Weitflug: A Research-Oriented Fork of Betaflight for Agile Autonomous Flight

Joel Klimont*, Jakob Buchsteiner, Alexander Lampalzer and Radu Grosu
Technical University of Vienna, Austria

ABSTRACT

Agile autonomous quadrotors require high-rate, accurately timestamped data exchange between the embedded flight controller and an onboard computer (OBC). Betaflight provides responsive rate control, DShot motor support, and mature gyroscope filtering, but its MultiWii Serial Protocol (MSP) interface is designed primarily for configuration and telemetry rather than for synchronized, high-rate transfer of low-level state to an onboard computer. We present *Weitflug*, a research-oriented fork of Betaflight that introduces the Flyby Serial Protocol (FSP) and RPM-aware accelerometer filtering. FSP periodically transmits fixed-format packets containing flight-controller timestamps, filtered inertial measurements, attitude, pilot inputs, motor speeds, and battery voltage. An OBC-side clock estimator maps flight-controller timestamps into the OBC time base. On a SPEEDYBEEF7V3 flight controller connected to a Jetson Orin NX, FSP provided a sensor stream at approximately 1.05 kHz. The 1st-to-99th-percentile spread of consecutive sample intervals was 0.007 ms, compared with 3.313 ms for MSP messages timestamped upon reception by the onboard computer. Spectral analysis of flight logs further showed strong attenuation of motor-synchronous accelerometer components after RPM-aware filtering. These results demonstrate a compact interface with flight-controller-side timestamps for using Betaflight as a low-level controller in autonomous multirotor research.

1 INTRODUCTION

Agile autonomous quadrotor flight places strict timing requirements on the sensing-to-actuation pipeline. At high speed, delayed or irregularly timed inertial measurements can degrade estimation and control, while limited access to low-level flight-controller data complicates system integration and analysis [1, 2, 3].

General-purpose autopilot stacks provide mature mission handling and companion-computer interaction, commonly through MAVLink [4, 5]. Betaflight occupies a different design point, emphasizing responsive FPV flight, high-rate rate control, DShot motor interfaces, and a mature tuning workflow [6]. Out of the box, however, it offers little support for synchronized high-rate state exchange with an onboard computer.

This paper presents *Weitflug*, a research-oriented fork of Betaflight[1] that retains Betaflight's embedded control stack while extending it for agile autonomous flight with an onboard computer (OBC). Our main contributions are:

- We introduce the *Flyby Serial Protocol* (FSP), a periodic serial interface with flight-controller-side sample timestamps and OBC-side clock alignment for high-rate state exchange.
- We implement RPM-aware accelerometer filtering by extending Betaflight's motor-speed-dependent notch-filtering infrastructure to acceleration measurements.
- We evaluate the timing information available to downstream software and the attenuation of motor-synchronous accelerometer components on a real flight-controller and onboard-computer setup.

To support reproducibility, the accompanying repository provides the firmware, OBC-side FSP implementation and clock estimator, build configuration, and analysis scripts used for the evaluation.

2 RELATED WORK

2.1 Autopilot and Research Architectures

Research UAV software spans different abstraction levels. PX4 provides a modular embedded robotics framework, Paparazzi an academic airborne and ground-control ecosystem, and ROSflight separates flight-controller firmware from ROS software on a Linux companion computer [4, 7, 8]. Complete platforms such as MRS and Agilicious additionally cover broader autonomy stacks, including estimation, control, simulation, perception, and hardware integration [9, 2]. Weitflug is complementary to these systems: it retains Betaflight's rate controller, receiver path, ESC support, tuning workflow, and filtering pipeline while providing a narrow embedded interface for independently developed OBC software.

*Email address(es): joel.klimont@tuwien.ac.at

[1] Open-source project repository: `https://github.com/F-WuTS/weitflug-open`

Within Betaflight-derived research firmware, INDIflight improves UART telemetry and uplink for offboard estimation and control, adds alternative embedded control architectures, and includes an optional accelerometer RPM filter [10]. Betalink modifies MSP to exchange motor commands and flight-controller state in one transaction and adds an FC-side motor-velocity controller using bidirectional DShot feedback [11]. Weitflug instead focuses on a periodic, fixed-format state stream with source timestamps, OBC-side clock alignment, and an experimentally evaluated accelerometer-filtering path while retaining Betaflight's rate controller. Because these projects expose different control abstractions and communication semantics, we compare them qualitatively rather than treating their reported rates as directly interchangeable performance baselines.

2.2 Serial Interfaces and Measurement Timing

MAVLink is the dominant messaging protocol in autopilot-centered UAV systems and supports telemetry, commands, mission services, and offboard control [5]. Betaflight commonly uses the MultiWii Serial Protocol (MSP) for configuration and selected telemetry functions [12]. MSP is practical for these purposes, but obtaining the low-level measurements needed by an OBC controller requires multiple command-specific requests and responses with different payloads and update rates. A MAVLink-based interface would provide broader interoperability, whereas this work requires a compact, fixed-layout state-and-command exchange with explicit source-timestamp semantics. These requirements motivate the specialized recurring packet format described in Section 4.

Timing is part of the sensing interface, not merely an implementation property. Cheng et al. show that bounded sensing-to-actuation timing and measurement freshness are important in a real-time multirotor controller [1]. Temporal-calibration methods for visual–inertial estimation explicitly estimate sensor time offset because timestamp error degrades fusion consistency, and processing-dependent offsets may vary over time [13, 14]. To handle this, distributed synchronization work commonly models clock offset and skew and estimates them using a recursive filter such as a Kalman filter [15]. Weitflug applies this model to flight-controller timestamps and OBC reception times. This allows the onboard computer to reconstruct when each sample was recorded by the flight controller.

2.3 Motor-Synchronous Vibration and Filtering

Motor and propeller vibration is directly observable in onboard inertial measurements. Ferede et al. report that propeller-induced vibration can severely degrade accelerometer measurements on racing quadrotors and therefore use mechanical vibration damping in their experiments [16]. Flight stacks track rotor-related frequencies rather than relying only on broad low-pass filtering. Betaflight uses bidirectional DShot RPM telemetry to place gyroscope notches at motor fundamentals and harmonics, while ArduPilot supports throttle-, RPM-sensor-, ESC-telemetry-, and in-flight-FFT-based dynamic harmonic-notch tracking for gyroscope signals [17, 18]. Weitflug uses per-motor RPM measurements already available through Betaflight's bidirectional DShot path, applies the resulting notch structure to the accelerometer signal, and exports both filtered acceleration and motor-speed measurements to the OBC. The present evaluation demonstrates spectral attenuation of motor-synchronous components, while estimator benefit, filter delay, and computational cost remain to be quantified.

3 CURRENT BETAFLIGHT ARCHITECTURE AND LIMITATIONS

Betaflight is designed primarily for manual first-person-view (FPV) flight, where the flight controller executes a low-latency inner control loop and the pilot provides high-level commands. In this setting, the gyroscope is the dominant feedback signal for rate control, while communication with external systems is mainly used for configuration, telemetry, and debugging. When Betaflight is used as a low-level platform for autonomous flight, two architectural limitations become apparent: the timing behavior of its serial interface and the limited filtering support for accelerometer measurements.

3.1 MSP-Based Communication

Betaflight commonly communicates with external software through the MultiWii Serial Protocol (MSP). MSP is a request-response protocol: to acquire data such as IMU measurements, the onboard computer (OBC) sends a request to the flight controller (FC). The FC processes incoming requests in the serial task, which polls the UART interface at a rate configured by `serial_update_rate_hz`. Once a request is handled, the FC sends the corresponding response back to the OBC.

This design is convenient for configuration and telemetry, but it is not ideal for deterministic high-rate sensor streaming. Since data is only produced in response to OBC requests, jitter in the OBC request loop, kernel scheduling, UART transmission, or FC-side serial polling directly affect the timing of the received data. If the OBC requests data faster than the serial task processes incoming packets, repeated samples or irregular update intervals may occur.

A further limitation is timestamping. MSP packets are typically timestamped on the OBC when they are received. The timestamp therefore represents packet arrival time, not the sensor acquisition time on the FC. Delays from serial-task scheduling, UART transmission, kernel buffering, interrupt batching, and userspace scheduling are included in the timestamp. As a result, packets generated at regular intervals on the FC may appear irregular on the OBC. Conversely, packets delivered together by the kernel may receive nearly identical timestamps although they were transmitted at different times. Different MSP packet sizes also lead to different

UART transmission times, which further distorts the observed inter-arrival intervals.

3.2 Accelerometer Filtering

Betaflight's sensor filtering pipeline is optimized mainly for gyroscope data. This reflects the requirements of manual FPV flight, where the gyro is the primary signal for the inner rate controller. Betaflight therefore provides extensive gyro filtering, including motor-RPM-aware filtering to suppress vibration components caused by motors and propellers [17].

The accelerometer is less central to typical FPV acro-mode flight and is therefore not filtered with the same emphasis. However, for autonomous flight, the accelerometer can be important for state estimation, inertial sensing, and higher-level control. Motor-induced vibrations are clearly visible in accelerometer measurements and can degrade estimator quality. Since these vibration frequencies depend on motor RPM, the lack of RPM-aware accelerometer filtering limits the usefulness of the accelerometer signal for onboard autonomy.

These limitations motivate Weitflug's two main extensions: the Flyby Serial Protocol (FSP), which periodically sends FC-timestamped state to the OBC, and RPM-aware filtering for accelerometer measurements.

4 WEITFLUG SYSTEM DESIGN

Weitflug adds a narrow research interface to flight controllers running Betaflight without changing the division of responsibilities between the embedded controller and the OBC. The flight controller continues to run IMU acquisition, filtering, attitude estimation, rate control, receiver handling, ESC communication, and motor output. The OBC runs experiment-specific control, estimation, and planning software and receives low-level measurements with flight-controller timestamps.

The exported state is restricted to quantities useful for agile-flight research: filtered accelerometer and gyroscope data, attitude, RC inputs, motor RPM, battery voltage, and timing metadata. This keeps the microcontroller-side interface compact while exposing the signals needed for estimator validation, controller debugging, and post-flight analysis.

4.1 Flyby Serial Protocol

For this purpose, we developed the *Flyby Serial Protocol* (FSP) as Weitflug's runtime serial interface. FSP transfers high-rate state and commands between the flight controller and the OBC, while MSP remains available for configuration. In contrast to MSP, where different pieces of state are distributed across several message types, FSP transmits one recurring packet containing the low-level state needed by the OBC.

FSP uses a fixed binary layout within each compiled protocol variant. As shown in Figure 1, the base FC-to-OBC and OBC-to-FC packets occupy 116 bytes and 24 bytes, respectively. Packets are COBS-encoded and terminated by a zero delimiter [19]. A CRC-8 with initial value `0xFF` and polynomial `0xD5` covers the unencoded packet except for its final checksum byte.

Each FC-to-OBC packet batches two sensor subframes to reduce packet-handling overhead. Each sensor subframe carries its own sample timestamp, filtered accelerometer measurement, accelerometer $1g$ scale, filtered gyroscope measurement and scale, quaternion attitude estimate, current RC values, four motor RPM values, and battery voltage. Including the accelerometer scale in every frame allows the current protocol to use a single self-contained message type, at the cost of retransmitting a configuration value. In future versions, we will explore separating configuration and measurement packets to reduce the transmission of redundant information. The OBC-to-FC payload contains commanded roll, pitch, yaw, and throttle values together with two auxiliary channels. After length, version, COBS, and CRC checks, these values enter Betaflight through the existing MSP receiver path and are interpreted as RC channels.

FSP adopts a periodic, flight-controller-initiated approach, in contrast to the request–response model of MSP. Sensor values are collected and timestamped from the PID-loop path and placed in a five-entry circular queue; two frames are removed for each transmitted packet. The default lockstep configuration sets an upper target of 1100 Hz and selects an integer PID-loop interval that does not exceed it. If the queue is full, the current implementation drops the new sensor frame without transmitting an explicit overflow counter.

For systems that also need to relay MAVLink traffic, FSP provides an optional compile-time tunnel. In this variant, a fixed 64-byte field in each FC-to-OBC packet is reserved for relaying MAVLink messages, increasing the packet size from 116 bytes to 180 bytes. A flag in the version/reserved field identifies the variant. The tunnel is auxiliary to the high-rate state exchange and is not evaluated further in this work.

4.2 Clock Alignment

The flight controller and OBC maintain independent clocks. On the OBC side, their relation is modeled as a linear map

$$t_{o,k} = \alpha_k t_{f,k} + \beta_k, \tag{1}$$

where $t_{f,k}$ is the flight-controller time, $t_{o,k}$ the corresponding OBC time, α_k the relative clock-rate ratio, and β_k the offset. After the first packet is received, both clocks are expressed relative to that reference sample. The Kalman filter state is

$$x_k = \begin{bmatrix} \alpha_k & \beta_k \end{bmatrix}^\top, \tag{2}$$

and is propagated with a random-walk process model

$$x_{k|k-1} = x_{k-1|k-1} + w_k, \qquad w_k \sim \mathcal{N}(0, Q). \tag{3}$$

For a received timestamp pair, the observation model is

$$z_k = H_k x_k + v_k, \qquad H_k = \begin{bmatrix} t_{f,k} & 1 \end{bmatrix}, \tag{4}$$

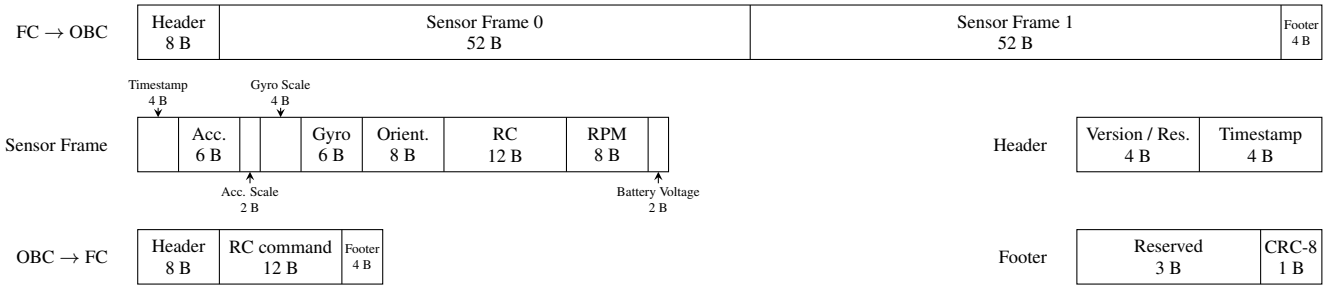


Figure 1: Logical layout of the base FSP packets in the current implementation. The FC-to-OBC packet batches two sensor subframes in one transmission, while the OBC-to-FC packet carries the current RC command set.

with v_k capturing transport delay variation and timestamping noise. In the current implementation, the filter uses $Q = \mathrm{diag}(10^{-10}, 10^{-6})$ and $R = 10^{-3}$, and every 100th received packet is used as an update sample. If the flight-controller timestamp decreases, the estimator resets; if the residual exceeds 0.5 s, which occurs for example when the flight-controller clock is paused during calibration, the reference times are reinitialized.

Once the filter has converged, the corrected OBC time of a packet is predicted from the packet-level timestamp. The two batched sensor samples are then assigned corrected timestamps by subtracting their within-packet offset in the flight-controller time base. This keeps clock estimation on the OBC and avoids adding a two-way synchronization exchange to the FC, while providing downstream estimation and control software with a common time base.

4.3 Integrity and Failure Handling

Packet validity is treated as part of the measurement interface. The receiver rejects packets with unexpected length, an unsupported protocol variant, failed COBS decoding, or an invalid CRC. FSP does not currently transmit a packet sequence number or queue-overflow counter. The OBC can nevertheless infer missing sensor frames from discontinuities in the embedded sample timestamps, but it cannot directly distinguish a queue overflow from other causes of loss.

4.4 RPM-Aware Accelerometer Filtering

Motor-synchronous vibration appears in both gyroscope and accelerometer measurements. Betaflight already uses RPM telemetry to place gyro notches at motor fundamentals and harmonics. Weitflug instantiates the same notch-filter implementation a second time for the three accelerometer axes and returns the filtered values together with motor RPM through FSP. The accelerometer configuration supports up to three harmonics; its current defaults enable the fundamental for each motor, use a minimum frequency of 60 Hz, a 50 Hz fade range, and a notch quality factor of 5. This targets motor-synchronous acceleration components without requiring a broadly lower low-pass cutoff.

5 EVALUATION

The evaluation benchmarks the two main extensions in Weitflug. First, we compare the timing behavior available to downstream software when transferring IMU data from the flight controller to the OBC using FSP and MSP. We then qualitatively assess the effect of RPM-aware accelerometer filtering using real flight logs.

5.1 Experimental Setup

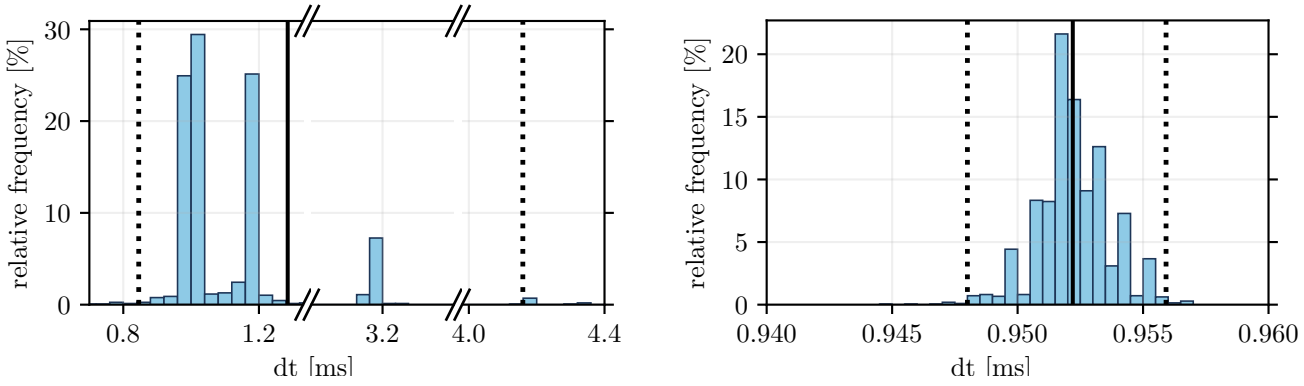
The test platform consists of a SPEEDYBEEF7V3 flight controller with an STM32 F7 processor and a Jetson Orin NX 16GB as the OBC, connected through MSP or FSP. The flight controller is configured for a nominal control-loop frequency of 3.2 kHz; the scheduler reports an actual frequency of 3148 Hz. We additionally configured the Betaflight parameter `serial_update_rate_hz` to 1000 Hz, because MSP messages were otherwise not processed responsively enough for the timing analysis. Acceleration is measured by the BMI270 six-axis IMU on the flight controller, and the electronic speed controller provides the per-motor RPM measurements used for filtering.

5.2 Timing Analysis of FSP and MSP

For FSP, the OBC records both packet-reception times and decoded, clock-aligned sensor-frame timestamps. For MSP, we request IMU data at 1000 Hz and use reception times because the protocol responses do not include the corresponding flight-controller sample timestamps. We additionally request data also carried by FSP, including RC commands and attitude at 100 Hz and battery voltage at 10 Hz, to make the serial workload more representative. The evaluated firmware sets the lockstep target to 1100 Hz. With the measured 3148 Hz PID-loop frequency, the integer lockstep interval is three iterations, yielding a nominal source-sample rate of approximately $3148/3 = 1049.3$ Hz.

The MSP result should therefore be interpreted as an application-level baseline under a representative mixed telemetry workload, rather than as the maximum standalone IMU throughput achievable with MSP.

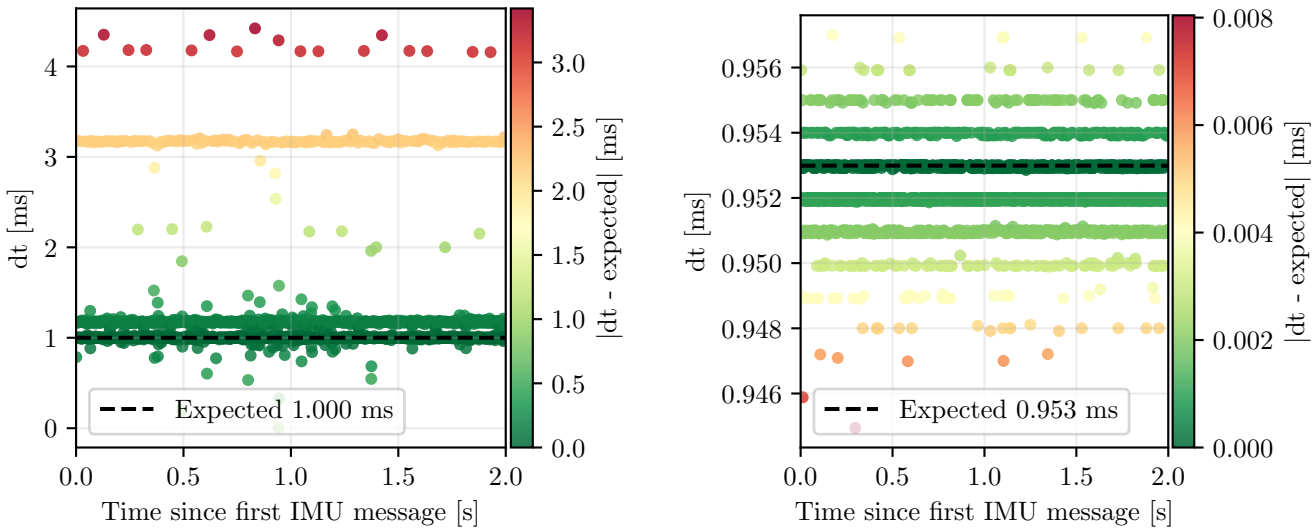
We report the 1st-to-99th-percentile spread of consecutive timestamp deltas and refer to this quantity as jitter. Table 1



(a) Betaflight with MSP

(b) Weitflug with FSP

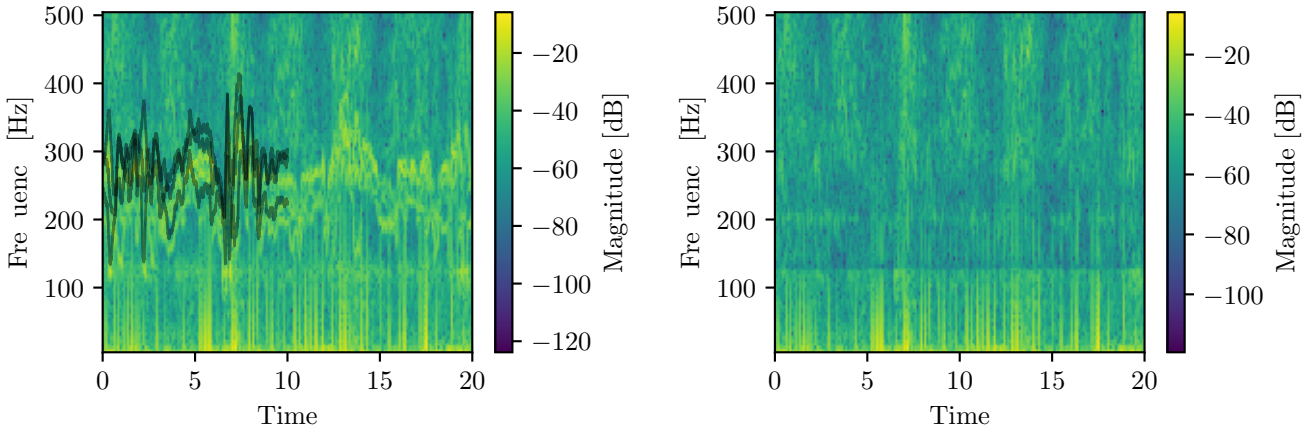
Figure 2: Distribution of timestamp delta times between MSP and FSP. The solid line indicates the mean, while the dotted lines denote the 1st and 99th percentile. The MSP distribution has multiple modes, one around the expected interval of 1 ms, and three additional ones around 1.2 ms, 3.2 ms and 4 ms. FSP shows a narrow, quantized distribution around the expected interval, with the remaining spread attributed to flight-controller scheduling and timer quantization.



(a) Betaflight with MSP

(b) Weitflug with FSP

Figure 3: Timestamp deltas between consecutive IMU messages for MSP and FSP. Subfigure (a) shows periodic MSP reception-time outliers of up to approximately four times the expected interval, whereas they are absent from the FSP source-time sequence in subfigure (b). MSP timestamps are captured on reception by the OBC, while FSP timestamps originate on the flight controller; the plots therefore show the timing information available to downstream software rather than an equal-condition measurement of transport latency.



(a) Unfiltered accelerometer with overlaid motor speed in revolutions per second (black, first half only)

(b) RPM-aware filtered accelerometer

Figure 4: Short-time Fourier analysis of accelerometer vibration before and after RPM-aware filtering. The unfiltered spectrum exposes motor-synchronous bands whose time-varying frequency follows the RPM trajectory. The filtered spectrum suppresses these bands, reducing motor-induced components in the accelerometer signal. Sampling frequency 1000 Hz, 0.2 s Hann window with 50% overlap.

Table 1: Timestamp deltas between consecutive IMU samples

Metric	MSP [ms]	FSP [ms]
Expected	1.000	0.953
Mean	1.284	0.952
Median	1.003	0.952
1st percentile	0.845	0.948
99th percentile	4.159	0.956
P99–P1 spread	3.313	0.007

and figs. 2 and 3 summarize the results. Under the timestamping conventions above, the FSP source-time sequence has substantially lower interval spread than the MSP reception-time sequence. This comparison quantifies the regularity of the sample timeline available to downstream software; it does not measure or remove the constant component of one-way transport latency.

FSP reports a mean rate of $f_{\mathrm{mean,FSP}} = 1/dt_{\mathrm{mean,FSP}} = 1050$ Hz. The small difference from the selected 1049 Hz rate may arise from clock-rate mismatch, timer quantization, and rounding in the synchronization estimate; a dedicated clock-accuracy evaluation is left for future work. MSP reaches a mean reception rate of $f_{\mathrm{mean,MSP}} = 1/dt_{\mathrm{mean,MSP}} = 778$ Hz in this configuration.

5.3 *RPM-Aware Accelerometer Filtering*

The accelerometer filter is evaluated from flight logs containing both unfiltered and filtered acceleration signals. Figure 4 compares short-time Fourier transforms of the acceleration magnitude over the same motor-speed sweep. In the unfiltered signal, motor-synchronous ridges remain sufficiently pronounced that the RPM trajectory can be read directly from the spectrum. After RPM-aware filtering, these ridges are strongly attenuated and no longer provide a reliable RPM trace. This result demonstrates attenuation of the dominant motor-synchronous components; quantifying total noise reduction, estimator impact, and filter delay remains future work.

6 Conclusion

We presented Weitflug, a research-oriented Betaflight fork that provides periodic, flight-controller-timestamped state through FSP and extends RPM-aware filtering to accelerometer measurements. In our evaluation, the 1st-to-99th-percentile spread of consecutive IMU intervals decreased from 3.313 ms for MSP reception timestamps to 0.007 ms for the FSP source-time sequence while maintaining a stream above 1000 Hz. Flight-log analysis also showed attenuation of motor-synchronous accelerometer components. Together, these extensions make Betaflight more suitable as a low-level controller for autonomous multirotor research.

The current evaluation does not measure absolute one-way transmission latency, and the one-way clock-alignment method cannot separate a constant transport delay from clock offset. This distinction is especially relevant when FSP timestamps are fused with measurements from sensors connected directly to the OBC. Future work should evaluate latency us-

ing an external reference or two-way exchange and quantify accelerometer-filter attenuation, phase delay, computational cost, and estimator impact over multiple flights. Timestamped command queuing could additionally improve input–output timing repeatability.

REFERENCES

[1] Zhuoqun Cheng, Richard West, and Craig Einstein. End-to-end analysis and design of a drone flight controller. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(11):2404–2415, 2018.

[2] Philipp Foehn, Elia Kaufmann, Angel Romero, Robert Penicka, Sihao Sun, Leonard Bauersfeld, Thomas Laengle, Giovanni Cioffi, Yunlong Song, Antonio Loquercio, and Davide Scaramuzza. Agilicious: Open-source and open-hardware agile quadrotor for vision-based flight. *Science Robotics*, 7(67):eabl6259, 2022.

[3] Antonio Loquercio, Elia Kaufmann, René Ranftl, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Learning high-speed flight in the wild. *Science Robotics*, 6(59):eabg5810, 2021.

[4] Lorenz Meier, Dominik Honegger, and Marc Pollefeys. PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6235–6240, 2015.

[5] Anis Koubaa, Azza Allouch, Maram Alajlan, Yasir Javed, Abdelfettah Belghith, and Mohamed Khalgui. Micro Air Vehicle Link (MAVLink) in a Nutshell: A Survey. *IEEE Access*, 7:87658–87680, 2019.

[6] Betaflight Development Team. Betaflight Documentation. Online documentation. Accessed: 2026-06-15.

[7] Balazs Gati. Open source autopilot for academic research: The Paparazzi system. In *2013 American Control Conference*, pages 1478–1481, 2013.

[8] James Jackson, Gary J. Ellingson, and Timothy W. McLain. ROSflight: A lightweight, inexpensive MAV research and development tool. In *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 758–762, 2016.

[9] Tomáš Báča, Matěj Petrlík, Matouš Vrba, Vojtěch Spurný, Robert Pěnička, Daniel Hert, and Martin Saska. The MRS UAV system: Pushing the frontiers of reproducible research, real-world deployment, and education with autonomous unmanned aerial vehicles. *Journal of Intelligent & Robotic Systems*, 102, 2021. Article 26.

[10] INDIflight Development Team. INDIflight: Research-focused fork of betaflight 4.4.2. Open-source repository. Accessed: 2026-07-24.

[11] Betalink Development Team. Betalink: Low-latency communication with betaflight. Open-source repository. Accessed: 2026-07-24.

[12] Betaflight Development Team. MSP Protocol Reference for Developers. Online documentation. Accessed: 2026-06-15.

[13] Mingyang Li and Anastasios I. Mourikis. Online temporal calibration for camera-IMU systems: Theory and algorithms. *The International Journal of Robotics Research*, 33(7):947–964, 2014.

[14] Yonggen Ling, Linchao Bao, Zequn Jie, Fengming Zhu, Ziyang Li, Shanmin Tang, Yongsheng Liu, Wei Liu, and Tong Zhang. Modeling varying camera-IMU time offset in optimization-based visual-inertial odometry. In *Computer Vision - ECCV 2018*, pages 491–507. Springer International Publishing, 2018.

[15] Brent R. Hamilton, Xiaoli Ma, Qiang Zhao, and Jun Xu. ACES: Adaptive clock estimation and synchronization using Kalman filtering. In *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking*, MobiCom '08, pages 152–162, 2008.

[16] Robin Ferede, Till Blaha, Erin Lucassen, Christophe De Wagter, and Guido C.H.E. De Croon. One net to rule them all: Domain randomization in quadcopter racing across different platforms. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6357–6363, 2025.

[17] Betaflight Development Team. DShot RPM Filtering. Online documentation. Accessed: 2026-06-15.

[18] ArduPilot Development Team. Managing gyro noise with dynamic harmonic notch filters. Online documentation. Accessed: 2026-06-15.

[19] Stuart Cheshire and Mary Baker. Consistent overhead byte stuffing. *IEEE/ACM Transactions on Networking*, 7(2):159–172, 1999.