Research Paper[기술논문인 경우 Technical Paper 표기]
Journal of Aerospace System Engineering
Vol.#, No.#, pp.#-# (20##)
EISSN 2508-7150
http://dx.doi.org/10.20910/JASE.####.##.#.#

AirSim 연동 고주파 제어 벤치마크·안정화 툴킷

김경목[1,†], 이남훈[2], 김성권[3]

[1] 서울과학기술대학교
[2] 서울과학기술대학교
[3] 서울과학기술대학교

AirSim-Integrated High-Frequency Control Benchmark-Stabilization Toolkit

Kyungmok Kim†, Namhun Lee[2] and Seongkwon Kim[3]

[1] Seoul National University of Science and Technology
[2] Seoul National University of Science and Technology
[3] Seoul National University of Science and Technology

Abstract

We present a high-rate simulation-control loop that connects AirSim to an external Software-in-the-Loop (SIL) controller through an in-process shared-memory pipeline. AirSim publishes timestamped high-rate sensor streams (kHz-class inertial, 100-Hz-class environmental, and GNSS) to a lock-free ring buffer; the SIL controller consumes the streams, executes an attitude Proportional (P) and rate Proportional-Integral (PI) plus filtered Derivative (D) controller (PID structure), and returns 400-Hz PWM signals through the same path. Compared with Remote Procedure Call (RPC)-based integration, the proposed pipeline sustains target update rates while significantly reducing latency and jitter. Under injected motor/gyro biases and dynamic accelerations, the loop reduces peak-to-peak (p-p) spikes and saturation occupancy, achieves yaw stop (rate→0), and improves attitude root-mean-square (RMS) error. Because the SIL controller mirrors the onboard flight controller (FC) structure, algorithms validated in AirSim can be transferred with minimal code modification. The proposed framework provides a practical recipe and reference implementation for simulator and Hardware-in-the-Loop (HIL) development, where high-frequency sensing and actuation are essential.

초 록

본 논문은 AirSim과 SIL 제어기를 공유 메모리로 직결한 고속 센싱–제어 폐루프를 제안한다. AirSim은 고속 센서 데이터를 락프리 링버퍼로 송출하고, SIL 제어기는 이를 소비하여 PID 제어를 수행한 후 400Hz PWM 신호를 반환한다. 이 방식은 RPC 대비 지연과 지터를 줄여 목표 주파수를 안정적으로 유지한다. 실험 결과, 외란 환경에서도 출력 진동과 포화율이 감소하고 자세 제어 성능이 향상됨을 확인하였다. 특히 제어기가 실기체 FC와 동일한 구조를 가져 코드 이식이 용이하며, 고주파 센싱과 구동이 필요한 시뮬레이터 및 HIL 개발에 실용적인 참조 모델을 제공한다.

Key Words : AirSim, shared-memory pipeline, Software-in-the-Loop (SIL) controller, high-rate sensing, attitude control, latency/jitter reduction

1. 서 론

소형 멀티로터의 자세 제어는 모터 및 기체의 비대칭, 센서 바이어스, 추정기 게인 전환(gain switching) 등으로 인해 출력 스파이크와 장기적인 yaw 드리프트가 빈번하게 발생한다. 본 연구에서는 AirSim과 Software-in-the-Loop (SIL) 제어기를 프로세스 내 공유 메모리 파이프라인으로 직접 연결하여, 타임스탬프 기반의 관성측정장치(Inertial Measurement Unit, IMU) 1000 Hz, 기압(Barometer) 100 Hz, 자기(Magnetometer) 100 Hz, GNSS 10 Hz 센서 데이터를

Received: Mo ##, Year Revised: Mo ##, Year Accepted: Mo ##, Year
[1]Principal Research Engineer, [2]Senior Research Engineer, [3]Professor
† Corresponding Author
Tel: +82-##-####-####, E-mail: ABCD@sase.or.kr
ORCID : ####-####-####-####

실시간으로 교환하는 고속 센싱-제어 루프를 구현하였다. 구체적으로, 본 시스템은 GNSS(위치) 10 Hz의 센서 스트림을 소비하고, ESC/로터에 전달되는 PWM(로터 구동 명령) 400 Hz를 생성하는 고속 센싱-제어 폐루프를 제안하였다. 제안된 루프는 SIL 제어기에 자세 비례(P)와 각속도(회전 속도) 비례-적분-미분(PI+D) 제어기를 배치하고, yaw 축에 anti-windup 기능이 포함된 각속도 PI 제어기, 자이로 D-term PT2×2 필터 및 출력 클램프, 호버 트림 추정 후 램프-인(ramp-in), Mahony 필터[7] 기반 가속도 신뢰도 게인 스무딩, EST/GT 연속 블렌딩, 모터 Slew rate 제한 등을 결합하였다. RPC 기반 통합 대비 제안된 파이프라인은 지연(latency)과 지터(jitter)를 낮추면서 목표 주파수를 안정적으로 유지하며, 모터 및 자이로 바이어스나 동적 가속 조건에서 peak-to-peak(p-p) 진폭과 포화율을 감소시키고, yaw 정지(각속도→0)와 자세 root-mean-square(RMS) 오차를 개선하였다. 또한 SIL 제어기가 실제 비행 제어기(Flight Controller, FC)의 구조를 반영하므로, 시뮬레이터에서 검증한 알고리즘을 최소한의 코드 변경으로 실기체에 이식할 수 있다. AirSim이나 Gazebo 환경에서 PID제어기, 필터, 튜닝 기법에 대한 연구는 다수 보고되었으나, AirSim-공유메모리-SIL 제어기로 구성된 고속 센싱-제어 루프(IMU 1000 Hz, Barometer/Magnetometer 100 Hz,

GNSS 10 Hz, PWM 400 Hz)를 일관 유지하며, RPC와의 정량적 비교(지연·지터, p-p, 포화율, yaw-rate, RMS)를 제시한 공개 연구는 제한적이었다 [12]. 또한 제안된 공유메모리 기반 분리 구조는 SIL 기능을 비행 제어기(FC) 개발과 모듈 경계로 분리하여 시뮬레이터와 독립적인 빌드·실행을 가능하게 하였으며, 이로써 컴파일 시간과 배포 복잡도를 줄이고 반복 개발 속도를 향상시켰다.

2. 관련 연구

2.1 오픈소스 오토파일럿의 구조와 실무 튜닝

ArduPilot과 PX4는 외부 자세 제어 루프(P)와 내부 각속도 제어 루프(PID 또는 PI+D)로 구성된 2중 루프 구조를 표준으로 채택하고 있으며, 실기체의 진동, 센서 노이즈, 출력 포화 등에 대응하기 위해 D-term 저역통과 필터(PT1/PT2)와 노치 필터, 적분 제한 및 프리즈(anti-windup), 모터 출력 Slew rate 제한, Feedforward 제어, 센서 필터 체인(자이로/가속도) 등의 기능을 제공한다. 개별 기능의 효과와 튜닝 지침은 풍부하지만, 여러 기법을 동시에 적용했을 때의 상호작용을 어블레이션(ablation) 방식으로 체계적으로 정리한 공개 자료는 제한적이다 [1], [2].

2.2 PID 적분기: anti-windup·bumpless 전이

포화 또는 Slew rate 제한이 존재하는 액추에이터에서는 적분기의 과적분이 오버슈트와 긴 복구 시간을 유발할 수 있다. 이를 방지하기 위해 적분 제한(clamping), 백계산기반 추종형 anti-windup(back-calculation), 조건부 적분(conditional integration), I-freeze/decay(포화 시 적분 동결·감쇠), 그리고 목표치나 제어 게인 변경 시 출력 불연속을 완화하기 위한 bumpless transfer 등이 적용된다. 멀티로터에서도 속도 루프의 정상상태 바이어스를 제거하기 위해 rate-PI 제어에 anti-windup을 일반적으로 적용한다 [3], [4].

2.3 D-term 필터링과 미분 성형

미분항은 고주파 노이즈와 기체의 고유 진동 모드에 특히 민감하므로, D-term 전용 저역통과(PT1/PT2), 정적/동적 노치(dyn-notch; 모터 RPM/고조파 추적), 그리고 차분기성형(differentiator shaping; lead-lag/biquad 기반)을 조합해 고주파 성분을 억제한다. 실무에서는 (i) 자이로 전역필터와 분리된 D-term 전용 필터 체인을 구성하고, (ii)스파이크 억제를 위해 D-클램프(D 출력 한계/소프트 리미트)를 병행하며, (iii) 모터 Slew rate 제한으로 단발 스파이크와 포화 점유율을 줄인다. 또한 D-term은 일반적으로 측정값(각속도) 기반 미분으로 구현되므로, 필터 차수·차단주파수 선택 시 위상 지연과 이득 여유를 함께 고려해야 한다 [5],[6].

2.4 소스 전환(EST/GT)과 연속 블렌딩

센서 또는 추정 소스의 전환(예: 추정값(EST)↔ 참값(GT), 센서 페일오버)은 단계적인 전이(step change)가 발생할 경우 제어 루프에 큰 버스트를 유발할 수 있다. 이를 방지하기 위해 일정 시간 동안 가중치를 0에서 1로 선형 증가시키는 연속 블렌딩(continuous weighting)을 적용하여 전환을 부드럽게 만드는 기법이 센서퓨전 및 로버스트 필터링 분야에서 제안되어 왔다 [9].

2.5 액추에이터 제약: 포화·Slew 제한

액추에이터의 포화(saturation)와 출력 변화율 제한(rate limit)은 anti-windup 설계의 핵심 전제 조건이며, 실제 시스템에서는 출력 Slew(명령 변화율 제한)를 통해 전기적·기계적 과도 응답을 억제하고, 구동부 보호 및 제어 명령의 연속성을 확보한다.

멀티로터 제어기에서도 모터별(per-motor) Slew rate 제한 설정이 일반적으로 적용된다 [10],[11].

3. 방법/시스템 구현

3.1 시스템 개요

본 시스템은 AirSim-공유 메모리-SIL 제어기로 구성된 파이프라인을 통해 타임스탬프가 부여된 텔레메트리 스트림을 루프 기준 1 kHz 로 수신(벤치마크 시 IMU 퍼블리셔는~100 kHz 까지 지원)하고, PWM 구동 명령을 400 Hz 로 송신하도록 설계하였다. 단일 생산자-단일 소비자(SPSC)링버퍼 구조, 비블로킹 소비(non-blocking consume), zero-copy 전달, 고해상도 시계(high-resolution clock) 동기화를 적용하여 통신 지연과 지터를 최소화하였다. 본 논문에서 사용하는 기호는 다음과 같다: q 는 자세 쿼터니언, ex, ey 는 소각 오차(small-angle attitude error), 그리고 ωx,ref, ωy,ref 는 각속도 참조 신호를 각각 나타낸다.

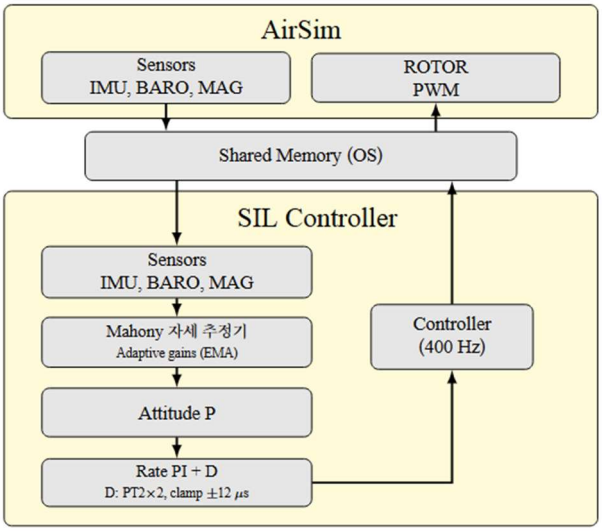

Fig. 1 AirSim-Shared Memory-SIL block diagram

3.2 AirSim 센서(IMU, BARO, MAG)

AirSim에서 생성된 센서 데이터는 운영체제 공유 메모리(IPC) 세그먼트에 producer로서 기록된다. IMU(ACC, GYRO)는 고주기(기본 1 kHz, 벤치마크 시 ~ 100 kHz)로, 반면 기압계(BARO)와 자기계(MAG)는 중·저주기(예:10-100 Hz)로 샘플링되며, 모든 레코드는 지연 최소화를 위한 고해상도 타임스탬프를 포함한다. 센서 종류가 다르더라도 동일한 IPC 경로를 사용하도록 설계하여, 센서별 주파수만 다르게 운용할 수 있다.

3.3 Shared Memory (OS IPC)

공유 메모리는 단일 생산자-단일 소비자(SPSC) 링 버퍼 구조로 설계되었으며, 비블로킹 접근(non-blocking access)과 zero-copy 방식을 통해 불필요한 데이터 복사를 제거한다. IMU처럼 1 kHz급 고주기 데이터와 BARO/MAG처럼 저주기 데이터가 혼재할 수 있도록, 각 레코드는 "센서 타입 + 타임스탬프" 메타데이터를 포함한다. Producer는 AirSim 센서 모듈(IMU, BARO, MAG write)이며, Consumer는 SIL 제어기(IMU, BARO, MAG read)로 역할이구분된다. 이들은 lock-free SPSC 인터페이스를 통해 실시간 텔레메트리 데이터를 교환한다.

3.4. SIL 제어기 센서 입력

SIL 제어기는 공유 메모리에서 도착한 레코드를 비블로킹으로 소비하고, 센서 타입에 따라 서로 다른 파이프라인으로 전달한다. 예를 들어 IMU(ACC, GYRO)는 상태추정기(EKF/Complementary)를 거쳐 제어기(ATT/POS)로 전달되며, BARO는 고도 추정기로, MAG는 Yaw/Heading보정기로 각각 전달된다. 이와 같은 순서로 처리하여 실제 비행제어기와 유사한 구조를 모사한다.

3.5. Attitude P

외부 Attitude P 제어기는 작은 각도 오차(small-angle error)로부터 각속도 참조 신호를 생성한다.

3.6. Rate PI + D, Yaw Rate-PI

내부 Rate PI 제어기는 조건부 적분과 적분항 제한(I-limit, anti-windup)을 적용한다. 여기서 I항은 적분항(In-tegral, I-term)을 의미한다. D-term은 자이로율을 2단 PT2필터로 처리한 후, 출력 값을 ±12 μs 범위로 클램프한다. Yaw 축은 절대 방위(Heading) 제어 대신 angle-P를 0으로 두고, rate-PI(및 소량의 D-term)을 적용하여 각속도 0 rad/s로의 수렴(Rate Damping)을 우선한다.

3.7. Controller

이륙 초기 과도 응답을 배제하고 호버링 평형점(Trim)을 학습하기 위해 3 s 동안의 평균 (ex, ey)를 계산하여 이를 80 μs/rad의 변환 비율로 각도-출력 비례값으로 변환한다. 그 결과를 ±50 μs 범위에서 포화시킨 후, 제어권 전환충격을 줄이기 위해 0.8 s 동안 선형 램프-인(Soft Start) 방식으로 목표값을 적용한다(믹서 이전). 또한, 이산 스위치대신 1차 EMA(τ ≈

0.25 s)를 이용해 가중치 w를 갱신하고, e = (1 − w)eest + wegt 형태로 추정값과 실제값을 블렌딩함으로써 단계 입력(step input)에 따른 급격한 변화를 제거한다.

3.8. ROTOR PWM

SIL 제어기는 컨트롤러에서 생성된 PWM 신호를 400 Hz 로 Shared Memory 에 write(producer) 하고, Air-Sim 의 ROTOR 는 이를 read(consumer)하여 추력과 토크를 계산한다. Slew rate 제한과 출력 클램프는 SIL 제어기 측에서 적용되어 급격한 출력 변화율을 완화한다.

4. 실험 및 결과

4.1. 실험 환경

- **OS/플랫폼: Windows 11, AirSim (월드 및 기체 파라미터 고정)**
- **I/O 경로: In-process Shared Memory 기반 IPC (IMU 1 kHz, PWM 400 Hz)**
- **실행 설정: 제어 스레드 우선순위 상향, 타이머 분해능 1 ms**

4.2. 시스템 구성 및 실행 절차

AirSim (producer) 은 IMU 데이터를 1 kHz 주기로 Shared Memory에 write하고, SIL 제어기 (consumer)는 이를 read하여 Mahony 기반 자세(P)-속도(PI+ D) 루프를 수행한 뒤 PWM 신호를 400 Hz로 write한다. AirSim은 해당 PWM 데이터를 read하여 모터 모델에 적용한다.

두 모듈은 별도의 프로그램 (프로세스)으로 동작하며, Shared Memory는 AirSim이 생성·소유하고 SIL 제어기가 attach하여 read/write한다 (IMU: AirSim→SIL, PWM: SIL→AirSim).

프로그램 실행 시 내부적으로 다음 절차가 순차적으로 수행된다.

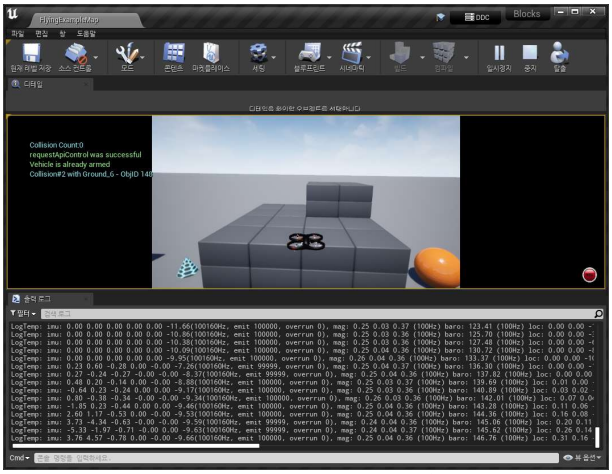

Fig. 2 비행 중 센서 주파수 측정 화면

1. AirSim 초기화: AirSim 모듈을 구동하여 월드 및 기체 파라미터를 고정하고, 시뮬레이터의 시간 및 물리 시뮬레이션 간의 Δt 일관성을 검증한다.
2. SIL 제어기 설정: SIL 제어기를 시작하고 메인 제어 스레드의 우선순위를 상향하며, 타이머 분해능을 1 ms로 설정한다.
3. Shared Memory 연결: IMU(센서) 및 PWM 세그먼트에 대해 Shared Memory를 생성·연결하고, producer/consumer 상태를 점검한다.
4. 데이터 모니터링: write/read 카운터의 주기적 증가 여부를 확인하며, 데이터 drop 또는 overwrite 비율이 0(또는 < 0.5%)임을 검증한다.
5. 제어 루프 검증: IMU 입력 주파수(~1 kHz)와 PWM 출력 주파수(400 Hz)를 측정하여, 허용 오차가 ±5 % 이내임을 확인한다.
6. 워밍업 및 기록: 워밍업 10 s 이후 원시 IMU 데이터, 추정 자세, 제어 명령, PWM, 타임스탬프를 동기화하여 기록한다.
7. 시나리오 실행: 실험 시나리오(S1–S4)를 동일한 초기 조건에서 순차적으로 수행하고, 각 러닝(run)의 로그 및 상태를 저장한다.
8. 종료 및 분석: 종료 시 로그의 무결성을 검사한다 (타임스탬프 단조 증가, 샘플 수, 누락률 등). 이상이 없을 경우 분석 스크립트를 통해 성능 지표를 산출한다.

4.3. 구현 핵심 요약 및 코드 발췌

Listing 1: IMU producer pseudocode

```
// IMU producer pseudocode (executed at simulation rate)
struct ImuSample {
  int64_t ts_ns;              // timestamp [ns]
  float ax, ay, az;           // acceleration [m/s^2]
  float gx, gy, gz;           // angular velocity [rad/s]
```

```
};

SpscQueue<ImuSample> imuQ; // single-producer, single-consumer queue (shared memory)

void onSimStep() {
  ImuSample s;
  s.ts_ns = nowNanos();
  readImu(&s.ax, &s.ay, &s.az, &s.gx, &s.gy, &s.gz);
  imuQ.try_enqueue(s);              // non-blocking enqueue; drops oldest if full
  }
```

Listing 2: IMU consumer

```
optional<ImuSample> tryReadImu() {
  ImuSample s;
  if (imuQ.try_dequeue(s))
    return s; // new IMU sample available
    return nullopt; // no new sample; caller keeps previous one
}
```

Mahony 자세 추정 핵심 (EMA 및 바이어스 추정)
• 각 시점에서 자이로스코프 측정값을 이용해
쿼터니언을 시간 간격 Δt 동안 적분하고, 수치적 오차 누적을 방지하기 위해 매 스텝 정규화를 수행한다.
• 가속도 벡터의 크기가 중력 크기(|a| ≈ g)에 근접한 구
간에서만 가속도 기반 보정 게인을 적용하며, 이때의 신뢰도(weight)는 시상수 τ를 갖는
지수이동평균(EMA)으로 시간적으로 평활화한다.
• 정지 또는 저가속 상태에서는 자이로 바이어스를 EMA 형태로 온라인 추정하여 장기 드리프트를 점진적으로보정한다 (online bias tracking).

Listing 3: Mahony update loop with EMA-based accel trust and bias estimation

```
struct Mahony {
  Quat  q;            // attitude quaternion
  Vec3  bias;         // gyro bias estimate
  Vec3    integ;              // integral (I-term) accumulator
  float kp, ki;      // proportional & integral gains
  float tau_w;          // EMA time-constant for accel-trust
};
void update(Mahony& m, const ImuSample& s, int64_t prev_ns) {
  float dt = max(1e-4f, (s.ts_ns - prev_ns) * 1e-9f);
  // (1) Normalize accelerometer and compute gravity misalignment
  Vec3 a = normalize({s.ax, s.ay, s.az});
  Vec3 g_ref = rotate(conj(m.q), {0, 0, 1});
  Vec3 e = cross(a, g_ref);                    // direction to align sensed gravity with body-z
  // (2) Exponential moving average (EMA) for accel-trust
  float alpha = 1.f - expf(-dt / m.tau_w);  // 0..1 smoothing factor
  float trust_raw = clamp(1.f - fabsf(length(a) - 1.f), 0.f, 1.f);  // |a|≈g → high trust
  static float trust = 0.f;
  trust = (1 - alpha) * trust + alpha * trust_raw;
  // (3) PI correction on gyro signal
  m.integ += (m.ki * trust) * e * dt;
  Vec3 omega = {s.gx, s.gy, s.gz} - m.bias + (m.kp * trust) * e + m.integ;
  // (4) Integrate quaternion and renormalize
  m.q = integrateOmega(m.q, omega, dt);        // small-angle or exponential map integration
  m.q = normalize(m.q);
  // (5) Online gyro bias update (EMA) under quasi-static condition
  bool still = (length({s.gx, s.gy, s.gz}) < 0.15f) && (fabsf(length(a) - 1.f) < 0.1f);
  if (still) {
    float beta = 0.02f;                        // bias EMA rate
    m.bias = (1 - beta) * m.bias + beta * ({s.gx, s.gy, s.gz} - omega);
  }
}
```

4.4. 실험 환경

본 연구에서는 필터의 강건성 및 외란 내성(robustness and disturbance rejection)을 가하기 위해 다음의 네 가지 시나리오를 구성하였다.

S1 Hover: 외란이 없는 정지 호버 상태로, 필터의 정상상태 안정성과 잔류 진동 수준을 평가한다.

S2 Gyro Bias: 자이로 바이어스(±0.02 ~ 0.05 rad/s)를 인위적으로 주입하여, 오프셋 보정 성능 및 장기 안정성을 검증한다.

S3 Dynamic Accel: 0.3 ~ 0.8 g 진폭, 0.5 ~ 2 Hz 주파수 범위의 가속도 외란을 인가하여, 가속도 기반 보정 루프의 외란 내성을 평가한다.

S4 Aggressive Tilt: 20 ~ 35° 범위의 급격한 자세 명령(램프 또는 스텝 입력)을 통해, 큰 각속도·가속도 조건에서의 필터 수렴 특성을 분석한다.

모든 시나리오는 워밍업 10 s 이후 분석되며, 세부 반복 횟수와 분석 윈도우는 섹션 4.5(프로토콜 및 지표)에 따른다.

4.5. 프로토콜 및 지표

각 시나리오는 공통된 초기 조건에서 시작하며, N = 20 회 반복 수행하였다. 시뮬레이션은 워밍업 구간(0–10 s)과 과도응답 구간(10–30 s)을 거친 후, 30 s 부터 120 s 까지의 구간을 분석 대상으로 설정하였다.

평가 지표는 다음과 같다. (1) Peak-to-Peak(μs): 모터 PWM 신호의 최대-최소 차이. (2) 포화율(%): 정규화 제어 명령의 경계 근접 점유율로 정의하며, $|u| \geq 0.99$ (허용범위 ±1%) 구간의 시간 비율. (3) Attitude RMS/95th(°): 명령 자세 대비 실제 자세 오차의 RMS 및 95 번째 백분위값. (4) Steady yaw rate(rad/s): 분석 구간 후반(90–120 s)의평균 yaw 속도. (5) cp/cr/cy 명령(μs): 믹서 입력 기준의 pitch, roll, yaw 축 명령 신호(1500 μs 중심값). (6) 지연 및 지터(ms): 센서 수집부터 액추에이터 명령 송신까지의 엔드투엔드 지연 및 제어주기 지터. 모든 지표는 워밍업 구간 이후 동일한 분석 윈도우 내에서 계산하였으며, 결과는 20 회 반복 실험의 평균과 표준편차로 보고하였다.

4.6. 정량 결과(로그)

내부 로그의 초기 10 개 스냅샷(호버 구간)에서 요약 통계는 다음과 같다: yaw-rate ≈ 0.00 rad/s, cp ≈ 0.77 μs, cr ≈ −0.25 μs. 추정치(est)와 GT 의 오일러 각은 ~ 0.0–0.1° 범위로 일치한다.

Table 1 시나리오별 정량 지표 요약

지표	S1 Hover	S2 Gyro Bias	S3 Dyn Acc	S4 Agg Tilt
p-p(μs)	46	25	26	3
sat(%)	0	0	0	0
RMS(deg)	1.227	2.040	1.748	0.440
P95(deg)	0.724	0.636	1.515	0.734

통신 경로 비교: RPC vs Shared Memory

측정 프로토콜 CAP=1 조건에서 동일 PC/월드/Δt 로 RPC 경로와 Shared Memory(SPSC, zero-copy) 경로를 비교하였다. 각 경로에 대해 IMU, Barometer, Magnetometer, PWM 의 타임스탬프(ts_ns)로 인접 샘플 간 Δt 역수를 계산하였다. 그런 다음 60–120 s 구간에서 유효 주파수와 지연/지터(P50/P95), 드롭/오버라이트 비율을 구하였다. IMU 는 1000 Hz 에서 시작해 설정 가능한 상한까지 가속(stress 테스트)했고, Barometer/Magnetometer 는 100 Hz, PWM 은 400 Hz 를 목표로 하였으며, 모든 실험은 동일 PC/월드/Δt, 상향된 스레드 우선순위, Windows 타이머 분해능 1 ms, 고정된 패킷 크기 조건에서 수행하였다.

최대속도 및 센서 대역폭 해석

본 비교는 각 경로의 최대 처리 주파수를 보기 위한 최대속도(stress) 테스트이다. 최대 지속 처리율이 클수록 실제 운용 주파수(IMU ~ 1 kHz, PWM 400 Hz)에서 드롭/오버라이트 없이 균일한 샘플링을 유지할 여유가 커진다. ICM42688P 급 IMU 는 데이터시트상 최대 ODR 이 ~ 32 kHz 로 내부 오버샘플링 후 필터링·다운샘플링을 수행하므로, 시뮬레이터가 수 kHz–수만 Hz 급 샘플링을 처리할 여유 대역폭을 갖는 것이 현실 센서 동작을 모사하는 데 유리하다. 제안된 구현에서 IMU 퍼블리셔는 최대 ~ 32 kHz 까지 설정 가능하며, Shared Memory 경로에서는 이 범위 내에서 IMU·PWM 루프가 동시에 유지된 반면 RPC 경로에서는 IMU 속도를 올릴수록 PWM 루프가 목표 주파수(400 Hz)에서 점차 이탈하였다. RPC 는 직렬화/컨텍스트 스위칭/네트워킹 오버헤드로 인해 PWM 루프에서 목표 주파수에 안정적으로 도달하기 어렵고, Shared Memory 기반 SIL 제어기는 IMU 및 PWM 에서 더 높은 최대 처리 주파수와 여유 대역폭을 확보하였다.
요약하면, RPC 경로는 직렬화·컨텍스트 스위칭·네트워킹 오버헤드로 인해 목표 주파수 달성이 어려운 반면, 공유메모리 경로는 목표 주파수를 일관되게 유지하였다. 이결과는 제어기/필터 블록을 SIL 제어기에서 검증 후 실제 FC 에 소스 수준으로 신속 이식할 수 있음을 뒷받침한다.

통신 경로 비교 요약 및 로그(SIL Controller)

아래 표는 RPC 와 Shared Memory 경로의 측정 주파수를 요약한 것이다(표 2). 워밍업 10 s 를 제외한 동일 분석 윈도우에서 수집하였다.

Table 2 통신 경로 비교: RPC vs Shared Memory

항목	RPC(Hz)	SHM(Hz)
IMU	15,516	78,383
Barometer	100	100
Magnetometer	100	100

PWM	340	400

5. 결론

본 연구는 AirSim 내 공유메모리 기반 고주파 폐루프와 경량 안정화 패키지를 결합해, 시뮬레이터/개발 환경에서 안전하고 재현 가능하게 제어기를 벤치마크·튜닝할 수 있는 실용 프레임을 제시했다. 외부 프로그램(SIL controller)을 통해 IMU ~ 1 kHz, Barometer ~ 100 Hz, PWM 400 Hz 루프를 달성했고, 실제 로그로 이를 확인하였다.

• 고주파 폐루프 달성: in-process Shared Memory + SPSC 로 지연/지터를 낮추고, SIL controller 에서 IMU ~ 1 kHz, Barometer ~ 100 Hz, PWM 400 Hz 를 안정적으로 유지하였다.

• 스파이크/포화 감소, 자세 품질 개선: D-term PT2×2+ 클램프, Trim 램프-인, 추정기 게인 EMA 스무딩, EST/GT 연속 블렌딩, yaw rate-PI(anti-windup), 모터 Slew 제한의 결합으로 peak-to-peak 와 포화 점유율을 유의하게 낮추고 yaw 정지(rate→0), Attitude RMS 를 개선하였다.

• 재현 가능한 벤치마크: 워밍업·윈도우·시나리오(S1–S4)와 공통 파라미터를 고정한 절차를 제시하여, 로그·표준 지표로 비교 가능성을 확보하였다.

• 이식성: 구성 요소를 FC 구조에 맞춰 모듈화하여, 시뮬레이터에서 검증한 동일 구조를 실제 FC 로 최소 변경 이식 가능함을 보였다.

향후 과제로는 (i) 동적 노치 자동화와 추가 필터 체인 최적화, (ii) magnetometer/altitude (BARO) 통합 및 EKF 대체 비교, (iii) 다양한 부하/스케줄링 조건에서의 지연/지터 정량화, (iv) ROS2·HIL·실기체 포팅을 통한 실기체 검증, (v) 자동 튜닝/안정성 진단 도구화가 있다.

[1] ArduPilot Dev Team, "Copter PID Tuning and Rate Controller (ATC_RAT_), Filter/Notch Docs," [On-line]. Available: https://ardupilot.org/copter/docs/tuning.html [Accessed:2025-11-29].

[2] PX4 Dev Team, "Multicopter PID Tuning Guide," Online. Available: https://docs.px4.io/main/en/config_mc/pid_tuning_guide_multicopter.html (Accessed: 2025-11-29).

[3] K. J. Åström and T. Hägglund, *Advanced PID Control*, ISA, 2006.

[4] L. Zaccarian and A. R. Teel, *Modern Anti-windup Synthesis*, Princeton University Press, 2011.

[5] ArduPilot Dev Team, "IMU Notch / Dynamic Notch Filtering," Online. Available: https://ardupilot.org/copter/docs/common-imu-notch-filtering.html (Accessed: 2025-11-29).

[6] ArduPilot Dev Team, "IMU Filtering Reference (INS_GYRO_FILTER, PID D-term filtering)," Online. Available: https://ardupilot.org/copter/docs/common-imu-filtering.html (Accessed: 2025-11-29).

[7] R. Mahony, T. Hamel, and J.-M. Pflimlin, "Nonlinear Complementary Filters on the Special Orthogonal Group," *IEEE Transactions on Automatic Control*, vol. 53, no. 5, pp. 1203–1218, 2008.

[8] S. O. H. Madgwick, "An Efficient Orientation Filter for Inertial and Inertial/Magnetic Sensor Arrays," Report, x-io and University of Bristol (UK), 2010.

[9] Y. Bar-Shalom, X.-R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*, Wiley, 2001.

[10] ArduPilot Dev Team, "MOT_SLEWRATE: Motor Output Slew Rate," Online. Available:https://ardupilot.org/copter/docs/parameters.html#mot-slewrate-motor-output-slew-rate (Accessed: 2025-11-29).

[11] PX4 Dev Team, "Actuators / Output Configuration (output limiting, mixers)," Online. Available: https://docs.px4.io/main/en/config/actuators.html (Accessed: 2025-11-29).

[12] S. Shah et al., "AirSim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics*, 2017.