IoT edge computing-enabled collaborative tracking system for manufacturing resources in industrial park

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

GROUP - 3
CLOUD 9

Introduction

- Large indoor manufacturing environments have **thousands of items spread across multiple areas**.
- Workers **waste time searching for materials** without knowing exact location.

Issuse Faced :

  - slow search, high delay, need for instant location information

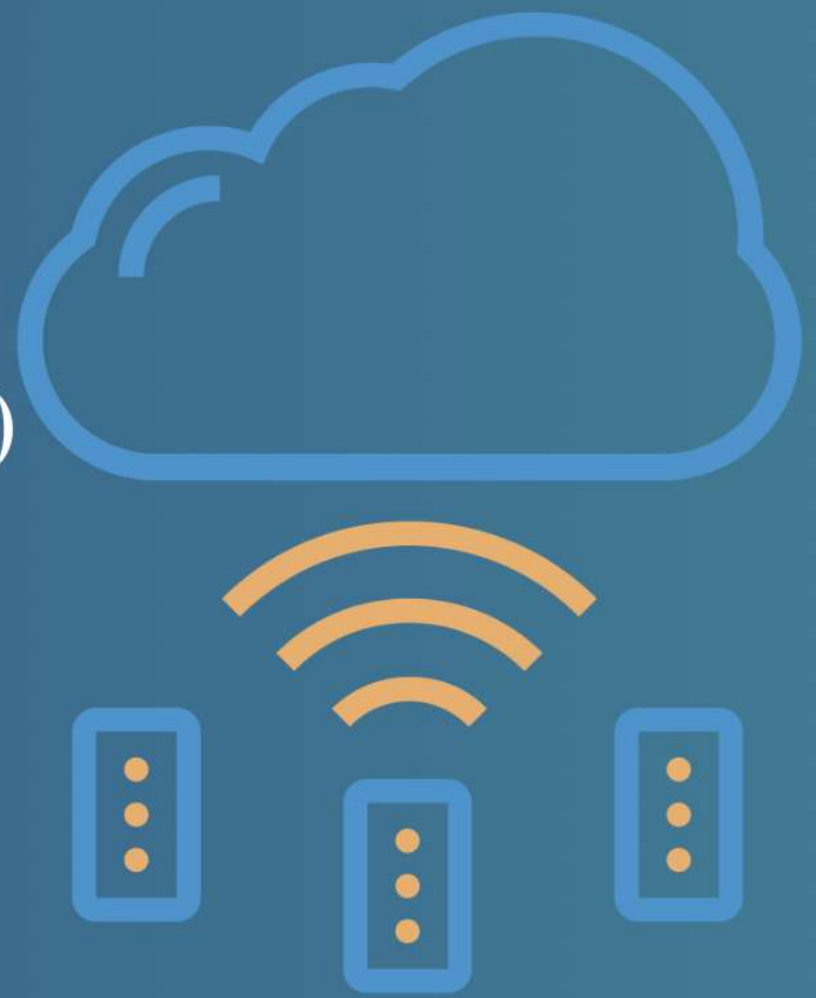
Why is Edge Computing needed here?

  - Real-time response.
  - Reducing latency and network load.
  - Immediate coarse location before cloud processing finishes.

State of Literature

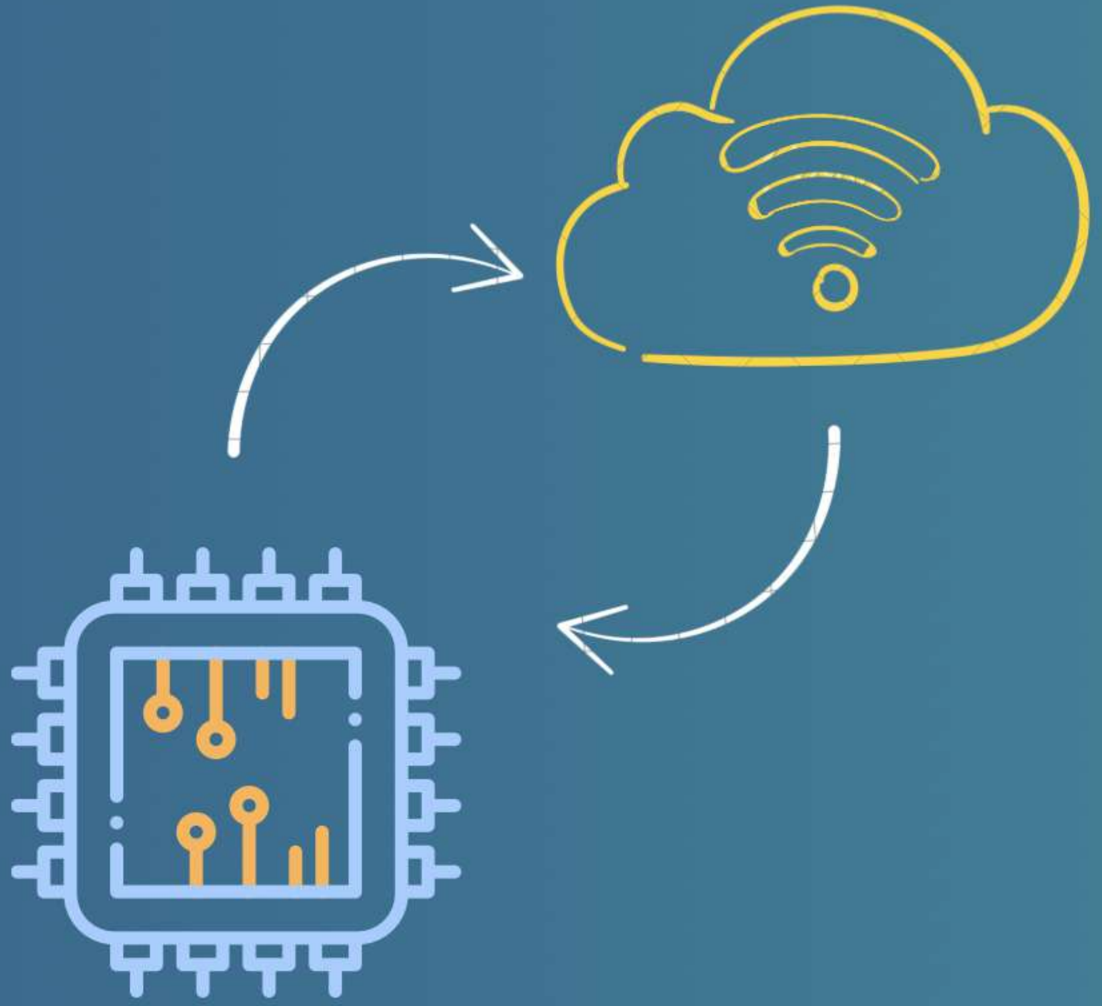
Existing Tracking Technologies in Manufacturing

1. **Bluetooth Low Energy (BLE)**
   - Low power consumption – up to 3 years on a single tag
   - Compatible with smartphones for easy integration
   - Used in equipment maintenance and asset tracking(Tei et al.)



2. **Radio-Frequency Identification (RFID)**
   - Low-cost & easy to deploy
   - Supports real-time info capture (passive & active RFID)
   - Used in warehouses & hospitals
   - Not feasible for full-scale deployment in large industrial parks

3. **Ultra-Wideband (UWB)**
    - High precision – location error in centimeters
    - Expensive – limits scalability in industrial use
4. **ZigBee**
    - Suitable for asset tracking
    - Used in predictive maintenance with Wi-Fi (Wan et al.)



Key Papers and Industrial Systems

Edge Computing in Manufacturing:

- Chen et al.:
    – Proposed an IoT-based edge computing architecture
    – Improved agility and security in manufacturing
- Wu et al.:
    – Developed a Edge Computing based real-time monitoring framework for pumps and machines
    – Utilized edge computing for process and prognosis data

- Hu et al.:
  - Designed an intelligent robot factory
  - Edge nodes (gateways & routers) reduce network congestion and latency

BLE-Based Tracking in Smart Factories

- Tei et al.:
  - Applied BLE for equipment and device maintenance
- Zhao et al.:
  - Proposed a collaborative BLE tracking method
  - Tracked finished products in a forklift manufacturing plant

What's New or Improved in This Approach

ML + GA on Edge Devices

- Introduces SLGT (Supervised Learning of Genetic Tracking)
  - Focuses on classification & accurate location estimation
  - Enhances traditional genetic algorithms (beyond optimization)
- Kalman filtering on edge gateways smooths data
- Refined signals sent to cloud for precise positioning via SLGT

Scalable Communication Architecture

- Front-End: BLE/Zigbee → transmits to nearby edge gateways
- Edge Gateways: Use NB-IoT / 4G / 5G to forward smaller, refined data
- Reduces cloud computation load
- Ensures low latency and scalability
- Real-Time Simulation & Dashboard

Three-Part Architecture:

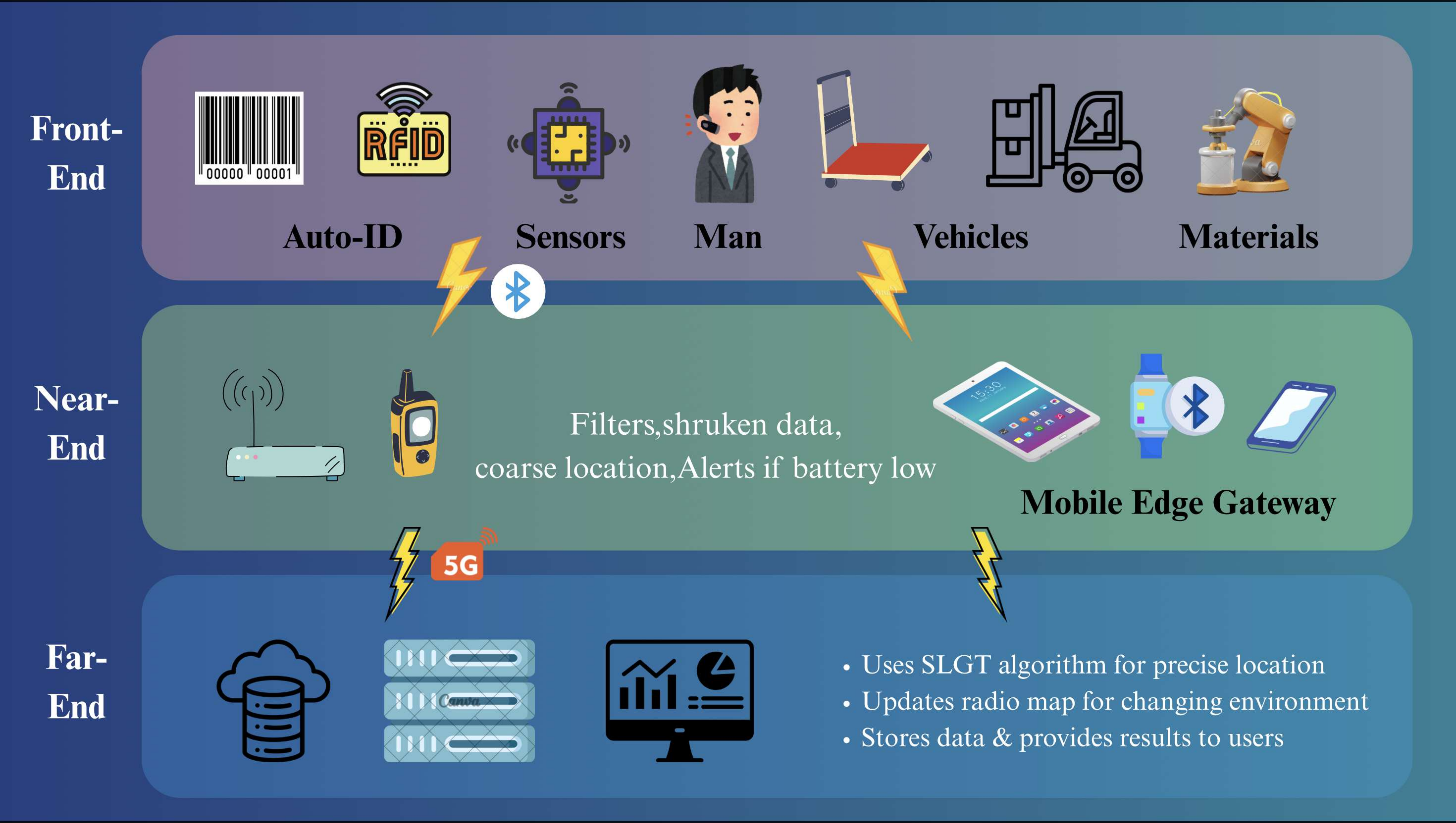
- Front-End: Smart manufacturing resources
- Near-End: Edge computing (gateways)
- Far-End: Cloud computing (dashboard & apps)

Cloud server hosts a real-time dashboard

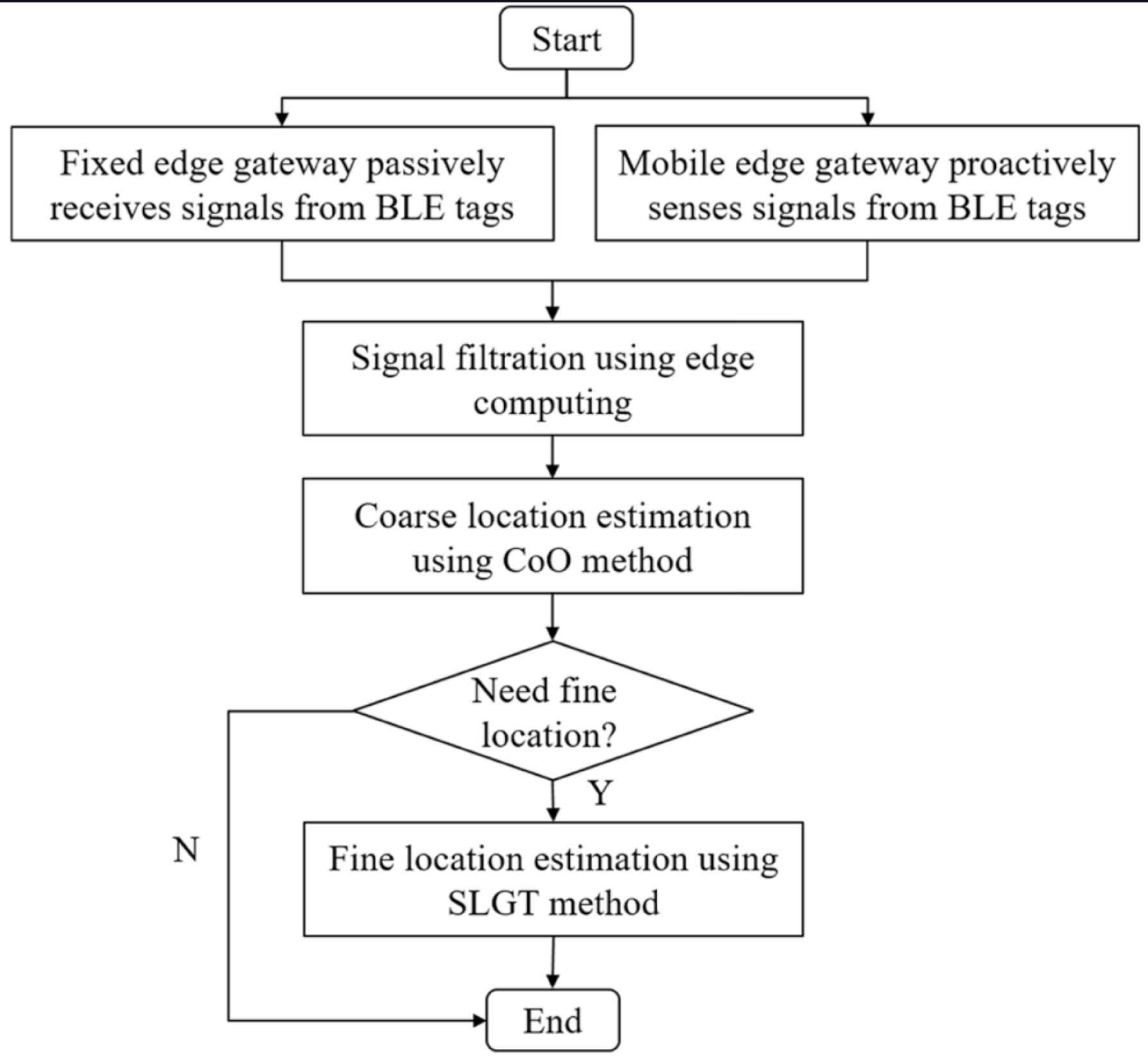
– Supports role-based access

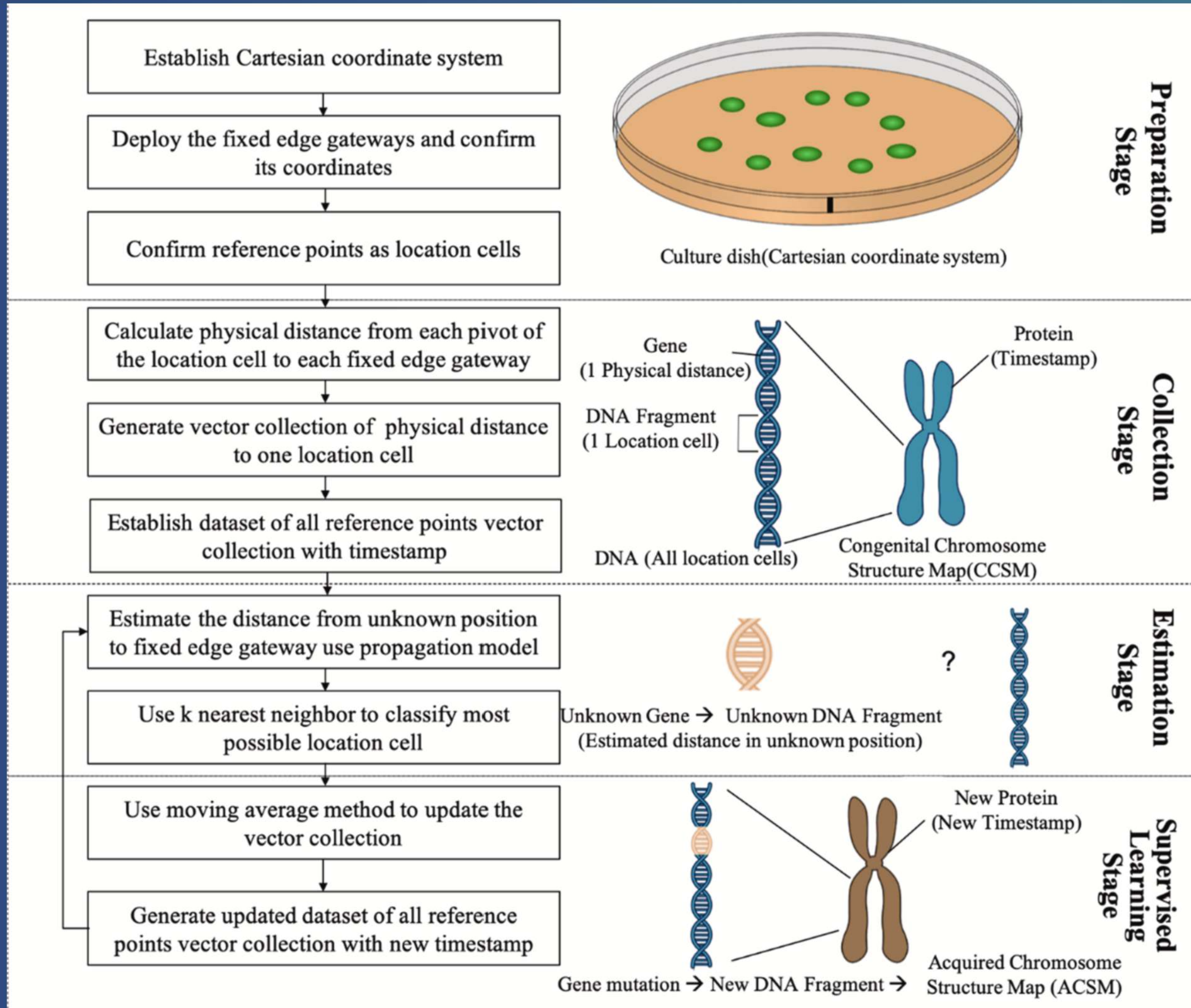
– Deployed in a real-life industrial park

Architecture



Front-End
00000 00001
RFID
Auto-ID
Sensors
Man
Vehicles
Materials
Near-End
Filters,shruken data,
coarse location,Alerts if battery low
Mobile Edge Gateway
5G
Far-End
• Uses SLGT algorithm for precise location
• Updates radio map for changing environment
• Stores data & provides results to users


Start
Fixed edge gateway passively receives signals from BLE tags
Mobile edge gateway proactively senses signals from BLE tags
Signal filtration using edge computing
Coarse location estimation using CoO method
Need fine location?
Y
N
Fine location estimation using SLGT method
End


Establish Cartesian coordinate system
Deploy the fixed edge gateways and confirm its coordinates
Confirm reference points as location cells
Culture dish(Cartesian coordinate system)
Preparation Stage
Calculate physical distance from each pivot of the location cell to each fixed edge gateway
Generate vector collection of physical distance to one location cell
Establish dataset of all reference points vector collection with timestamp
Gene
(1 Physical distance)
DNA Fragment
(1 Location cell)
DNA (All location cells)
Protein
(Timestamp)
Congenital Chromosome Structure Map(CCSM)
Collection Stage
Estimate the distance from unknown position to fixed edge gateway use propagation model
Use k nearest neighbor to classify most possible location cell
Unknown Gene → Unknown DNA Fragment
(Estimated distance in unknown position)
?
Estimation Stage
Use moving average method to update the vector collection
Generate updated dataset of all reference points vector collection with new timestamp
New Protein
(New Timestamp)
Gene mutation → New DNA Fragment →
Acquired Chromosome Structure Map (ACSM)
Supervised Learning Stage

How we gonna Implement it?

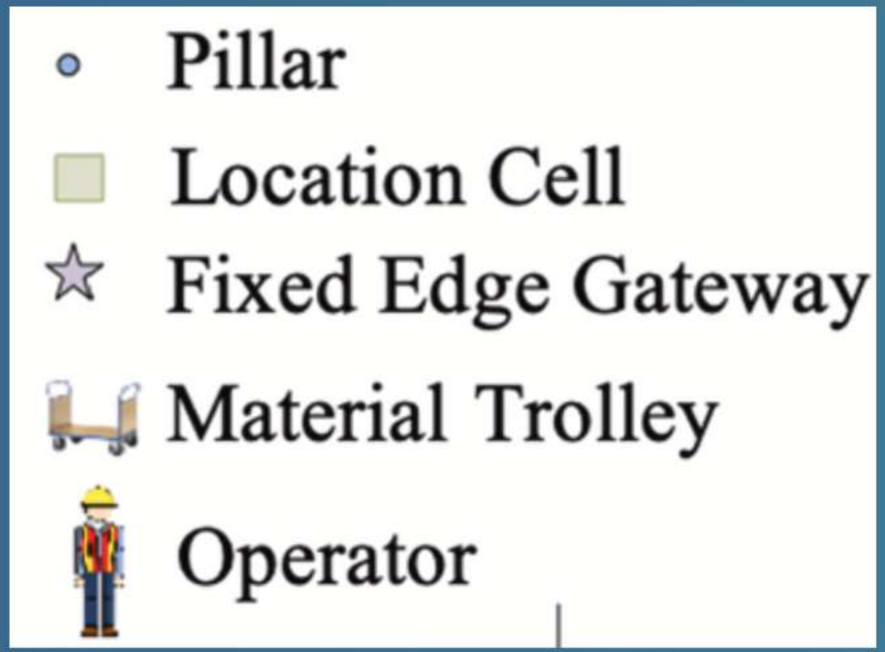
Environment Setup

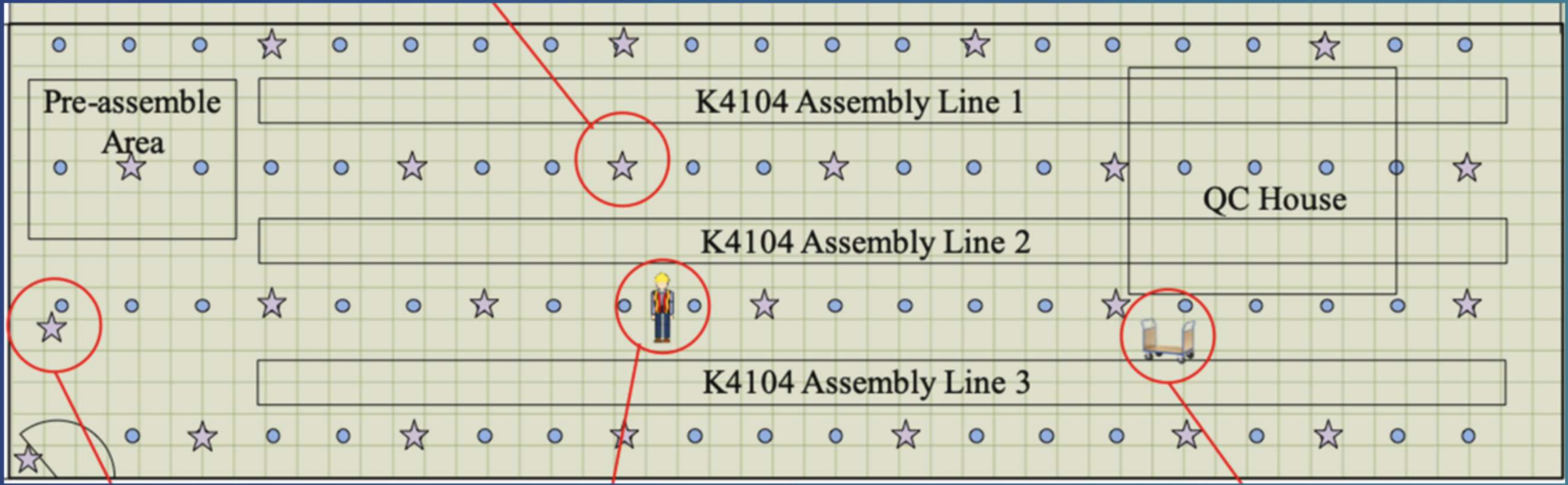
- **Platform**: Python (Matplotlib).
- Create **2D segmented grid** to represent indoor layout.
- Place **fixed edge gateways** at defined coordinates.
- **Assign virtual BLE tags** to moving materials/agents.

$$RSSI = P_{tx} - 10 \cdot n \cdot \log_{10}(d) + X_{\sigma}$$

Signal Simulation

- Platform: Python (NumPy, SciPy).
- Generate RSSI values using a **log-normal path loss model**.
- Add Gaussian noise to mimic real-world interference.
- Apply **Kalman filtering** at the edge (**filterpy library**) to stabilize signals.


Pillar
Location Cell
Fixed Edge Gateway
Material Trolley
Operator


Pre-assemble Area
K4104 Assembly Line 1
QC House
K4104 Assembly Line 2
K4104 Assembly Line 3

Processing Flow

- Edge: Runs Cell of Origin algorithm for coarse location.
- Cloud: Runs SLGT (CCSM construction, k-NN classification, weighted averaging).
- **Platform**: Cloud simulated locally in Python.

Resource Allocation

- **Platform**: Python (SimPy) - define edge and cloud nodes, each with fixed CPU and memory capacities.
- Task Modeling: Represent each computation as a task.
- **Scheduling:** FCFS, Load Balancing, and Latency-Aware Scheduling.

Communication Simulation

- **Platform**: Python (MQTT with paho-mqtt)
- Tag → Edge → Cloud
- Simulate network latency & bandwidth constraints.

User Interaction

- **Platform**: Python GUI (Tkinter) & **Web dashboard (Flask + HTML)**.
- Worker searches by entering material ID → instant coarse location from edge
- Refined location from cloud displayed after SLGT processing.
- Manual "Item Found" button for validation.

Visualization

matplotlib.animation → live plot of tags, gateways, positions and CPU load per node
Dashboard map → display positions & node load.

Task Split-up

1. **Environment & Signal Simulation -** *Geethika*
2. **Processing Flow (Algorithm) -** *Kanishka*
3. **Resource Allocation & Communication -** *Meera*
4. **User Interaction & Visualization -** *Adwaith*