iPhone–RoboMaster S1 Sensor Fusion on Raspberry Pi:
Real-Life Pipeline and Experiments

Tuna Akyürek

28.08.2025

Abstract

This report documents an iPhone-driven sensor fusion pipeline for the RoboMaster S1 running on a Raspberry Pi. Two Extended Kalman Filters (EKFs) are used in deployment: a full 9-DOF EKF (3D position/velocity/attitude) and a planar 8-DOF EKF specialized for ground operation with yaw observability remedies. The iPhone streams IMU, GPS, and magnetometer data to the Pi for real-time estimation; the system logs for offline analysis. Implementation details, calibration, runtime behavior (50 Hz), and practical guidance are presented. **All mathematical definitions and EKF equations are centralized in `Total_Formulary.tex`; this report omits duplicated formulas.**

Contents

1 Quick Start (Demo-Friendly)

1. Mount the iPhone securely on the RoboMaster S1; power the Raspberry Pi.
2. On the iPhone app, set streaming to the Pi's IP on UDP/TCP port `5555`; enable accelerometer, gyroscope, magnetometer, GPS (and barometer if available).
3. Optional: hold the robot stationary for 5 s to 10 s to auto-estimate IMU biases.

4. Start one of:
   - **8-DOF (ground)**: `pythoniphone_integration/pi_phone_connection/main_integration_robomaster.py`
   - **9-DOF (full 3D)**: `pythoniphone_integration/pi_phone_connection/main_integration_enhanced.py`
5. Inspect live printouts; CSV logs are saved under `iphone_integration/data/` for analysis.

Practical tip

For first-time demos or when yaw stability is critical, prefer the **8-DOF** EKF; switch to **9-DOF** for operations involving vertical motion or roll/pitch dynamics.

2 System Overview

Hardware. iPhone (sensor source) rigidly mounted on RoboMaster S1; Raspberry Pi executes EKF and logging.

Communication. iPhone streams JSON packets via UDP/TCP to the Pi.

EKF variants (key files).

9-DOF `iphone_integration/pi_phone_connection/ekf_drone_9dof.py`

8-DOF `iphone_integration/pi_phone_connection/ekf_robomaster_8dof.py`

Integration modules.

- 9-DOF main: `main_integration_enhanced.py`
- 8-DOF main: `main_integration_robomaster.py`
- iPhone receiver: `iphone_sensor_receiver.py`

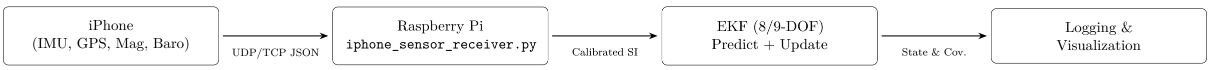

Figure 1: End-to-end pipeline on the robot.

3 Coordinate Frames & Sensor Conventions

World frame. NED (North–East–Down).

Body frame. Fixed to the phone/robot; rotation body→world uses ZYX (yaw–pitch–roll).

Sensors.

- Accelerometer measures specific force.
- Gyroscope measures angular rate.
- Magnetometer gives heading (needs calibration).
- GPS provides position (and speed/course if available).

Why it matters

These conventions let us project body accelerations into the world frame and map gravity for attitude cues. ZYX matches many mobile IMU APIs and our EKF formulary.

4 EKF Usage in Deployment (No Formulas Here)

Equations reference: see `Total_Formulary.tex` for all EKF notation, 8-DOF/9-DOF models, Jacobians, and the Joseph-form appendix.

9-DOF (full 3D)

Sensors used: GPS position, magnetometer yaw, optional barometer; gravity alignment pseudo-measurements are optional during gentle motion.

Why it matters

The 9-DOF filter captures full attitude and vertical motion for ramps, tilts, or uneven terrain.

How to use it

Tune process $\boldsymbol{Q}$ for attitude slightly higher than position; set GPS/Baro/Mag $\boldsymbol{R}$ from specs; down-weight gravity alignment during aggressive maneuvers.

8-DOF (planar with yaw remedies)

Measurements/constraints used: GPS position (and velocity if available), magnetometer yaw, GPS-course yaw when fast, NHC (lateral body velocity ≈ 0), ZUPT/ZARU when stationary.

Why it matters

Ground robots poorly observe yaw from accelerometers; combining heading updates with NHC/ZUPT/ZARU couples θ with (v_x, v_y) and bounds yaw uncertainty.

How to use it

Increase q_{b_ω} for bias learning; gate GPS-course yaw by speed; gate NHC/ZUPT by IMU thresholds to avoid misuse in motion.

5 iPhone Data Ingestion & Processing

`iphone_sensor_receiver.py`:

- Reassembles JSON via UDP/TCP; parses Core Motion fields.
- Converts to SI units; derives GPS velocity when available.
- Time-stamped, calibrated messages to the EKF loop ($\approx$50 Hz); optional stationary bias calibration.

Practical tip

Run the stationary calibration right after startup to capture true mounting biases.

6 Integration Flow on the Pi

Start-up.

1. Optional pre-delay; run auto-calibration while stationary.
2. Lock GPS local reference on first valid fix.
3. Start EKF loop at ~50 Hz.

Per-cycle.

1. Predict with elapsed dt.
2. Apply available updates (as listed above per EKF variant).
3. Normalize angles; publish/log state and raw sensors to CSV.

7 Calibration & Parameters (What to Tune)

Auto-calibration. Estimate accel/gyro biases from 5 s to 10 s of stationary data.

Typical knobs.

- **Process noise** ($\boldsymbol{Q}$): accel/gyro driving noises; bias random walks.
- **Measurement noise** ($\boldsymbol{R}$): GPS pos(/vel), yaw (mag/GPS-course), NHC, ZUPT, ZARU; baro for 9-DOF if used.
- **Gates & thresholds**: speed for GPS-course yaw; IMU thresholds for NHC/ZUPT/ZARU activation.

Practical tip

If yaw drifts: (i) raise q_{b_ω}, (ii) enable GPS-course yaw at higher speeds, (iii) strengthen NHC/ZUPT weights, (iv) verify mag calibration and down-weight near interference.

8 Performance & Results (Observed)

- Raspberry Pi 4 sustains ~50 Hz EKF rate.
- 8-DOF runs lighter; 9-DOF covers full 3D.
- With constraints & heading updates active, 8-DOF yaw drift reduces markedly; yaw covariance remains bounded.

9 Safety & Troubleshooting

- Test in controlled spaces; cap speeds and use gradual commands.
- No data? Check phone app IP/port, same network, firewall, JSON schema.
- Poor estimates? Re-run stationary calibration; ensure rigid mount; retune $\boldsymbol{Q}/\boldsymbol{R}$; confirm magnetometer calibration.

10 File Map (For the Reader)

- `iphone_integration/pi_phone_connection/ekf_drone_9dof.py`

- `iphone_integration/pi_phone_connection/ekf_robomaster_8dof.py`
- `iphone_integration/pi_phone_connection/main_integration_enhanced.py`
- `iphone_integration/pi_phone_connection/main_integration_robomaster.py`
- `iphone_integration/pi_phone_connection/iphone_sensor_receiver.py`
- Logs: `iphone_integration/data/`

11 Appendix: Practical Tuning Recipe

1. Start with datasheet noises for $\boldsymbol{R}$ (GPS pos/vel, mag yaw, baro).
2. Raise q_{b_ω} until yaw bias converges quickly after ZUPT/ZARU.
3. Enable GPS-course yaw above 0.5 m/s; keep mag for low-speed continuity.
4. Adjust NHC/ZUPT weights to suppress lateral slip and standstill drift without fighting real motion.
5. In 9-DOF, down-weight gravity pseudo-measurements during high dynamics.

References (Real-Life report)

[1] S. Särkkä and L. Svensson, *Bayesian Filtering and Smoothing*, 2nd ed. Cambridge University Press, 2023.

[2] S. J. Julier and J. K. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2004.

[3] A. Barrau and S. Bonnabel, "The invariant extended kalman filter as a stable observer," *IEEE Transactions on Automatic Control*, vol. 62, no. 4, pp. 1797–1812, 2017.

[4] P. D. Groves, *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*, 2nd ed. Artech House, 2013.

[6] R. Mahony, T. Hamel, and J.-M. Pflimlin, "Nonlinear complementary filters on SO(3)," *IEEE Transactions on Automatic Control*, vol. 53, no. 5, pp. 1203–1218, 2008.

[8] M. W. Mueller, M. Hamer, and R. D'Andrea, "Fusing UWB ranges with IMU/gyros for quadrocopter state estimation," in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2015, pp. 1730–1736.

[9] T. Moore and D. Stouch, "A generalized ekf implementation for ros (robot_localization)," in *Proc. Intelligent Autonomous Systems 13 (IAS-13)*, 2016, pp. 335–348.