Next Steps: Mitigating Drift Across Weights & Implementing More Stable Data Storage

1. Background

HX711-based scale shows drift and hysteresis across different weight ranges. Additionally, there's a need to log and retrieve weight data without relying on an SD card.

2. Drift Mitigation Strategies

2.1. Enhanced Calibration

- **Multi-Point Calibration**: Expand beyond 3 points to cover the full measurement range (e.g., 0 kg, 1 kg, 5 kg, 10 kg).
- **Polynomial Fit**: If nonlinearity persists, perform a 2nd-order fit ($y = ax^2 + bx + c$) and store coefficients.

2.2. Temperature Compensation

- **On-board Temperature Sensor**: Read ambient temperature and apply a lookup table or linear correction.
- **Periodic Zero Offset Update**: Re-tare automatically at temperature thresholds.

2.3. Mechanical and Electrical Isolation

- **Anti-Vibration Mount**: Use dampers or rubber feet to isolate the load cell.
- **Shielded Wiring & Grounding**: Ensure all load cell leads are shielded and properly grounded.

2.4. Software Filtering & Hysteresis Correction

- **Adaptive Moving Average**: Increase window size at higher weights or when rapid change is unlikely.
- **Hysteresis Compensation**: Implement a small deadband: ignore fluctuations <0.02 kg once weight stabilizes.

**3. Data Storage Solutions

3.1. Volatile Buffering (SRAM)

- **Ring Buffer in SRAM**: Maintain a circular array of the most recent N readings (e.g., 100 entries) while powered.
- **Serial Dump**: Provide a command to output buffered data over USB serial for PC capture or logging.

3.2. SD Card Module (Nonvolatile)

- **SPI-Based microSD Interface**: Use a microSD breakout (FAT32 formatted) connected via SPI.

- **SdFat or SD Library**: Initialize the card in `setup()`, create or open a CSV file (e.g., `readings.csv`).
- **Buffered Writes**: Accumulate blocks of readings in SRAM (e.g., 16 or 32 entries) before appending to the file to minimize card wear.
- **Data Format**: Append timestamped entries in CSV or binary format: `timestamp_ms,weight_kg`.
- **File Safety**: After each block write, call `file.flush()` to safeguard against data loss on power fail.

3.3. External FRAM Module

- **I²C FRAM (e.g., MB85RC256V)**: Offers virtually unlimited write cycles (~10^{14}) and ~32 KB capacity.
- **Integration**: Use `Wire` and FRAM library; map FRAM addresses to timestamped readings for quick random access.

3.4. Wireless/Data Extraction

- **Bluetooth LE Module**: Stream stored data from either SRAM buffer or SD file to a mobile app for logging and analysis.
- **WiFi with MQTT**: Periodically publish readings or file chunks to a local broker for remote monitoring and dashboard integration.

4. Implementation Plan Wireless/Data Extraction**

- **Bluetooth LE Module**: Stream stored data to a mobile app for logging and analysis.
- **WiFi with MQTT**: Post readings to a local broker periodically.

4. Implementation Plan

Phase	Task	Owner	Timeline
1	Expand calibration points & polynomial fitting	Dev Team	1 week
2	Integrate temperature compensation & periodic tare	Dev Team	1 week
3	Add adaptive moving average and hysteresis deadband	Dev Team	3 days
4	Prototype SRAM ring-buffer & serial dump	Dev Team	3 days
5	Evaluate EEPROM wear-leveling vs. FRAM module	Dev Team	1 week
6	Add wireless extraction (optional)	Dev Team	1 week

5. Verification & Testing

- **Drift Characterization**: Measure stability at 0 kg, 1 kg, 5 kg, 10 kg over 24 hrs.
- **Data Integrity**: Write/read cycle tests of EEPROM/FRAM to >10,000 writes.
- **Field Validation**: Deploy prototype in end-use scenario to confirm reliability.

6. Documentation & Maintenance

- **User Guide**: Describe calibration routine, data dump commands, and wireless setup.
- **Code Comments**: Annotate filter parameters, calibration coefficients, and storage routines.
- **Future Enhancements**: Add real-time clock for timestamped logging; web dashboard for data visualization.