Carriage Stabilization — Simulation Refactor Spec

Objective

Delete `plot_time_series_cmd.py` and rebuild the simulation stack for clarity and testability. The new system simulates a motorized carriage traveling on the **outer rim of a fixed wheel**, with feedback from the FLC's `motor_cmd`. It outputs realistic IMU-like raw signals and a clean plot script.

Artifacts to (Re)Create

1. `simulation.py` — physics + IMU synthesis (no plotting, no CLI).
2. `mock_spi.py` — thin adapter exposing `imu_read()` by delegating to `simulation.py`.
3. `plot_simulation.py` — plotting only (θ, ω, motor_cmd vs time). No simulation logic, no argparse.
4. `sim_config.toml` — simulation parameters/constants.
5. `flc_config.toml` — FLC parameters only (membership functions, rule base, scaling). Keep existing content, remove simulation-related items.

Config Separation

1) `flc_config.toml` (FLC-only)

- Contents:
  - `[flc_scaling]` (e.g., `THETA_SCALE_FACTOR`, `OMEGA_SCALE_FACTOR`)
  - `[membership_functions.theta]`, `[membership_functions.omega]`
  - `[[rule_base]]` blocks (theta_coeff, omega_coeff, bias)
- Keep controller-side items that truly belong to the FLC pipeline (e.g., `TARGET_HZ` only if your controller loop timing depends on it).
- **Remove** simulation/system items (wheel radius, mass, torque/force, initial angle, IMU noise, etc.) and relocate them to `sim_config.toml`.
- Current reference file (to update accordingly).

2) `sim_config.toml` (new; simulation-only)

Suggested schema:

```
[simulation.timing]
```

```
SAMPLE_RATE_HZ = 50.0           # dt = 1 / SAMPLE_RATE_HZ
DURATION_S     = 10.0           # total simulated seconds

[simulation.initial_conditions]
THETA_INITIAL_RAD = 0.0         # initial angle (rad)
OMEGA_INITIAL_RAD_S = 0.0       # initial angular rate (rad/s)

[simulation.mechanics]
WHEEL_RADIUS_M   = 0.604        # 604 mm -> meters
CARRIAGE_MASS_KG = 0.80         # or from prior estimate; adjust as needed
MOTOR_FORCE_N    = 8.71         # max tangential force at rim for motor_cmd=±1
GRAVITY_M_S2     = 9.80665      # g

[simulation.imu_model]
GYRO_FS_RAD_S = 4.363           # gyro full-scale in rad/s (maps to ±16384 raw)
ACCEL_RAW_FS  = 16384           # raw full-scale counts for accel mapping
NOISE_SPAN    = 3               # randint(-NOISE_SPAN, +NOISE_SPAN) added to x_raw,
y_raw

[logging]
LEVEL = "INFO"
```

Simulation Model (in `simulation.py`)

- The simulator advances one fixed timestep `dt = 1 / SAMPLE_RATE_HZ`.
- Controller computes `motor_cmd ∈ [-1, +1]` each step (FLC external).
- Dynamics (simple, requested form):
  - `alpha = (MOTOR_FORCE_N * motor_cmd) / CARRIAGE_MASS_KG + GRAVITY_M_S2 * sin(theta)`
  - `omega ← omega + alpha * dt`
  - `theta ← theta + omega * dt`
- IMU synthesis:
  - `x_raw = int(ACCEL_RAW_FS * sin(theta)) + randint(-NOISE_SPAN, +NOISE_SPAN)`
  - `y_raw = int(ACCEL_RAW_FS * cos(theta)) + randint(-NOISE_SPAN, +NOISE_SPAN)`
  - `omega_norm = clamp(omega / GYRO_FS_RAD_S, -1, +1)`
  - `omega_raw = int(ACCEL_RAW_FS * omega_norm)` (reuse ±16384 scale for 16-bit alignment)
  - Clamp all raw outputs to ±16384.
- No legacy "modes" (e.g., `OMEGA_MODE`)—they're removed.

`mock_spi.py` (adapter only)

- Provide `imu_read()` returning 6 bytes `[xL,xH, yL,yH, omegaL,omegaH]`.

- Internally hold a `CarriageSimulator` instance from `simulation.py`.
- Read most-recent `motor_cmd` from a small mailbox (e.g., `set_motor_cmd(u)` setter called by the caller) before stepping the sim.
- No slope ramps, no synthetic step trig—**all kinematics come from the simulator**.
- Keep code short and obvious.

`plot_simulation.py` (plot only; no argparse)

- Imports both configs:
  - Loads **FLC** via `flc_config.toml` (for the controller object only).
  - Loads **SIM** via `sim_config.toml` (to build the simulator).
- Runs a loop for `DURATION_S * SAMPLE_RATE_HZ` steps:
  - Read last `(x_raw, y_raw, omega_raw)` from `mock_spi.imu_read()`.
  - Convert to normalized `theta_norm`, `omega_norm` exactly as your runtime does.
  - Compute `motor_cmd = flc.calculate_motor_cmd(theta_norm, omega_norm)`.
  - Call `mock_spi.set_motor_cmd(motor_cmd)` for the next step.
  - Log arrays of `t, theta, omega, motor_cmd` for plotting.
- Plot θ, ω, and motor_cmd vs time using the **same axis conventions** you had before (shared time axis, clean labels, grid, dashed zero-lines). No command-line arguments—everything comes from the two TOML files.

Migration Notes (from current `flc_config.toml`)

Move these **out** of FLC config into `sim_config.toml`:

- Wheel/drive geometry, mass/force, gravity, initial θ, IMU noise span, gyro FS (if treated as sensor model).
- Keep in FLC config: scaling factors, membership functions, and rule base (plus any FLC-only runtime constants). See current entries to relocate (e.g., `CARRIAGE_MASS`, `MOTOR_TORQUE`, `THETA_INITIAL`, `NOISE_SPAN`, possibly `GYRO_FULL_SCALE_RADS_S`).

Non-Goals / Removed

- Delete `plot_time_series_cmd.py`.
- Remove argparse usage entirely.
- Remove legacy `OMEGA_MODE`, slope ramps, base offsets, and other ad-hoc synthesis.

Acceptance Criteria

- Clean module boundaries:
    - Physics in `simulation.py`
    - Device adapter in `mock_spi.py`
    - Visualization in `plot_simulation.py`
- End-to-end closed loop runs with **only** the two TOML files.
- Plots show responsive negative feedback (θ and ω stabilize/track reasonable behavior when FLC scaling is sane).
- Code is concise, readable, and well-commented; no dead paths or legacy flags.
