FluidNC vs grblHAL: Technical Assessment for Custom CNC Firmware Development

grblHAL emerges as the superior foundation for custom firmware development, particularly when advanced motion control and real-time performance are priorities. However, the optimal strategy involves hybridizing both architectures to capture grblHAL's technical superiority while incorporating FluidNC's innovative configuration approach.

Architecture and Codebase Analysis

grblHAL: Superior Technical Foundation

grblHAL demonstrates exceptional software engineering through its **Hardware Abstraction Layer (HAL) architecture**, enabling support for 15+ MCU families including STM32, ESP32, Teensy, and RP2040. GitHub The codebase separates hardware-specific implementations from core GRBL logic The Grbl Project through function pointer-based abstractions: GitHub PrintNC Wiki

```c
typedef struct {
    bool (*driver_init)(void);
    void (*stepper_pulse_start)(stepper_t *stepper);
    spindle_hal_t spindle;
    coolant_hal_t coolant;
    // Platform-agnostic interfaces
} hal_t;
```

This architecture enables **400kHz+ step rates** on high-performance platforms like Teensy 4.1, Wordpress Google Groups compared to ESP32's inherent limitations. GitHub +2 The modular plugin system allows feature additions without core modification, using runtime function pointer redirection for extensibility. GitHub +3

FluidNC: Modern Configuration Excellence

FluidNC's **YAML-based configuration system** represents a revolutionary approach to CNC firmware configuration, eliminating recompilation requirements: fluidnc +3

```yaml
```

```yaml
axes:
  x:
    steps_per_mm: 800
    max_rate_mm_per_min: 2000
    motor0:
      stepstick:
        step_pin: gpio.12
        direction_pin: gpio.14
```

The ESP32-specific architecture leverages advanced hardware features including **RMT peripheral stepping** and **I2S output drivers** for GPIO expansion, FluidNC Wiki while supporting runtime configuration modification through `$` commands and WiFi-based web interfaces. github +4

Motion Control Capabilities Assessment

S-curve Acceleration: grblHAL's Critical Advantage

grblHAL has fully implemented 3rd-order jerk control with configurable jerk parameters ($800, $801, etc.) providing true 7-segment S-curve acceleration profiles. GitHub +2 This implementation demonstrates **measurable improvements in surface finish quality** and mechanical stress reduction during milling operations. github

FluidNC completely lacks S-curve acceleration, utilizing only traditional trapezoidal profiles with linear acceleration ramps. Stack Exchange This represents a fundamental limitation for precision machining applications.

Probing Features Comparison

Both systems support comprehensive **G38.2-G38.5 probing commands**, but with different strengths: GitHub fluidnc

grblHAL advantages:

- Advanced tool length offset (TLO) management with automatic calculation GitHub
- G59.3 coordinate system for repeatable tool measurements GitHub
- Integration with M6 tool change sequences GitHub
- Plugin-extensible architecture for custom probing workflows GitHub

FluidNC advantages:

- Simplified YAML-based probe configuration FluidNC Wiki fluidnc
- Built-in dual-speed probing macro support fluidnc
- Integrated safety features with check_mode_start validation fluidnc
- Web-based probe result visualization Mitov84

Configuration System Analysis

FluidNC's Revolutionary Approach

FluidNC's YAML configuration system enables **real-time parameter modification** without firmware recompilation: fluidnc +2

- **Runtime modification**: `$/axes/x/steps_per_mm=80` changes parameters in volatile memory fluidnc FluidNC Wiki
- **Configuration streaming**: Multiple config files switchable via `$Config/Filename=<file.yaml>` fluidnc +2
- **Web interface integration**: Browser-based configuration editor at fluidnc.local FluidNC Wiki GitHub
- **Validation system**: Comprehensive error checking with detailed startup messages

grblHAL's Traditional Approach

grblHAL uses hierarchical $ settings with plugin support, GitHub requiring compilation for major architectural changes but supporting: GitHub

- **Plugin settings**: Dynamically allocated setting ranges GitHub GitHub
- **External storage**: EEPROM/FRAM plugin support with 100 trillion write cycles GitHub +2
- **Web Builder**: Browser-based firmware configuration eliminating local toolchain requirements GitHub The Grbl Project

Real-time Configuration Capabilities

FluidNC: Superior Runtime Flexibility

FluidNC excels in runtime configuration modification through: FluidNC Wiki GitHub

- **Extensive $ command support** for real-time parameter changes FluidNC Wiki +2
- **WiFi-based configuration streaming** without physical connections
- **Hot-swapping capabilities** between stored configuration files
- **Persistent storage** in ESP32 LITTLEFS filesystem FluidNC Wiki

grblHAL: Limited Runtime Modification

grblHAL provides basic runtime tuning through traditional $ settings but requires firmware recompilation for major configuration changes, though the Web Builder tool simplifies this process. GitHub The Grbl Project

Hardware Platform Support and Performance

grblHAL: Multi-Platform Excellence

15+ supported MCU families including: GitHub +2

- **STM32 series**: F1xx through H7xx with varying performance levels
- **High-performance platforms**: Teensy 4.1 (600MHz, 400kHz+ step rates) Wordpress +2
- **ESP32**: Full networking capabilities with >300kHz performance
- **RP2040**: Raspberry Pi Pico/PicoW support GitHub GitHub

FluidNC: ESP32-Exclusive Optimization

ESP32-only support with sophisticated hardware utilization: GitHub +2

- **Dual-core architecture**: Core 0 for communications, Core 1 for real-time motion
- **RMT engine**: Hardware-assisted stepping FluidNC Wiki with 15-bit timing precision FluidNC Wiki
- **I2S output**: GPIO expansion through shift registers for pin multiplexing FluidNC Wiki

Development and Extensibility Analysis

Code Modularity and Modification Ease

grblHAL demonstrates superior modularity through its HAL architecture enabling: GitHub +2

- **Clean separation** between hardware and application logic GitHub
- **Template-based development** for new processor support The Grbl Project
- **Plugin system** allowing feature additions without core modifications GitHub +4
- **Multiple toolchain support** (PlatformIO, Arduino IDE, STM32CubeIDE)

FluidNC offers good modularity within ESP32 constraints: GitHub GitHub

- **Object-oriented C++ design** with hardware abstraction GitHub
- **Single-platform focus** simplifying development complexity
- **PlatformIO-exclusive** build system with comprehensive tooling GitHub +2

Community Support and Documentation

Both systems maintain **comprehensive documentation** and active communities:

grblHAL:

- GitHub wiki with technical depth GitHub
- ioSender integration for Windows users GitHub
- Web Builder tool reducing development barriers GitHub The Grbl Project
- 46+ repositories covering diverse hardware platforms GitHub GitHub

FluidNC:

- Professional wiki at wiki.fluidnc.com FluidNC Wiki

- Active Discord community for real-time support GitHub FluidNC Wiki
- Extensive configuration example repository GitHub GitHub
- FluidTerm integrated development environment GitHub FluidNC Wiki

Licensing Considerations

Both projects use **GPL v3.0 licensing** with identical implications: TLDRLegal

- **Commercial use permitted** with source code disclosure requirements Quora Stack Exchange
- **Derivative works** must be released under GPL v3.0 if distributed Stack Exchange +3
- **Patent protection** included in GPL v3 license terms Stacklok +2

Strategic Recommendation for Custom Firmware Development

Recommended Hybrid Approach: grblHAL Foundation + FluidNC Configuration System

Use grblHAL as the primary codebase foundation while porting FluidNC's YAML configuration architecture:

Phase 1: grblHAL Base Implementation

1. **Start with grblHAL core** for superior motion control and S-curve acceleration GitHub +2
2. **Leverage existing HAL architecture** for multi-platform support GitHub +3
3. **Utilize proven plugin system** for extensibility GitHub +4
4. **Maintain advanced probing and tool management features** GitHub

Phase 2: Configuration System Integration

1. **Port FluidNC's YAML parser** to grblHAL architecture FluidNC Wiki
2. **Implement runtime configuration streaming** using grblHAL's plugin system github GitHub
3. **Add web interface capabilities** based on FluidNC's ESP3D-WebUI integration GitHub +2
4. **Maintain backward compatibility** with existing grblHAL $ settings

Phase 3: Real-time Streaming Enhancement

1. **Extend YAML system** for real-time parameter modification FluidNC Wiki
2. **Implement configuration hot-swapping** using EEPROM/FRAM plugins GitHub
3. **Add WiFi/Ethernet streaming** capabilities for remote configuration GitHub

Technical Implementation Strategy

Architecture Benefits:

- **grblHAL's proven S-curve acceleration** provides immediate advanced motion control GitHub +2

- **HAL abstraction** enables deployment across optimal hardware platforms GitHub +3
- **Plugin system** simplifies addition of YAML configuration engine GitHub +3
- **Existing real-time performance** maintains precision timing requirements

Configuration System Integration:

- **YAML parser implementation** can be added as a grblHAL plugin github +2
- **Runtime configuration** leverages existing settings infrastructure
- **Web interface** can be integrated using networking plugins GitHub
- **File system support** available through existing SD card plugins github

Development Complexity Assessment

Moderate complexity (3-6 months development time):

- **YAML parser porting**: Well-defined interface requiring C++ adaptation FluidNC Wiki
- **Configuration streaming**: Building on existing plugin architecture
- **Web interface integration**: Leveraging proven ESP3D-WebUI codebase GitHub FluidNC Wiki
- **Cross-platform compatibility**: HAL architecture simplifies hardware abstraction GitHub PrintNC Wiki

Performance and Feature Matrix

Requirement	grblHAL Base	+ FluidNC Config	Custom Result
S-curve Acceleration	✅ Full implementation	➡️ Maintained	✅ Advanced motion control
YAML Configuration	❌ Not available	✅ Full port	✅ Runtime flexibility
Real-time Streaming	⚠️ Limited	✅ Enhanced	✅ Complete solution
Advanced Probing	✅ Superior features	➡️ Enhanced	✅ Best-in-class
Multi-platform	✅ 15+ MCUs	➡️ Maintained	✅ Hardware flexibility
Performance	✅ 400kHz+ capable	➡️ Preserved	✅ Maximum performance

This hybrid approach delivers **superior technical capabilities** while maintaining **modern configuration flexibility**, creating a next-generation CNC firmware that combines the best aspects of both systems while addressing their individual limitations.