IOT EDGE COMPUTING-ENABLED COLLABORATIVE TRACKING SYSTEM FOR MANUFACTURING RESOURCES IN INDUSTRIAL PARK

- Zhiheng Zhao, Peng Lin,
Leidi Shen, Mengdi Zhang

Group No : 3

PROBLEM STATEMENT

Large manufacturing floors hold thousands of items, making it hard for workers to quickly find materials. Edge computing processes data locally to give instant coarse location, ensuring reliability for production. This reduces dependency on cloud, saving computation and storage while speeding up search efficiency.

Environment & Signal Simulation

- Platform: Matplotlib, NumPy, SciPy
- Create 2D grid layout with gateways & BLE tags.
- Implement RSSI generation (log-normal path loss + Gaussian noise).
- Integrate Kalman filtering (filterpy).
- Deliver a function that, given tag positions, returns filtered RSSI values.

Processing Flow

Implementation :

- **Edge Processing (Coarse Location)**: RSSI from gateways via log-distance model with noise; strongest gateway chosen. (to be improved with Kalman filter integration from person1).

- **Cloud Processing (Refined Location):** Built fingerprinting dataset; applied k-NN on tag RSSI; refined location computed by weighted averaging of neighbors.

Resource Allocation & Communication

Enhancements from Review 1 → Review 2

- **Node Definition**
  Edge & cloud with specs: CPU, memory, throughput, uplink/downlink delays,gw_id

- **Task Modeling**
  Tasks generated via material_task_source(), not just a simple generator

Scheduling Policies (Scheduler.decide)

FCFS → Always to cloud
Load Balancing (LB) →Based on Shortest estimated finish time
Latency-Aware → based on end-to-end latency
Tracks arrivals, route counts, completed tasks, latencies

Additional Features

MockBus → Simulates publish/subscribe communication
Integration → Linked with Person 2's processing flow outputs

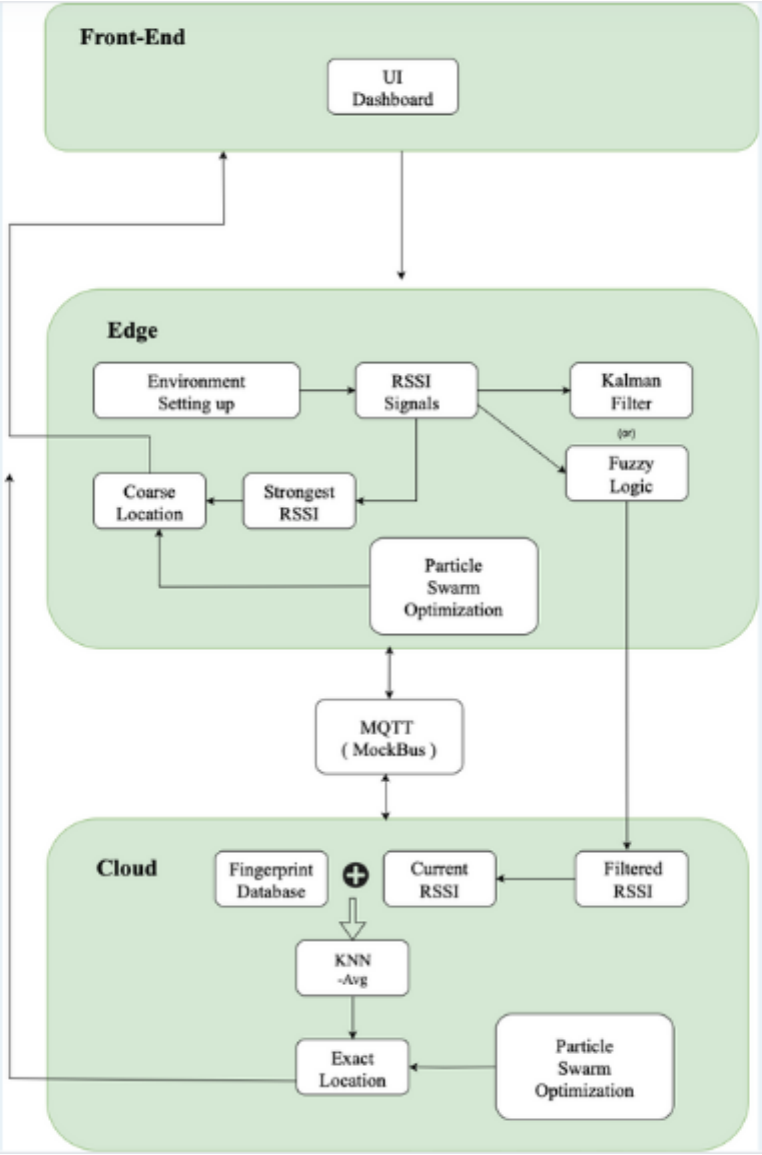
User Interaction

Implemented :

- A Login page for the workers
- Resource search page which gives instant location(dummy data as of now)
- Manual “Item found button”
- Resource location Map

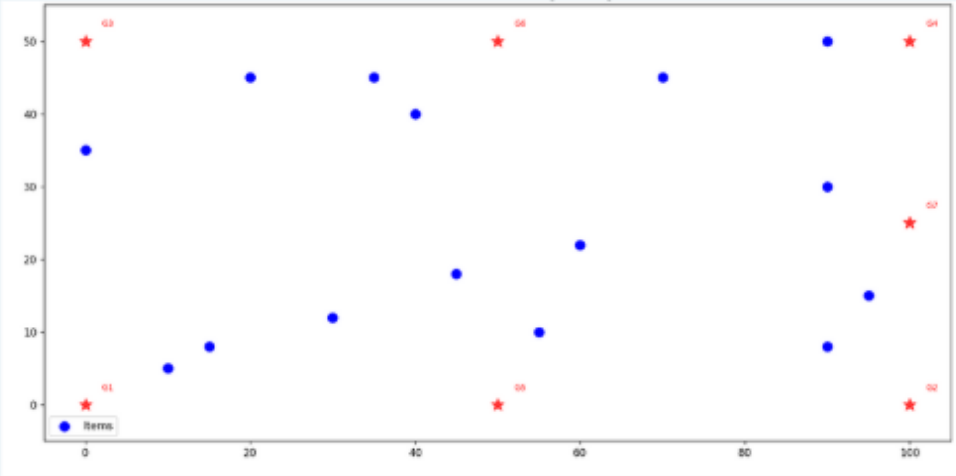
Tools Used :

FLASK,HTML,CSS,MATPLOTLIB,PYTHON


Front-End
UI
Dashboard
Edge
Environment
Setting up
RSSI
Signals
Kalman
Filter
(or)
Fuzzy
Logic
Coarse
Location
Strongest
RSSI
Particle
Swarm
Optimization
MQTT
(MockBus)
Cloud
Fingerprint
Database
Current
RSSI
Filtered
RSSI
KNN
-Avg
Exact
Location
Particle
Swarm
Optimization

CI Implementation

- Removes Kalman Filter Dependencies
- Uses **Fuzzy Signal Classification**
  - Each RSSI value gets classified into poor/avg/good categories
  - Triangular membership functions determine quality levels
- Uses **Fuzzy Rule-Based Filtering**
  - Weighted averaging based on signal quality assessment
  - Better signals get more influence in the final output
- **PSO Parameter Optimization**
  - Minimizes signal variance for smoother output
  - Adapts automatically to manufacturing environment



- This simulation is about tracking materials in a warehouse using RSSI and refining the location using **PSO (Particle Swarm Optimization).**
- Start with many random guesses (particles) for the tag's location
- For each guess, calculate predicted RSSI and compare with actual RSSI then finds error.
  - **Actual RSSI value:**

$$RSSI = P_{ref} - 10 \cdot n \cdot \log_{10}(d + \epsilon) + \text{noise}$$

  - **Predicted RSSI value:**

$$RSSI = -59 - 20 \cdot \log_{10}(distance)$$

  - **Error:**

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(Measured_i - Predicted_i)^2}$$

- Each particle remembers its best position (personal best).
- All particles also know the best position among everyone (global best).
- Particles move closer to their personal best and global best.
- Repeat steps until particles gather near the true tag location.

Fine Location Estimation