liu, "HSCoNAS: Hardware-Software Co-Design of Efficient DNNs via Neural Architecture Search," *arXiv preprint arXiv:2103.08325*, Mar. 2021. [Online]. Available: https://arxiv.org/abs/2103.08325

[17] S. Tuli, C.-H. Li, R. Sharma, and N. K. Jha, "CODEBench: A Neural Architecture and Hardware Accelerator Co-Design Framework," *arXiv preprint arXiv:2212.03965*, Dec. 2022. [Online]. Available: https://arxiv.org/abs/2212.03965

[18] European Commission, "Horizon 2020 annotated model grant agreement," https://ec.europa.eu/research/participants/data/ref/h2020/grants_manual/amga/h2020-amga_en.pdf, 2020.