RISCVML: Teaching RISC-V Embedded ML with Rust — From ESP32-C3 to ESP32-P4

Scottie von Bruchhausen
RISCVML — riscvml.org — scottie@riscvml.org

Abstract

The rapid deployment of RISC-V in embedded systems, IoT, and edge AI has outpaced developer education: most existing tutorials target C/C++ and cover only basic microcontroller tasks, leaving a gap for engineers who need to build machine-learning-capable systems with modern toolchains. RISCVML addresses this gap with a structured, Rust-first curriculum spanning 172 chapters across seven modules, progressing from entry-level hardware to on-device ML inference. The curriculum uses commercially available Espressif RISC-V SoCs as its teaching platform: the ESP32-C3 (single-core, BLE 5.0, ~€3) and ESP32-C6 (Wi-Fi 6, Thread/Matter, ~€4) introduce embedded Rust fundamentals — GPIO, sensors, power management, and wireless protocols. The ESP32-P4 (dual-core 400 MHz, AI extensions, 128-bit vector ISA, ~€25 dev board) anchors an advanced module covering its ISP camera pipeline, hardware-accelerated 2D rendering, H.264 video encoding, DMA orchestration, and vector-accelerated ML inference. These subsystems converge in a real-world capstone: an on-device bird-detection pipeline that captures frames via MIPI-CSI, runs quantized object detection through esp-dl, drives pan/tilt servos for tracking, and records H.264 video — all orchestrated in async Rust with ESP-IDF drivers integrated via FFI where hardware support requires it. By pairing Rust's memory-safety guarantees with production-ready toolchains (esp-hal, esp-idf-hal) on affordable, widely available hardware, RISCVML lowers the barrier for engineers, students, and hobbyists entering the RISC-V ecosystem — directly supporting Europe's push for open-standard, sovereign silicon literacy.

1. Summary of Contribution

RISCVML (riscvml.com) is an educational platform that provides a structured, hands-on curriculum for learning Rust-based embedded systems development on RISC-V microcontrollers. The platform targets the full range of Espressif's RISC-V SoCs: the entry-level ESP32-C3, the Wi-Fi 6-capable ESP32-C6, the TTGO T-Beam (a LoRa-capable development board), and the high-performance ESP32-P4 — a dual-core 400 MHz RISC-V processor with AI instruction extensions, 128-bit vector operations, MIPI-CSI/DSI camera and display interfaces, and H.264 hardware encoding.

The curriculum comprises 172 chapters organized into seven progressive modules. Modules 1–5 cover Rust fundamentals, GPIO/sensor/peripheral control (I2C, SPI, PCA9685), power management and solar harvesting, LoRa on T-Beam, and multi-device interconnection via ESP-NOW/MQTT. Module 7 bridges firmware to desktop applications using Tauri.

Module 6 is dedicated to the ESP32-P4, with Rust exercises for each advanced subsystem: type-safe ISP camera pipeline configuration (MIPI-CSI capture, white balance, demosaicing), hardware-accelerated 2D rendering via PPA with LVGL bindings, end-to-end H.264 encoding at 1080p@30fps with zero-copy buffer management, async DMA orchestration via embassy with compile-time borrow checking, and vectorized ML inference using 128-bit SIMD inline assembly wrappers around the RISC-V vector ISA.

Capstone: Bird Detection Pipeline

Module 6 culminates in a real-world capstone: a complete bird-detection system on ESP32-P4 that exercises every hardware subsystem in a single, deployable application. The pipeline is Rust-first while leveraging ESP-IDF drivers (esp-video, esp-detection/esp-dl) via FFI, demonstrating pragmatic interoperability:

Phase 1 — Camera → ISP → Display: MIPI-CSI capture at 30–60 FPS using the Camera Controller Driver, ISP pipeline (white balance, auto-exposure, demosaicing) via type-safe Rust abstractions, DMA-driven frame transfer to MIPI-DSI display, PPA for scaling/rotation. Embassy async runtime orchestrates zero-copy buffer flow enforced by ownership semantics.

Phase 2 — On-Device Inference: Espressif's esp-detection/esp-dl runs quantized object detection on the P4's AI-accelerated cores. PPA hardware-downscales frames to model input resolution, avoiding CPU-bound resizing. Bounding boxes and confidence scores overlay the preview via PPA alpha blending. The 128-bit vector extensions accelerate tensor operations within esp-dl. Detection results (species, confidence, bounding box, timestamp) are logged to a SQLite3 database (riscvml_detect.db). An RGB LED is driven to a species-specific color via a bird_led_colors lookup table in the same database, providing instant visual identification of detected birds.

Phase 3 — Tracking, Servos & Recording: Detection coordinates drive PCA9685 pan/tilt servos (reusing Module 2 abstractions). H.264 hardware encoder records events at 1080p@30fps with zero-copy frame ingestion. A companion ESP32-C6 provides Wi-Fi 6 via ESP-Hosted/SDIO for MQTT alerts and RTSP streaming — demonstrating the P4's intended companion-chip architecture.

The bird detection capstone instantiates a reusable Detect → Visualize → React pattern: Sensor Input → ML Classification → SQLite Lookup (class → RGB color) → RGB LED → Reaction. The curriculum applies this same architecture to object/obstacle detection, plant health assessment, and sound classification — swapping sensors, models, and color maps while sharing the ESP32-P4 infrastructure, teaching transferable ML-on-edge skills. Each chapter functions as a self-contained 30–60 minute learning

unit with complete source code, wiring diagrams, and expected output.

2. Importance for the Community

The RISC-V ecosystem faces an asymmetric growth challenge: hardware availability has scaled rapidly — over 20 billion cores projected by 2025 — but developer education has not kept pace. Industry surveys consistently identify the software ecosystem as the primary barrier to RISC-V adoption. RISCVML addresses this with three differentiators:

Rust-first approach: While most RISC-V tutorials rely on C/C++, RISCVML uses Rust throughout. Compile-time memory safety eliminates entire bug classes common in embedded C, and zero-cost abstractions are particularly valuable on resource-constrained RISC-V cores. The Rust embedded ecosystem (esp-hal, embassy) has matured sufficiently to make this production-viable.

Commercially available hardware: The ESP32-C3 (~€3), ESP32-C6 (~€4), and ESP32-P4 dev boards (~€25) ensure accessibility across the performance spectrum — learners progress from low-cost modules to the P4's 400 MHz dual-core with AI extensions within a unified Espressif RISC-V ecosystem.

End-to-end pipeline: From bare-metal firmware to Tauri desktop applications, and from minimal IoT nodes (C3) to multimedia edge computing (P4 with MIPI displays and cameras), RISCVML spans the full embedded spectrum.

3. Ecosystem Contribution

RISCVML contributes at multiple levels: as a talent pipeline providing English-language, self-paced training comparable to China's SOPIC program; as implicit ecosystem testing through real-world exercises against Espressif's esp-rs toolchains with upstream bug reports; and as a freemium distribution model (€2/chapter, €30 bundle, free introductory content) that sustains development while remaining accessible. The platform also offers partnership opportunities for ecosystem companies seeking developer onboarding content.

4. Target Audience

The primary audiences are: embedded engineers evaluating ARM-to-RISC-V transition with Rust; university educators seeking structured RISC-V coursework on affordable hardware; IoT/LoRa hobbyists wanting guided RISC-V learning paths; and ecosystem companies interested in educational partnerships or curriculum licensing.

The poster presentation will include QR codes linking to the live platform, sample chapter previews, and demonstrations of the firmware-to-desktop pipeline running on ESP32-C6 and ESP32-P4 hardware with MIPI display output.

References

[1] RISC-V International, "RISC-V Ecosystem Overview," riscv.org, 2025.

[2] Espressif Systems, "ESP32-C3, ESP32-C6, and ESP32-P4 Technical Reference Manuals," espressif.com, 2024-2025.

[3] The Embedded Rust Book, docs.rust-embedded.org, 2025.

[4] Espressif, "esp-dl: Deep Learning Library for ESP32-S3/P4," github.com/espressif/esp-dl, 2025.

[5] Espressif, "esp-video: Video Framework for ESP32-P4," github.com/espressif/esp-video-components, 2025.

[6] EE Times, "RISC-V Exceeding Expectations in AI, China Deployment," October 2025.