1. Decide the overall structure before touching code

Your book is already logically structured for you:

- Ch2: Coord systems & attitude
- Ch3: Estimators & filters
- Ch4: RF point positioning (TOA/TDOA/AOA)
- Ch5: Fingerprinting
- Ch6: Inertial / PDR / environmental sensors
- Ch7: SLAM (LiDAR & visual)
- Ch8: Practical sensor fusion
- Ch9: Advanced / crowdsourcing / collaborative / AI PDR / RIS, etc.

Hsu IPIN book pdf

So mirror that in code, but with **shared core modules**:

```
ipin-examples/
  core/          # reusable math & models used across chapters
    coords/      # frames, ENU/NED/LLH, rotations, quaternions
    estimators/  # LS, robust LS, KF/EKF/UKF, PF, FGO wrappers
    rf/          # generic RF models (RSS, TOA/TDOA/AOA)
    sensors/     # IMU, wheel odom, mag, barometer models
    sim/         # generic simulators, trajectory, noise, maps
    eval/        # metrics, error stats, DOP, etc.

  ch2_coords/
  ch3_estimators/
  ch4_rf_point_positioning/
  ch5_fingerprinting/
  ch6_dead_reckoning/
  ch7_slam/
  ch8_sensor_fusion/
```

```
 ch9_advanced/


 data/
   sim/
   real/      # optional, small demo logs


 notebooks/    # Jupyter examples per chapter
 docs/
```

Rule: anything that is *conceptually reused* by more than one chapter goes in core/. Each chX_... folder only contains:

- very thin wiring code,
- example scripts / notebooks,
- chapter-specific plots.

That stops you rewriting the same Kalman filter, coordinate conversions, or RSS path-loss 10 times.

Hsu IPIN book pdf

---

2. Break work into three big phases (for yourself + Cursor)

Instead of “all 9 chapters”, I’d group into phases:

Phase 1 – Infrastructure + Chapters 2–3

Goal: stable **math & estimation core** plus minimal examples.

- core/coords from Ch2:
  - ENU/NED/body/map frames.
  - LLH↔ECEF↔ENU transforms.
  - Quaternion / rotation matrix / Euler conversions.

Hsu IPIN book pdf

- core/estimators from Ch3:
  - Linear & nonlinear LS (Gauss–Newton, LM).
  - A small robust M-estimator wrapper (Huber, Cauchy).

    - KF, EKF, UKF skeletons.
    - A tiny particle filter class.
    - A minimal FGO wrapper (e.g. Ceres/GTSAM optional hooks, or just your own LM solver with factor abstraction).

Hsu IPIN book pdf

- ch2_coords:
    - Scripts that:
        - build 2–3 example coordinate systems,
        - print & plot transformations (like Fig. 2.4 / 2.5 style cases).
- ch3_estimators:
    - Examples:
        - 1D LS & robust LS outlier demo.
        - GNSS-like toy positioning example: KF vs EKF vs UKF vs PF vs FGO.

Phase 2 – RF positioning & fingerprinting (Ch4–5)

Goal: **radio models + 2 styles of indoor systems**.

- core/rf from Ch4–5:
    - TOA / two-way TOA / TDOA / AOA measurement models.
    - RSS path-loss model, including log-normal shadowing.

Hsu IPIN book pdf

    - DOP computation using core/coords + estimators (LS/WLS).

Hsu IPIN book pdf

- ch4_rf_point_positioning:
    - Simulator: beacons in a 2D floor, generate noisy TOA/TDOA/AOA.
    - Example scripts:
        - TOA LS / WLS trilateration.
        - TDOA with Chan & Fang algorithms.
        - AOA examples (OVEs / PLE) with nice plots.
- ch5_fingerprinting:

    - A simple RSS fingerprint DB generator for a grid map.

Hsu IPIN book pdf

    - Deterministic: k-NN, weighted k-NN.
    - Probabilistic: Naive Bayes / simple Gaussian likelihood model.
    - Maybe 1 deep model (small MLP) as “modern method”.

Phase 3 – Sensors, SLAM, fusion, advanced (Ch6–9)

Goal: **“systems-level” examples** building on core.

- core/sensors from Ch6: IMU, wheel odom, PDR, mag, baro error models.

Hsu IPIN book pdf

- core/sim: reusable 2D/3D trajectory & noisy sensor simulation.

Then per chapter:

- ch6_dead_reckoning:
    - Foot-mounted PDR sim + ZUPT example.
    - Vehicle odom + IMU dead-reckoning example.
- ch7_slam:
    - VERY light-weight 2D LiDAR or range-bearing SLAM (don’t re-implement LOAM).

Hsu IPIN book pdf

    - One factor-graph SLAM toy example.

- ch8_sensor_fusion:
    - A few fusion patterns: loosely vs. tightly coupled; observability demo; calibration example.

Hsu IPIN book pdf

- ch9_advanced:
    - Crowdsourcing demo: simple Wi-Fi map built from multiple agents.
    - Collaborative positioning toy example.

Hsu IPIN book pdf

You *don’t* need full production-grade systems; you need **clear, minimal examples that map 1:1 to sections of the book.**

---

3. How to split work into Cursor-friendly tasks

You already have Cursor + GitHub wired. The question is: *what is a "task"*?

Think in terms of **repeatable micro-pipeline** per topic:

For each "unit" (e.g., "Ch3 Kalman filters" or "Chapter 4 TOA trilateration"):

1. **Spec extraction task**
   Prompt Cursor:

"From chapter_3_estimators.pdf and the TOC, list the exact algorithms I need to implement as reusable functions for students (KF, EKF, UKF, PF, FGO) and propose function signatures in Python consistent with core/estimators."

This forces Cursor to give you a **concrete API list** before writing code.

2. **Core module skeleton task**
   Once the spec is set, another prompt:

"Create core/estimators/kalman.py with classes KalmanFilter, ExtendedKalmanFilter, UnscentedKalmanFilter matching these signatures. Include type hints, docstrings that reference the equations in Chapter 3, and TODOs where derivations are non-trivial."

3. **Unit test task**
   Immediately follow with a test task:

"Generate tests/test_kalman.py with 2–3 simple linear systems and check that the filters converge to the true state. Use very small synthetic examples."

4. **Example / notebook task**
   For the chapter example:

"Create ch3_estimators/linear_vs_ekf_example.ipynb that:

- simulates a 1D position+velocity system,
- runs LS, KF, EKF, PF, FGO,
- plots errors and prints final RMSE, in a way that mirrors the comparison in Section 3.5."

5. **Doc task**
   Finally:

"Update docs/ch3_estimators.md with a short explanation of the example, the code entry points, and how students should run it."

You can apply this exact 5-step task template to **every subtopic**:

- coords → Ch2 sections (body/map/ENU/NED/LLH).
- rf → TOA, TDOA, AOA, RSS.
- fingerprinting → deterministic vs probabilistic.
- etc.

With Skeet / sequential MCP you can chain these automatically (e.g. “for each chapter, run these 5 tasks for each subsection X, Y, Z”). But the important bit is: **keep each unit of work small enough that Cursor can see the whole file and test it.**

---

4. Suggested initial backlog (very concrete)

If you want a strict starter list:

Epic 0 – Repo + infra

1. Init repo + core/ layout, pyproject.toml, tests/, docs/.
2. Set up basic CI (pytest + minimal lint).

Epic 1 – Coordinates (Ch2)

3. core/coords implementation (all transforms in Ch2).
4. Tests: round-trip LLH↔ECEF↔ENU, ENU↔NED, body↔map.
5. Example: small script plotting transformed positions.

Epic 2 – Estimators (Ch3)

6. core/estimators LS + robust LS.
7. core/estimators KF/EKF/UKF/PF skeletons.
8. Tests for LS & KF.
9. Example notebook: estimator comparison on toy problem.

Epic 3 – RF positioning (Ch4)

10. core/rf for TOA/TDOA/AOA/RSS & DOP.
11. Simulator for beacons + noise.
12. TOA trilateration example.
13. TDOA (Chan + Fang) example.

Once those are solid, you can decide how aggressively you want to push into Ch5–9.

---

5. One last piece of "don't overdo it" advice

You're writing this for **master-level & entry-level researchers**, not for a startup turning this into a product tomorrow. You do *not* need:

- perfect abstraction from day 1,
- full-blown SLAM systems,
- every RF technology under the sun.

You need:

- **clean, readable code** that maps directly to the book's sections,
- small simulations that clearly illustrate the math,
- and enough structure that future students (or contributors) can extend it.