NAVR BOTEC

Man meets machine

Vayu Flight Control Stack

Technical Report Summary

Prepared by:

Ashutosh Vishwakarma

2026

v0.0.1

Abstract

This document provides a concise summary of the Vayu Flight Control Stack technical report. Vayu is an end-to-end flight control system built on a hardware-agnostic infrastructure comprising NavHAL (Hardware Abstraction Layer) and VAIOS (Real-Time Operating System). The stack emphasizes modularity, deterministic execution, and hardware independence, offering a streamlined alternative to complex open-source platforms like PX4 and ArduPilot. This summary covers the system architecture, core components, performance characteristics, and development roadmap.

1 Introduction and Motivation

The Vayu flight control stack was developed to address limitations in existing open-source drone platforms, which often introduce unnecessary complexity, hardware lock-in, and limited customization capabilities. The project originated from practical challenges encountered during undergraduate research at IIT Jammu, particularly when attempting to implement secure encrypted telemetry—a feature that required external hardware and complex workarounds in traditional stacks.

Vayu adopts a first-principles approach with three primary objectives:

- **Modularity:** Clear separation between hardware abstraction, execution management, and control logic.
- **Determinism:** Predictable real-time execution for stable flight control.
- **Hardware Independence:** Portability across microcontroller platforms through structured abstraction.

The stack is vertically integrated across three layers: **NavHAL** (Hardware Abstraction), **VAIOS** (Execution Environment), and **Vayu** (Flight Control Logic).

2 System Architecture Overview

2.1 Layered Design

The Vayu system follows a four-layer hierarchical architecture:

1. **Hardware Layer:** Physical components including STM32F401RE MCU, BMX160 IMU, BMP180 barometer, and motor drivers.
2. **NavHAL:** Compile-time configured abstraction providing uniform peripheral interfaces.
3. **VAIOS:** Preemptive RTOS managing task scheduling, timing, and inter-task communication.
4. **Vayu Control Layer:** State estimation, cascaded PID control, and actuation logic.

2.2 Core Components

The system integrates the following functional modules:

- **Sensor Interface:** BMX160 9-axis IMU (accelerometer, gyroscope, magnetometer) sampled at > 1.5 kHz via DMA-driven I2C.
- **RC Input:** iBus protocol over UART with circular DMA buffering.
- **State Estimation:** Complementary and Mahony filters for attitude fusion.
- **Cascaded PID Control:** Outer angle loop (250 Hz) and inner rate loop (1 kHz).
- **Motor Output:** PWM generation via TIM1 at 400 Hz for quadrotor X4 configuration.

2.3 Data Flow Pipeline

Sensor data flows through a deterministic pipeline: Acquisition → Estimation → Control → Mixing → Actuation. Time-critical tasks operate at high priority with bounded jitter, while telemetry and logging run asynchronously at lower priorities.

3 NavHAL: Hardware Abstraction Layer

NavHAL provides a deterministic, compile-time configured interface between hardware and higher-level software. Unlike conventional HALs that introduce runtime overhead, **NavHAL** achieves *zero-cost abstraction* through inline functions and macro-based register access.

3.1 Design Principles

- **Compile-Time Specialization:** Processor, vendor, and board selections resolved at build time.
- **Direct Register Access:** API calls map directly to memory-mapped registers.
- **No Dynamic Dispatch:** Eliminates virtual functions and runtime lookup tables.
- **Execution Independence:** Operates identically in bare-metal or RTOS environments.

3.2 API Structure

NavHAL follows a consistent naming convention: `hal_<peripheral>_<operation>`. Example GPIO usage:

```
hal_gpio_setmode(GPIO_PA05, GPIO_OUTPUT, GPIO_PUPD_NONE);
hal_gpio_digitalwrite(GPIO_PA05, GPIO_HIGH);
```

3.3 Performance Evaluation

Benchmarks on STM32F401RE @ 84 MHz demonstrate **NavHAL**'s efficiency:

Table 1: GPIO Toggle Performance Comparison

Abstraction Layer	Cycles per Pulse
Direct Register Access (LL)	5
NavHAL	**5**
STM32 HAL	26
Arduino Framework	100

Table 2: Interrupt Latency Comparison

Layer	Latency	Dispatch Overhead
NavHAL	33 cycles	16 cycles
STM32 HAL	46 cycles	18 cycles
Arduino	187 cycles	159 cycles

3.4 Portability Model

Porting **NavHAL** requires modifications only at the affected layer:

- **New Board (same MCU):** Update board layer (pin mappings).
- **New MCU (same architecture):** Update vendor layer (register definitions).
- **New Architecture:** Implement new core layer (startup, interrupt handling).

4 VAIOS: Real-Time Operating System

VAIOS provides a lightweight, preemptive execution environment tailored for real-time embedded control. It manages task scheduling, synchronization, memory allocation, and system services.

4.1 Task Model and Scheduling

- **Task Control Block (TCB):** Stores stack pointer, priority, state, and scheduling metadata.
- **Priority-Based Preemption:** Higher numerical values indicate higher priority.
- **Round-Robin within Priority:** Configurable time slicing for equal-priority tasks.
- **Bitmap-Accelerated Selection:** $O(1)$ identification of highest-priority ready task.

4.2 Task States

Tasks transition through: `READY` → `RUNNING` → `BLOCKED/DELAYED` → `TERMINATED`. An idle task reclaims resources from terminated tasks.

4.3 Inter-Task Communication

VAIOS provides:

- Binary and counting semaphores

- Mutexes (with recursive support)
- Lock-free SPSC/MPMC queues
- ISR-safe signaling variants

4.4 Memory Management

A first-fit allocator with block splitting and coalescing manages the heap. All allocations are protected by critical sections. Safety features include magic-value integrity checks and configurable heap watermark monitoring.

4.5 Performance Summary

Table 3: VAIOS Benchmark Results (STM32F401RE @ 84 MHz)

Metric	Value
Context Switch Rate	3,773 switches/sec
Context Switch Cycles	∼22,000
Delay Accuracy (100 ms requested)	101 ms (±1% with 1 ms tick)
FPU sinf Throughput	250,000 ops/sec
FPU sqrtf Throughput	333,333 ops/sec

5 Vayu: Flight Control Layer

Vayu implements the application-level logic for drone stabilization and control, organized as coordinated tasks within the **VAIOS** environment.

5.1 State Estimation

Two sensor fusion algorithms are available:

5.1.1 Complementary Filter

Combines high-frequency gyroscope integration with low-frequency accelerometer/magnetometer corrections:

$$\theta = \alpha(\theta + \omega \cdot dt) + (1 - \alpha)\theta_{\text{acc/mag}}$$

where $\alpha \approx 0.98$ blends the estimates.

5.1.2 Mahony Filter

Quaternion-based nonlinear observer with PI correction:

$$\mathbf{e} = \mathbf{v}_{\text{measured}} \times \mathbf{v}_{\text{estimated}}$$

$$\omega_{\text{corrected}} = \omega + K_p\mathbf{e} + K_i \int \mathbf{e}\, dt$$

$$\dot{q} = \frac{1}{2} q \otimes \omega_{\text{corrected}}$$

5.2 Cascaded PID Control

The control architecture uses two nested loops:

- **Attitude Controller (250 Hz):** Computes desired angular rates from orientation error.
- **Rate Controller (1 kHz):** Tracks desired rates using gyroscope feedback.

The PID formulation includes practical enhancements:

- **Derivative on Measurement:** Avoids derivative kick on setpoint changes.
- **Low-Pass Filtering:** Reduces noise amplification in derivative term.
- **Integral Anti-Windup:** Clamping and back-calculation prevent integrator windup.
- **Feedforward:** Setpoint rate feedforward improves tracking.

5.3 Motor Mixing and Actuation

For an X4 quadrotor configuration, mixing is computed as:

$$
\begin{aligned}
M_1 &= T - R + P + Y \\
M_2 &= T - R - P - Y \\
M_3 &= T + R - P + Y \\
M_4 &= T + R + P - Y
\end{aligned}
$$

where T is throttle and R, P, Y are roll, pitch, yaw commands. Outputs are normalized to PWM bounds and applied via TIM1 at 400 Hz.

5.4 Communication and Telemetry

- **RC Input:** iBus protocol with DMA circular buffering.
- **Telemetry:** Multi-rate packet-based streaming with delta compression for high-frequency data.
- **Command Processing:** Supports heartbeat, identification, calibration, and runtime task management.

5.5 Data Logging

File-based logging using the Virtual File System (VFS) layer with SDIO-connected microSD card:

- Preallocated files ensure deterministic write latency.
- Circular logging prevents storage exhaustion.
- Mutex-protected for thread-safe concurrent access.

5.6 Task Organization

Table 4: Task Priority Assignment

Tier	Priority	Tasks
High	2	IMU acquisition (DMA-driven)
Medium	1	Attitude/Rate control, Motor output
Low	0	Telemetry, Logging, Heartbeat

6 Hardware Platform

The Vayu flight controller is a custom-designed PCB built around the STM32F401RE (Cortex-M4 @ 84 MHz with FPU). Key specifications:

Table 5: Hardware Specifications

Component	Details
MCU	STM32F401RE, 84 MHz, 512KB Flash, 96KB SRAM
IMU	Bosch BMX160 (9-axis: accel, gyro, mag)
Barometer	Bosch BMP180
GPS	u-blox M9N (UART)
Storage	microSD via SDIO
Power	Two-stage: Buck (5V) $\rightarrow$ LDO (3.3V)
Interfaces	UART, I2C, SPI, USB CDC, SWD

The PCB is a 2-layer design created in KiCad, with careful attention to signal integrity, power distribution, and noise isolation for sensitive sensors.

7 Comparison with Existing Stacks

Table 6: Architectural Comparison

Aspect	PX4 / ArduPilot	Vayu
Architecture	Feature-rich, monolithic	Layered, modular
Hardware Support	Broad, vendor-linked	Hardware-agnostic (NavHAL)
Execution Model	General-purpose RTOS (NuttX/ChibiOS)	Custom deterministic scheduler
Complexity	High	Controlled and minimal
Customization	Moderate (complex codebase)	High (modular design)
Abstraction Overhead	Moderate to high	Near-zero (compile-time)

8 Current Status and Roadmap

8.1 Current Status (2026)

The Vayu stack is fully functional with:

- End-to-end sensor-to-actuator pipeline operational.
- Stable real-time execution under **VAIOS**.
- Validated core subsystems in integrated testing.

8.2 Development Roadmap

Short-Term:

- Flight testing and hardware validation
- Sensor calibration automation
- Control loop tuning and optimization
- Enhanced failsafe mechanisms

Mid-Term:

- GPS integration and position control
- Advanced estimation (Extended Kalman Filter)

- Full blackbox flight data recording
- Additional communication interfaces

Long-Term:

- Autonomous navigation stack
- Multi-vehicle coordination
- Porting to STM32H7 and AVR platforms
- Mission planning integration

9 Conclusion

The Vayu flight control stack demonstrates a complete, indigenous solution for embedded drone control systems. By combining a zero-overhead hardware abstraction layer (**NavHAL**), a deterministic real-time kernel (**VAIOS**), and a modular flight control application (**Vayu**), the system achieves:

- **Performance:** Near-register-level efficiency with high-level usability.
- **Predictability:** Bounded jitter and deterministic task execution.
- **Portability:** Layered design enabling migration across MCU platforms.
- **Ownership:** Full control over the entire software stack without external dependencies.

This work establishes a foundation for future development in autonomous aerial systems, prioritizing simplicity, transparency, and adaptability over feature bloat. The modular architecture enables incremental enhancement toward full autonomous capabilities while maintaining the real-time guarantees essential for stable flight.