## 🚀 Goal

A drone that autonomously scans inventory items (via vision or tags), matches the data with weight measurements from a smart scale, and syncs this info to a backend/dashboard.

## 📌 PHASE 1: PLANNING & REQUIREMENTS

## Define Use Case

- Type of inventory: boxes, pallets, individual items?

- Environment: Warehouse? Open storage?

- Item identification: Barcode, QR, RFID, or image-based object recognition?

## Tech Stack Decision

- Drone OS: PX4, ArduPilot

- Compute: Raspberry Pi + companion computer (Jetson Nano/Xavier)

- Vision: OpenCV, YOLOv8

- Weight Communication: MQTT/HTTP API from smart scale

- Backend: Firebase, AWS IoT, or custom Node.js backend

## 🛠️ PHASE 2: HARDWARE SETUP

## 1. Drone Base

- Off-the-shelf drone (like DJI Matrice or DIY with Pixhawk)

- Add:

- LiDAR sensor (for obstacle avoidance)

- Camera (HD or thermal depending on need)

- Ultrasonic sensor (for height sensing)

- Companion Computer (Jetson Nano or Raspberry Pi)

## 2. Smart Scale

- Smart load cell-based scale

- Must have:

- Wi-Fi/Bluetooth module (ESP32 or BLE-enabled microcontroller)

- API support (to fetch weight data)

- MQTT/REST communication capability

## 💻 PHASE 3: SOFTWARE DEVELOPMENT

## 1. Item Detection (Computer Vision)

- Train a custom model (YOLOv8 or MobileNet SSD) with labeled images of inventory items

- Run object detection on the drone camera feed using OpenCV + PyTorch

- Output: Item ID, confidence, bounding box

## 2. Location Tagging (Optional but powerful)

- Use ARUCO markers or QR codes on shelves

- Drone reads marker to associate items with specific shelf locations

## 3. Drone Navigation (Autonomous)

- Use ROS + SLAM (Simultaneous Localization and Mapping)

- Program flight path via waypoints and define scanning zones

- Integrate obstacle detection and return-to-base logic

## 4. Smart Scale Communication

- Each weight scale sends:

- Item ID or shelf ID

- Weight

- Timestamp

- Scale posts to MQTT broker (e.g., Mosquitto) or REST API

## 5. Synchronization Logic

- Drone sends: Detected items + location

- Scale sends: Weight + shelf ID

- Backend matches:

{

"shelf\_id": "A2",

"detected\_items": ["SKU\_245", "SKU\_333"],

"total\_weight": "3.5 kg"

}

- Discrepancies are flagged (missing item, weight mismatch)

## 🔄 PHASE 4: INTEGRATION

## 1. Backend Server

- Node.js or Flask app

- Handles:

- Auth

- Inventory database (MongoDB or Firebase)

- API endpoints to receive drone and scale data

## 2. Dashboard UI

- React/Vue.js frontend

- Real-time updates via WebSocket

- Show:

- Inventory items with weight

- Discrepancy alerts

- Drone flight logs

## ✅ PHASE 5: TESTING & CALIBRATION

## Test Item Detection

- Start with static tests: hold an object in front of the drone

- Validate item classification and confidence levels

## Test Synchronization

- Place known-weight item on scale

- Fly drone to scan it

- Match both datasets in backend

## Full System Test

- Run full flight path

- Detect multiple items across shelves

- Compare with smart scale values

- Check alert triggers for mismatches

## 🧠 Bonus: Advanced Features

- RFID Integration: Combine vision with RFID scanning for double-check.

- Voice Command / App Control: Use a simple app to dispatch drones on command.

- Multi-Drone Coordination: Use swarm logic or area partitioning to cover large warehouses.