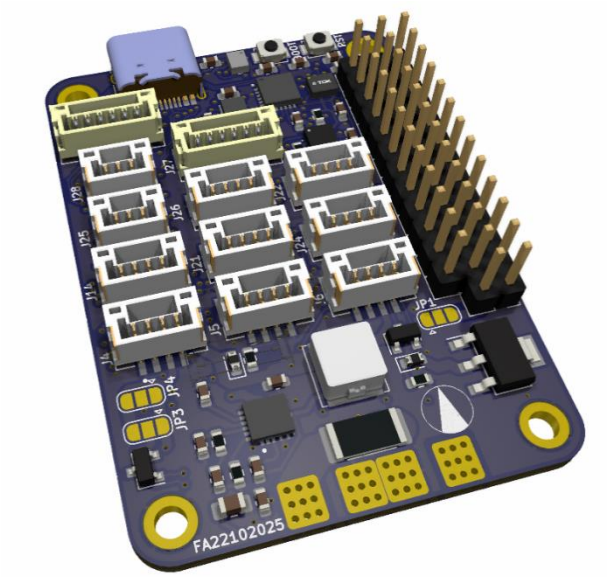
Ochin STM32H743 User Manual



1. Introduction

The öchìn_STM32H743 is not only a companion board for the öchìn_CM4, it's a fully featured, high-performance microcontroller platform ideal for robotics, IoT, and automation. Powered by the STM32H743, one of the most capable MCUs in the STM32 family, it offers exceptional processing performance, advanced peripherals, and deterministic real-time control capabilities.

Thanks to its ready-to-use software ecosystem, developers can start immediately:

- Complete STM32CubeIDE projects (bare-metal and FreeRTOS)
- micro-ROS examples for ROS2 integration
- Arduino-compatible libraries for rapid prototyping
- Reference implementations for sensor acquisition, PWM control, video OSD management, and communication with the öchìn_CM4

This makes the platform ideal for both rapid experimentation and industrial-grade embedded solutions.

The board interfaces with the öchìn_CM4 and CM5 through a direct mezzanine connector, providing power, dual UART links, I²C, and analog video signals without any external wiring.

This direct connection ensures a clean, compact, and robust architecture, allowing the öchìn_CM4/CM5 to act as the high-level computation unit while the öchìn_STM32H743 manages timing-critical hardware tasks.

Moreover, the mezzanine system is intentionally designed to be expandable. Developers can design and insert custom intermediate boards between the öchìn_CM4 and öchìn_STM32H743. These expansion modules can communicate using one of the available UART channels or the I²C bus, providing a straightforward way to integrate:

- New sensors
- Actuation modules
- Communication interfaces
- Application-specific hardware

This modular architecture enables the creation of powerful, scalable embedded systems while maintaining mechanical compatibility and electrical simplicity.

Key Features:

- Perfect mechanical and electrical integration with the öchìn_CM4 via a direct mezzanine connector, no external cabling required.
- Same form factor as the öchìn_CM4, designed to stack seamlessly.
- STM32CubeIDE support, with both bare-metal and FreeRTOS project templates available.
- micro-ROS support for full ROS2 interoperability.
- Arduino IDE compatibility for rapid prototyping and simplified development.
- Full support for open-source INAV and Ardupilot firmware, matching the MatekH7 architecture for effortless autopilot integration.
- Suited for systems combining INAV/Ardupilot with OpenHD, enabling advanced autonomous capabilities alongside low-latency HD video links.

2. Technical Specifications

2.1 Power

- Power input voltage: 7.5–28V
- Power output: voltage follows the Power Input (up to 90A continuous and 200A peak).
- Current and voltage monitoring: integrated before the 5V regulator
- 5V 7A regulator: main load supply
- 3.3V linear regulator: powers MCU and onboard peripherals

2.2 Onboard Sensors and Leds

- IMUs: mpu-6000 and ICM-42688: 2x inertial measurement units for full redundancy
- DPS310: Environmental sensor for pressure and temperature
- HMC5883: Magnetometer for the orientation
- Led1 and Led2: +5V and +3,3V power monitor
- Led3 and Led4: User Leds (connected to PE3 and PE4)

2.3 Interfaces

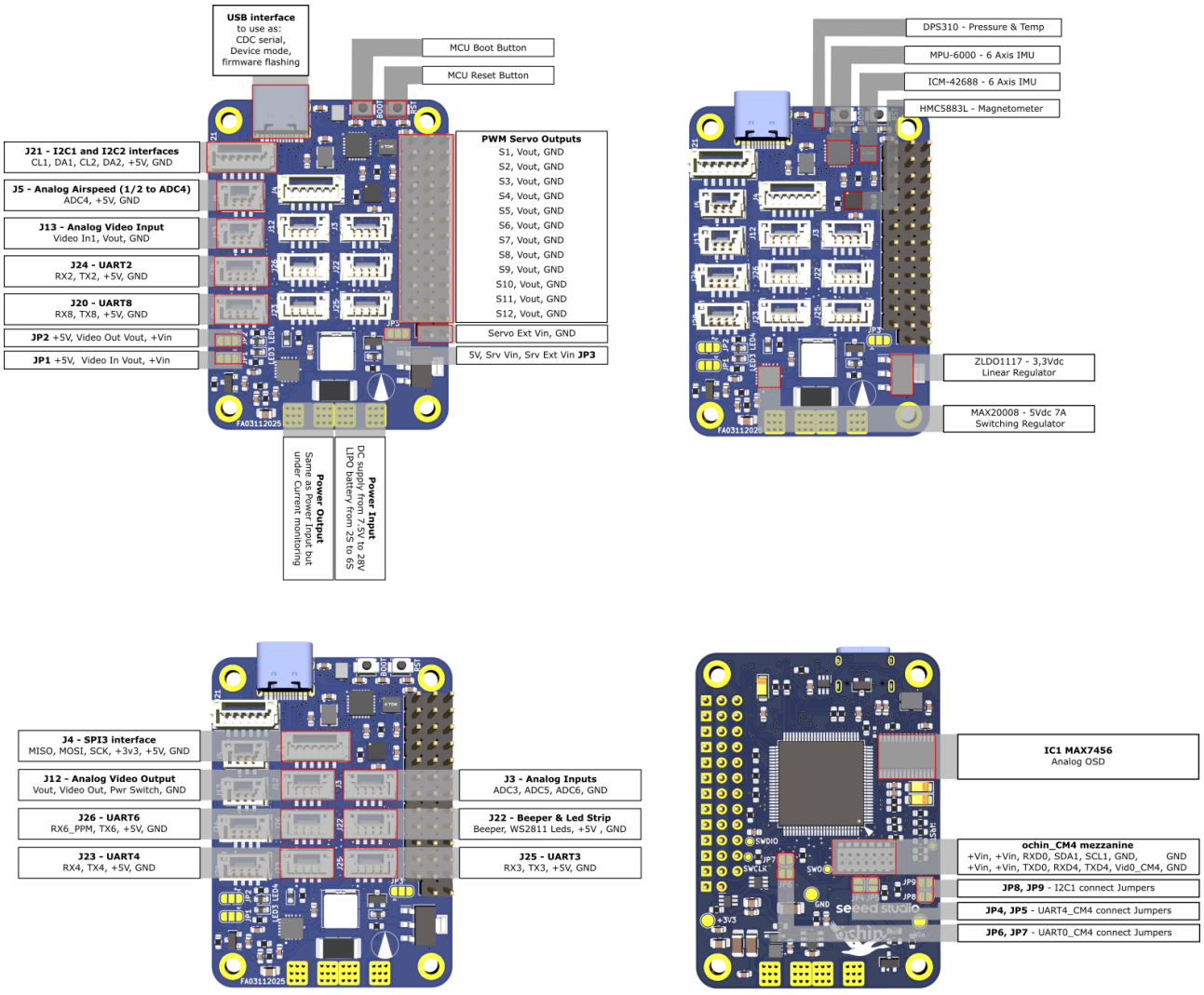
- PWM outputs: 12 channels for servo control
- UART: 7 ports

- I2C: 2 ports
- ADC: 4 analog inputs
- SPI: 1 interface
- Beeper output: 1
- Video:
    - 2 analog video inputs (one via mezzanine, one via top connector)
    - 1 analog video output with onboard analog OSD
    - MCU-controlled switch to select input for OSD overlay

2.4 Mechanical and Electrical

- Same form factor as öchìn_CM4
- Mezzanine connector provides 2xUART, I2C, video, and power interfaces to öchìn_CM4

3. Architecture



3.1 Overview

The Ochin STM32H743 is designed as a low-level controller bridging the CM4 and real-world hardware. The STM32H743 executes deterministic, time-critical tasks such as sensor data acquisition and data fusion, PWM control, and OSD overlay, while the CM4 handles high-level algorithms and system coordination.

Block diagram overview:

STM32H743 Controls:
- PWM outputs → Servos
- UART/I2C → Sensors & peripherals
- ADC → Analog sensors
- SPI → External devices
- OSD → Video overlay
- Beeper → Audio notifications

STM32H743 Sensors:
- Vin 1/10 → ADC1 Main Voltage Monitoring
- Max44284 → ADC2 Main Current Monitoring
- ADC 3,4,5 → analog input
- IMUs → Attitude measurements
- Magnetometer → heading
- Pressure and Temperature → Pressure / Altitude and Temperature

3.2 Power Flow

- The main power input (VIN) is monitored for voltage and current.
  VIN is protected against reverse polarity by a P-channel MOSFET (U3), ensuring safe operation in case of incorrect power connection.
- A 5 V / 7 A switching regulator supplies the main loads, including PWM outputs, sensors, and the onboard OSD.
- A 3.3 V linear regulator supplies the MCU and low-power peripherals.
  The MCU can be powered either from VIN or from USB-C VBUS during development and configuration. When powered from USB-C, only the MCU and 3.3 V-powered sensors are energized; all high-power loads and 5 V domains remain disabled. This allows safe firmware development and system configuration without powering external peripherals. An ad-hoc power selection and isolation circuit automatically manages the selection between VIN and USB-C VBUS. This circuit guarantees that the two power sources are electrically isolated from each other, preventing back-feeding and any mutual interference. As a result, the board can be safely powered from USB during development without risk of damaging the host PC, while normal operation from VIN remains unaffected when the main supply is present.

3.3 Sensor Management

- Dual IMUs provide redundancy; data can be fused in firmware.
- Environmental sensors (pressure, temperature, magnetometer) are accessible via I2C.
- MCU provides standardized interfaces for easy integration.

3.4 Video Handling

The öchìn_STM32H743 features a flexible analog video subsystem designed to integrate both traditional analog FPV cameras and digitally generated video from the öchìn_CM4.

- Dual Analog Video Inputs:
  The board provides two independent analog video input channels:
  - Top-mounted analog video connector:
    Typically used for connecting a standard analog FPV camera.
  - Mezzanine analog video input from the öchìn_CM4:
    This is the analog output generated by the Raspberry Pi Compute Module 4. The CM4 can render digital graphics, UI elements, telemetry panels, or low-resolution video streams and output them as an analog signal.

This allows the system to generate custom visuals digitally on the CM4 and transmit them as analog video—ideal for FPV setups or legacy analog systems.

- MCU-controlled Video Source Selection:
  The STM32H743 microcontroller controls an electronic switch that selects which of the two analog sources is routed to the onboard OSD (On-Screen Display). This selection can be changed dynamically by firmware.

- Integrated Analog OSD:
  The onboard OSD module can overlay text, symbols, and graphical elements onto the selected analog video feed.
  This enables flight or system telemetry to be displayed regardless of the active video input.

- Flexible Output for External VTX:
  The OSD-processed video is routed to the analog video output, which can be connected directly to an external video transmitter (VTX).
  This means the transmitted analog signal can originate from:

  - a traditional analog camera,
  - any digital UI rendered by the CM4 and converted to analog.

This architecture makes the öchìn_STM32H743 ideal for hybrid systems that combine digital processing on the CM4 with analog transmission, giving developers the freedom to merge classic FPV workflows with modern, programmable visual overlays.

3.5 Communication Interfaces

- UART ports for serial peripherals and debugging
  The board provides 7 UART interfaces in total.
  - Top connectors (GH 4-pin): UART2, UART3, UART4, UART6, UART8
  - Mezzanine connector (bottom, for connection to öchìn_CM4 or custom boards): UART1, UART7

  All UARTs operate at 0–3.3V logic levels, but the RX pins are 5V tolerant except for UART7, which is only 3.3V tolerant.

- I2C buses for sensors and external modules
  The MCU provides two I2C buses, which are used to connect onboard sensors:
  - I2C1: connected to the HMC5883 magnetometer
  - I2C2: connected to the DPS310 barometer

  Both I2C buses are also accessible via connector J21 for adding additional I2C devices. Each bus includes 4.7 kΩ pull-up resistors on the lines.

- SPI interface

  The board provides SPI3 accessible via connector J4, which can be used to connect high-speed external devices.

3.6 ADC channels

The board provides several analog inputs accessible via connectors:

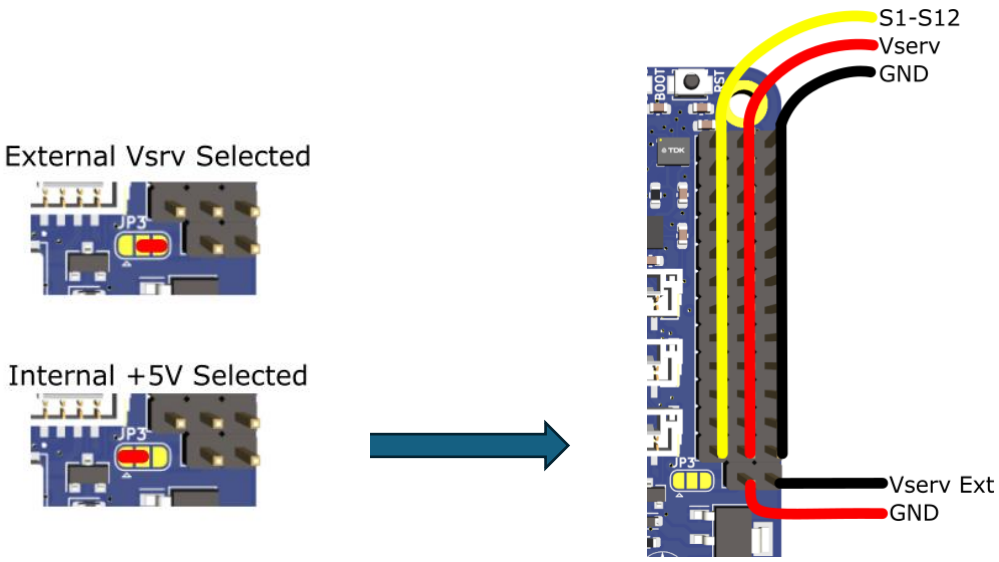
- J3 connector: three analog inputs ADC3, ADC5, ADC6, connected to pins PC5, PA4, PA7 respectively. All are 3.3 V tolerant, so it is recommended to properly protect any connected devices to avoid exceeding the maximum voltage.

- J5 connector: ADC4, connected to pin PC4, designed to receive an analog sensor signal between 0 V and +6.6 V. The signal is scaled by 1/2 using a voltage divider.
- Internal measurement: ADC1 and ADC2, connected to pins PC0 and PC1, are used internally to measure the main input voltage (Vin) and current consumption

3.7 PWM outputs

The board provides 12 connectors (S1–S12) for PWM outputs. Each connector is a 3-pin strip: VServo, Signal, GND.
The VServo voltage can be selected via JP1:

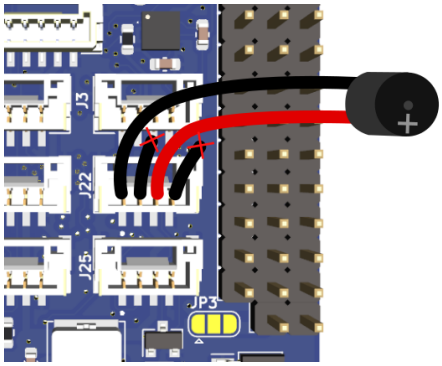
- Connected to the onboard +5 VDC supply
- Or externally injected via connector J30



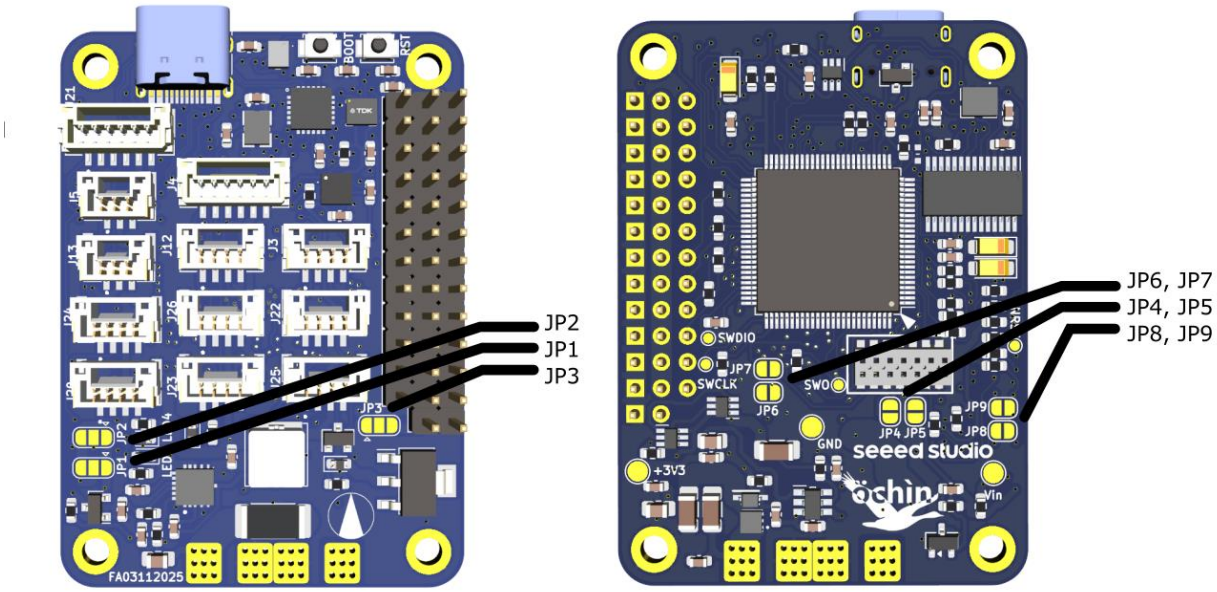
The signal pins are general-purpose GPIOs that can also function as PWM outputs connected to timers. Timers allow automatic generation of square waves with configurable frequency and duty cycle. Various waveform types can be generated, including the standard 50 Hz square wave with a high-level duration between 1 ms and 2 ms, suitable for controlling typical RC servos.

3.8 Beeper output for alerts or notifications

The beeper can be controlled via connector J22. The control signal is a GPIO connected to pin PA15, which can also be configured as PWM output Channel 1 of TIM2.
This signal drives a MOSFET in common-source configuration. The beeper should be connected with its positive to +5 V and its negative to pin 1 of J22. When the control signal applies a voltage to the MOSFET gate, the MOSFET closes the circuit to ground, allowing current to flow through the beeper. By varying the PWM signal on the gate, the beeper can produce audible tones or alerts.



3.9 SMD Jumpers



These SMD jumpers have been included to make the board more flexible, allowing users to decide how to connect it to the öchìn_CM4 or to custom expansion boards.

JP1 – Camera Power Output Selection (J25)
JP1 selects the output voltage available on connector J25, intended for powering an external analog video source.

- Pins 1–2 shorted: Output voltage = VIN.
- Pins 2–3 shorted: Output voltage = +5V.

JP2 – VTX Power Output Selection (J26)
JP2 selects the output voltage available on connector J26, intended for powering a video transmitter (VTX).

- Pins 1–2 shorted: Output voltage = VIN.
- Pins 2–3 shorted: Output voltage = +5V.

JP3 – Servo Power Source Selection
JP3 is used to select the power source for the 12 servo connectors (S1–S12).

- Pins 1–2 shorted: Servos are powered from the onboard 5V rail.
- Pins 2–3 shorted: Power must be externally injected through connector J42.

JP4, JP5 – Routing CM4 UART4 to STM32 UART7
These jumpers should be soldered if you want to connect the CM4's UART4 to the STM32's UART7, with TX and RX crossed to allow proper communication.

- JP4: Connects RXD4_CM4 → TX7
- JP5: Connects TXD4_CM4 → RX7

If UART4_CM4 is used by a third-party board, these jumpers should remain open.

JP6, JP7 – Routing CM4 UART0 to STM32 UART1
These jumpers should be soldered if you want to connect the CM4's UART0 to the STM32's UART1, again with TX and RX crossed.

- JP6: Connects RXD0_CM4 → TX1
- JP7: Connects TXD0_CM4 → RX1

If UART0_CM4 is used by another board, these jumpers should remain open.

JP8, JP9 - Routing CM4 I2C1 to STM32 I2C1
These jumpers normally remain open. They should only be closed in the following cases:

- To connect a custom third-party board to the STM32's I2C1, leaving it disconnected from the CM4's I2C1.
- To configure STM32 I2C1 in slave mode.

Closing these jumpers inappropriately can result in two I2C masters being connected to the same bus, which must be avoided to prevent communication conflicts or hardware damage.

4. Firmware and Software

A significant amount of work has been dedicated to providing a complete, ready-to-use firmware ecosystem for the öchìn_STM32H743. The goal is to ensure that developers can take full advantage of every sensor and peripheral on the board immediately, without needing to design their software architecture from scratch.

4.1 Comprehensive Example Projects

The firmware package includes a wide collection of example applications demonstrating how to interact with all onboard sensors and hardware features, including:

- Dual IMU acquisition and redundancy handling
- Barometric pressure and temperature sensing
- Magnetometer interfacing
- ADC sampling
- PWM servo control
- OSD management and video source switching
- UART, I²C, and SPI communication examples

Each example is designed to be practical, modular, and easy to extend, serving both as a learning tool and as a starting point for real-world applications.

4.2 Multi-Environment Development Support

To support different workflows, the examples are implemented across several development environments:

- **STM32CubeIDE**
  - Full bare-metal and FreeRTOS projects
  - Based on ST's HAL and LL drivers
  - Ideal for high-performance embedded development
- **micro-ROS (ROS2)**
  - Ready-to-use micro-ROS nodes for sensor publishing and actuator control
  - Designed to integrate seamlessly with a ROS2 system running on the öchìn_CM4
- **Arduino IDE**
  - Simplified sketches for quick prototyping
  - Ideal for education, experimentation, or rapid feature testing

4.3 Based on Both Open-Source and ST Proprietary Libraries

The firmware examples combine:

- Open-source libraries (sensor drivers, middleware, ROS2 client code, utility modules)
- ST's official HAL/LL libraries for robust low-level hardware access
- Custom middleware developed specifically for the öchìn_STM32H743

This hybrid approach ensures maximum flexibility while maintaining reliability and performance.

4.4 Rapid Development and Easy Integration

All provided examples are structured to be easily reused and adapted. Developers can quickly:

- Use the examples as templates
- Integrate them into larger applications
- Combine them into a fully functional autopilot or robotic controller
- Build on them while interacting with the öchìn_CM4 or öchìn_CM5 through the mezzanine connector

Thanks to this ecosystem, it is possible to go from unboxing the board to running custom code in just minutes.

4.5 Recommended Development Workflow

To help developers get the most out of the öchìn_STM32H743 from day one, the firmware package follows a clear and efficient development workflow. This workflow minimizes setup time, reduces complexity, and ensures a smooth progression from basic testing to full application development.

Step 1 — Explore the Provided Examples

Begin by reviewing and running the example projects included in the firmware package. These examples demonstrate how to interface with all onboard sensors and peripherals and serve as practical references for your own application code.

Recommended first steps:

Run the IMU and barometer reading examples

Test PWM outputs using a servo

Verify UART/I²C communication

Try video source switching and OSD overlay

Test micro-ROS node communication with the öchìn_CM4

Each example is self-contained and includes comments to explain the logic and configuration.

Step 2 — Choose Your Development Environment

Select the environment that best fits your project:

STM32CubeIDE for high-performance embedded applications

FreeRTOS when deterministic task scheduling is needed

micro-ROS for ROS2 robotics and distributed systems

Arduino IDE for fast prototyping or educational use

Step 3 — Use Example Code as Your Project Template

All examples are structured to be used as starting points:

Copy the example folder

Rename it as your project

Replace or expand the application logic

Reuse initialization code and drivers

This approach ensures a solid, validated foundation and drastically reduces development time.

Step 4 — Integrate with the öchìn_CM4 / CM5

Once your firmware runs on the STM32:

Connect the board directly via the mezzanine connector

Exchange data using UART or I²C

Optionally use micro-ROS for ROS2-based communication

Leverage the CM4/CM5 as a high-level computation unit

This architecture allows you to split computation between real-time tasks (STM32) and advanced algorithms (CM4/CM5).

Step 5 — Expand the System if Needed

Using the mezzanine interface, you may insert custom expansion boards between the öchìn_STM32H743 and the öchìn_CM4/CM5:

New sensors

RF modules

Custom electronics

Additional controllers

Communication can be performed via the available UART or I²C bus, maintaining a clean and modular design.

Step 6 — Build, Test, Iterate

The ecosystem is designed for rapid iteration:

Modify code

Upload via USB DFU mode and STM32 CubeProgrammer

Monitor outputs (UART console, ROS2 topics, analog video OSD)

Debug and refine

Thanks to the provided examples and unified driver architecture, iteration cycles remain fast and predictable.

5.Compatibility with INAV and Ardupilot

In the öchìn_STM32H743, the MCU pins for sensors, connectors, and peripherals are organized to meet the expectations of open-source firmware such as INAV and Ardupilot. This ensures full firmware compatibility without requiring custom builds or modifications.

Because the board follows the same logical architecture that these firmware expect, existing INAV and Ardupilot binaries targeting boards that mounts the same MCU, such as MatekH7, can be used directly on the öchìn_STM32H743.

Developers can consult the official schematic or the CubeMX-generated PDF to understand the MCU-to-peripheral mappings. Additionally, reviewing the INAV source code illustrates how the firmware expects signals and peripherals to be organized.

This approach guarantees a seamless drop-in experience while maintaining full compliance with open-source firmware expectations.

6. Safety and Notes

To ensure safe operation and to prevent damage to the device or connected equipment, please observe the following guidelines:

Verify the input voltage range before powering the board. Supplying voltages outside the recommended range may permanently damage the power circuitry.

Respect the current limits of the onboard regulators. Exceeding these limits may cause overheating, instability, or failure of the power system.

Ensure proper grounding and shielding when connecting analog video signals to prevent noise, interference, or potential damage to the video subsystem.

Avoid short circuits and accidental contact with conductive materials when the board is powered.

Operate the device in an appropriate environment, avoiding excessive humidity, dust, vibration, or temperature extremes unless specifically designed for such conditions.

Do not modify the hardware unless you fully understand the electrical and mechanical implications of such changes.

7. License and Legal Disclaimer

The firmware, schematics, and documentation are provided under the terms of the license included with this product.
Users are required to read and fully understand the license before using, modifying, or distributing any part of the design or software.

While the hardware and software have been developed with great care, the author assumes no responsibility for any damages, malfunctions, data loss, or legal issues resulting from the use, misuse, or improper handling of this device.

The board is intended for research, development, prototyping, and educational use.
It must not be used in safety-critical systems or applications where failure could lead to injury, death, or significant property or environmental damage.

By using this product, the user agrees that all risks associated with installation, configuration, and operation are solely their responsibility.