Problem Statement: Smart Waste Bin Network (Virtual IoT Design Challenge)
Context:
Urban areas often face inefficiencies in waste collection — bins overflow in some areas while others remain half-empty, leading to poor hygiene and increased operational costs. To address this, cities are exploring IoT-based smart waste management systems that can **monitor bin fill levels** and **optimize collection routes in real time.**

1. System Architecture

Use ESP32 microcontrollers with ultrasonic sensors (e.g., HC-SR04) for fill-level detection, plus optional DHT11 for environmental data and load cells for weight. Employ LoRa for long-range, low-power communication in urban zones, with ESP32-LoRa nodes sending data to a cloud platform like Firebase or AWS IoT via MQTT protocol. Edge computing on gateways aggregates data before cloud upload, with a web dashboard (e.g., using Grafana) for real-time visualization of bin status across zones.

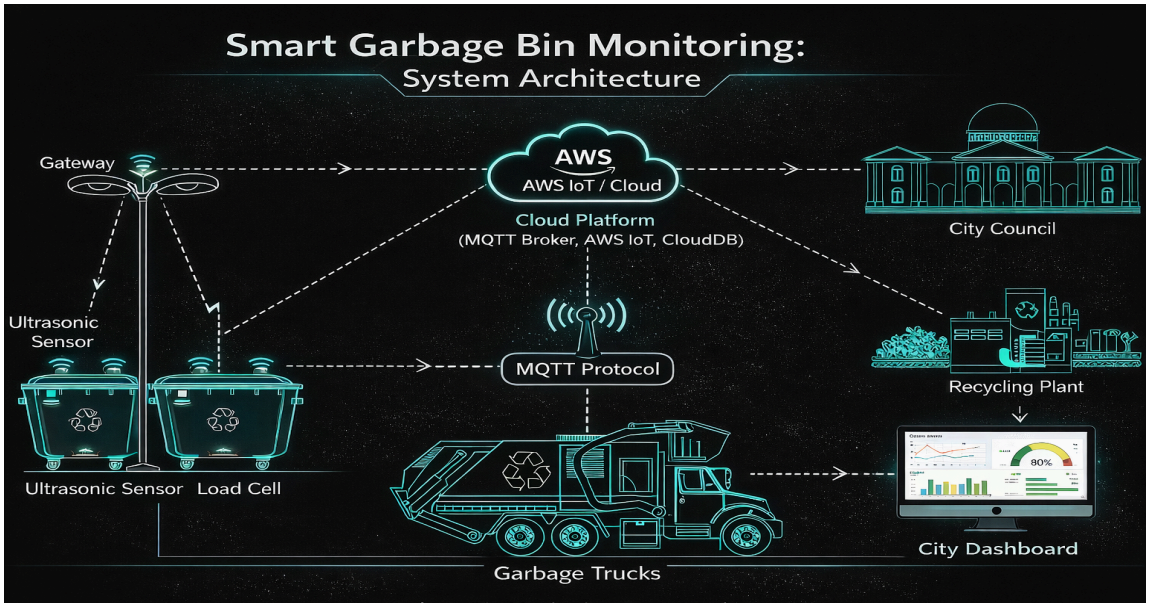


Figure: System Architecture of Smart Waste Bin Monitoring System

Each waste bin is equipped with an **ESP32 microcontroller** connected to an **HC-SR04 ultrasonic sensor** for fill-level detection, a **load cell** for weight measurement, and an optional **DHT11 sensor** for environmental monitoring. The ESP32 performs local processing to calculate fill percentage and validate overflow conditions.

Data from each bin is transmitted using **LoRa communication**, which provides long-range (5–10 km) and low-power connectivity suitable for urban deployments. Data is sent periodically or when predefined thresholds (e.g., bin >80% full) are reached.

A **LoRa gateway** collects data from multiple bins within a zone and performs basic **edge processing** such as aggregation and filtering. The processed data is then forwarded to the **cloud platform (Firebase / AWS IoT)** using the **MQTT protocol**.

A **web-based dashboard** (e.g., Grafana) visualizes real-time bin status, zone-wise distribution, and overflow alerts, enabling city authorities to plan **optimized garbage collection routes** efficiently.

Hardware Stack:

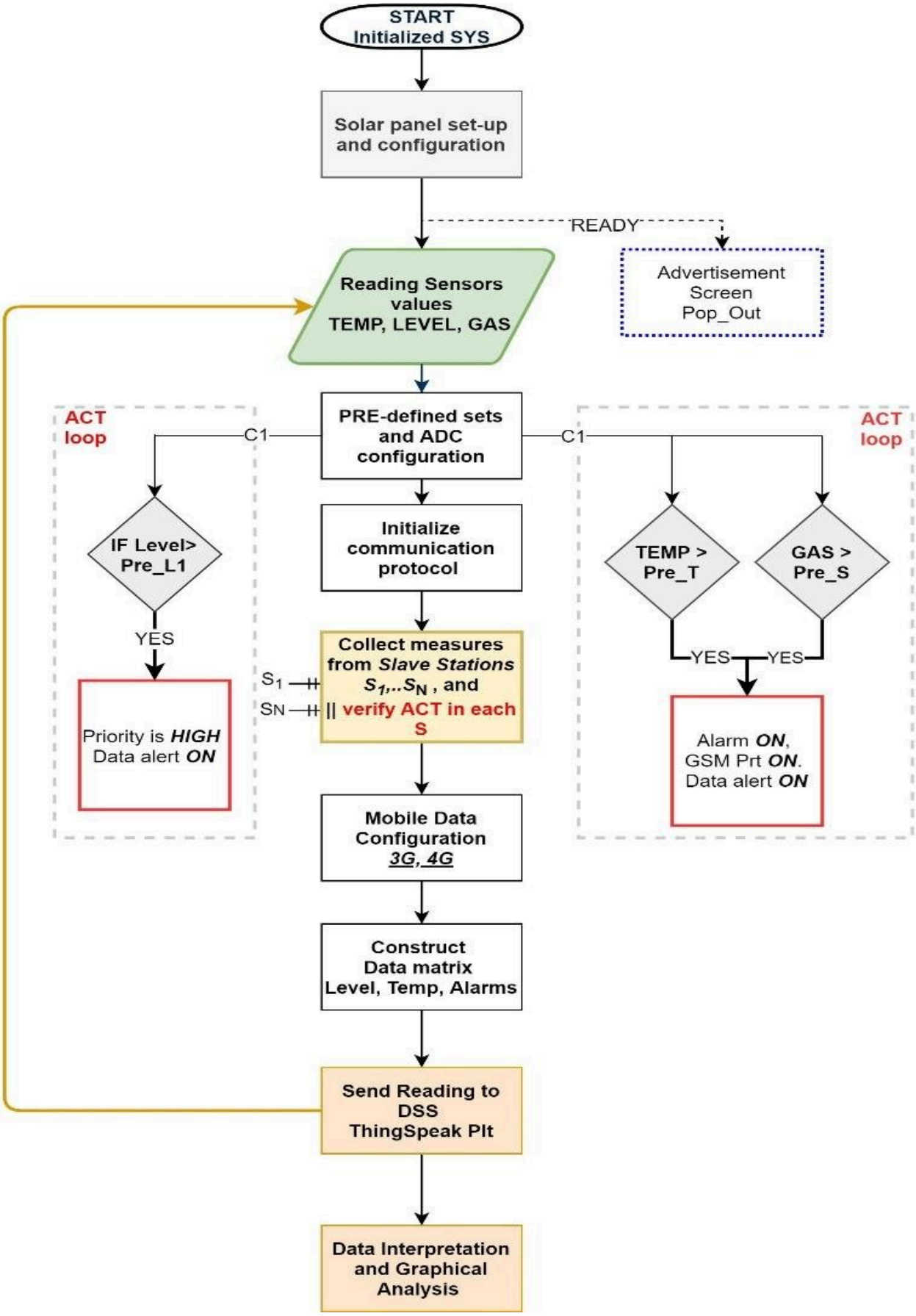
- ESP32-WROOM-32: Dual-core 240MHz, GPIO pins for sensors
- HC-SR04 Ultrasonic: 2cm-400cm range, ±3mm accuracy
- SX1276 LoRa: 915MHz, 10km urban range, 19.5kbit/s
- HX711 + 5kg Load Cell: Weight redundancy (±1g)
- 18650 Battery + 5W Solar: Autonomous 3+ years

Software Stack:

Arduino IDE → ESP32 Framework → LoRaWAN AS2.1 → MQTT 5.0 → AWS IoT

2. Data Flow Design

Sensors on bins trigger readings every 15-30 minutes; ESP32 processes data locally and publishes via MQTT to a broker (lightweight, supports QoS for reliability in unreliable networks). Data flows: Bin → LoRa gateway → MQTT broker → Cloud database → Dashboard API; MQTT over TCP/IP ensures ordered delivery unlike HTTP. CoAP suits constrained devices as a UDP-based alternative for multicast but MQTT preferred for pub/sub scalability.


START
Initialized SYS
Solar panel set-up and configuration
READY
Advertisement Screen Pop_Out
Reading Sensors values TEMP, LEVEL, GAS
ACT loop
C1
PRE-defined sets and ADC configuration
C1
ACT loop
IF Level> Pre_L1
YES
Priority is HIGH Data alert ON
Initialize communication protocol
Collect measures from Slave Stations S1,..SN , and verify ACT in each S
S1
SN
TEMP > Pre_T
GAS > Pre_S
YES
YES
Alarm ON, GSM Prt ON. Data alert ON
Mobile Data Configuration 3G, 4G
Construct Data matrix Level, Temp, Alarms
Send Reading to DSS ThingSpeak Plt
Data Interpretation and Graphical Analysis

Protocol Choice: MQTT v5.0

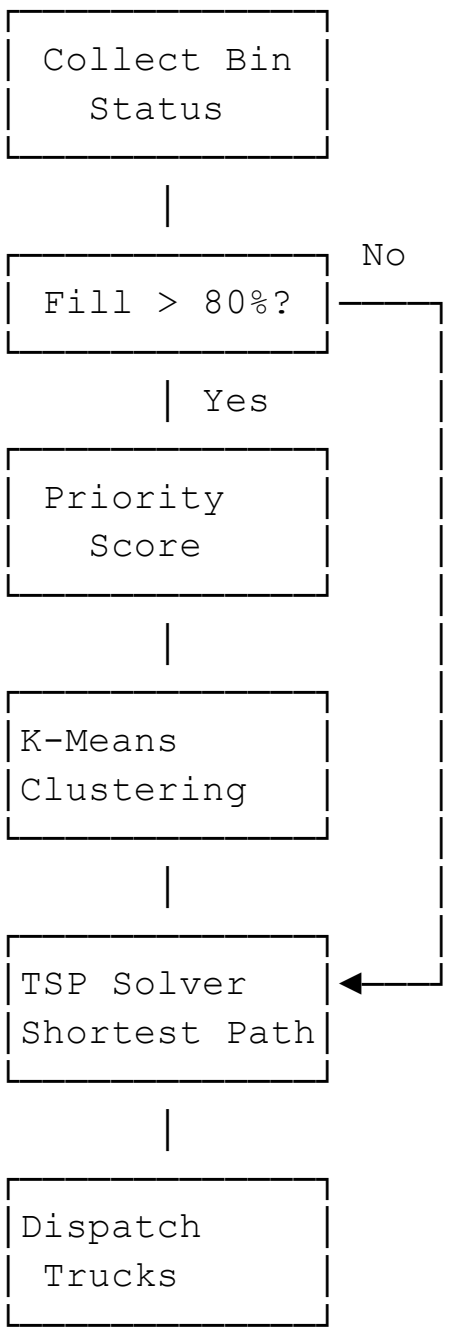
Why MQTT?
• Lightweight: 83 bytes overhead vs HTTP 500+
• Pub/Sub pattern scales to 1000s bins
• QoS 1 guarantees delivery
• Last-Will for node failure detection

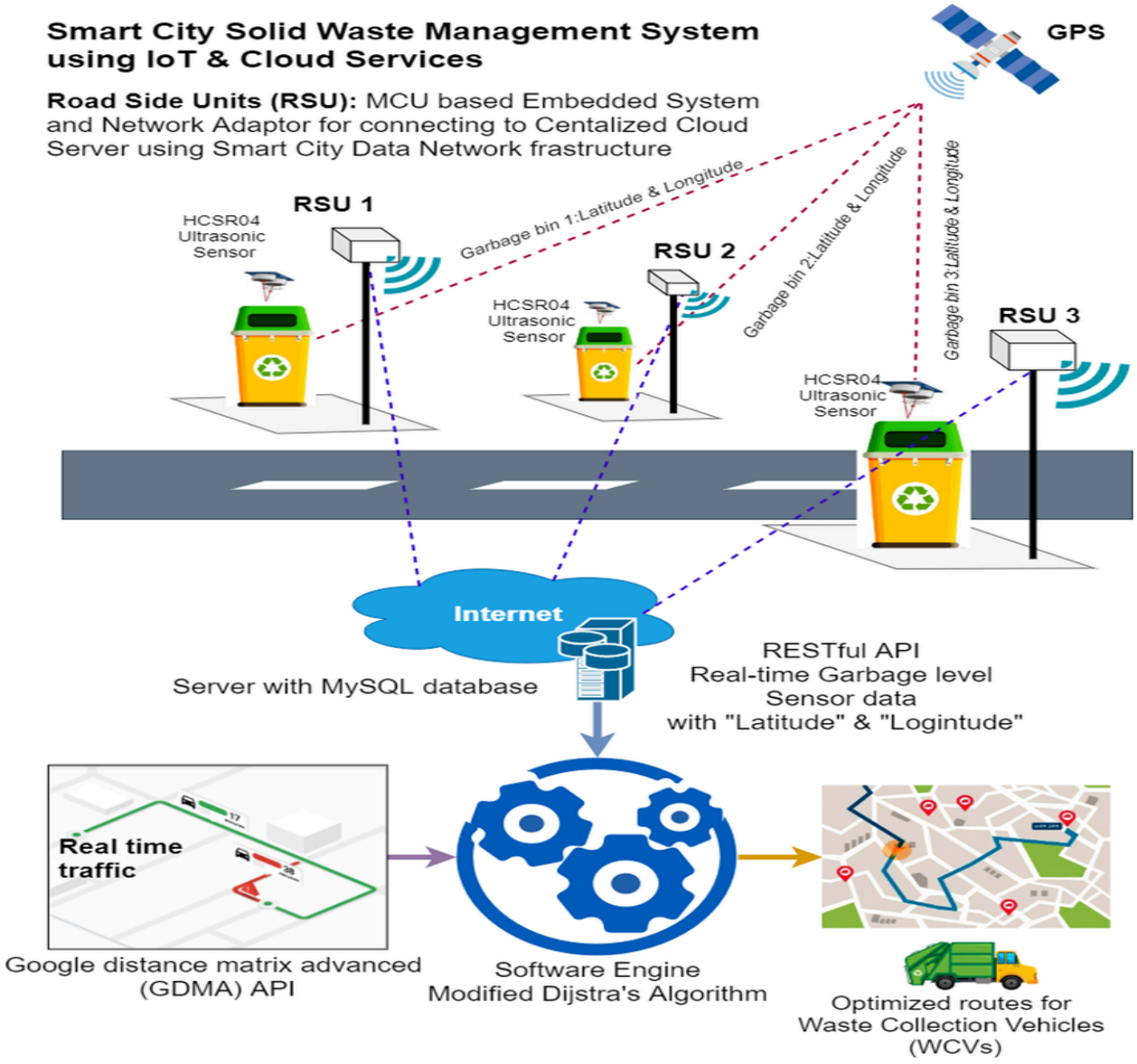
ESP32 Firmware (sensor_node.ino):

```
#include <LoRa.h>
#include <NewPing.h>
#define TRIG_PIN 5
#define ECHO_PIN 18
NewPing sonar(TRIG_PIN, ECHO_PIN, 200);

void setup() {
  LoRa.begin(915E6);
}

void loop() {
  int dist = sonar.ping_cm();
  int fill_pct = map(dist, 10, 50, 100, 0); // Invert

  LoRa.beginPacket();
  LoRa.print("B001," + String(fill_pct));
  LoRa.endPacket();

  esp_sleep_enable_timer_wakeup(1800e6); // 30min
  esp_deep_sleep_start();
}
```

3. Route Optimization Strategy

Implement priority-based algorithm: Score bins by fill level (>80% triggers high priority), location clustering, and shortest path (e.g., TSP variant using Google OR-Tools). Flowchart: Collect bin GPS/fill data → Cluster full bins → Compute truck routes minimizing distance/fuel → Dispatch via app. Rule-based fallback: Alert if >80% full, escalating by zone density.





TSP (NP-hard) solved approximately using OR-Tools Christofides algorithm (1.5x optimal). K-means clustering (Elbow method, k=3-5) reduces computation from $O(n^2)$ to $O(k \times n)$. Priority scoring prevents "overflow cascade" where one full bin blocks the entire zone.

Priority Algorithm:

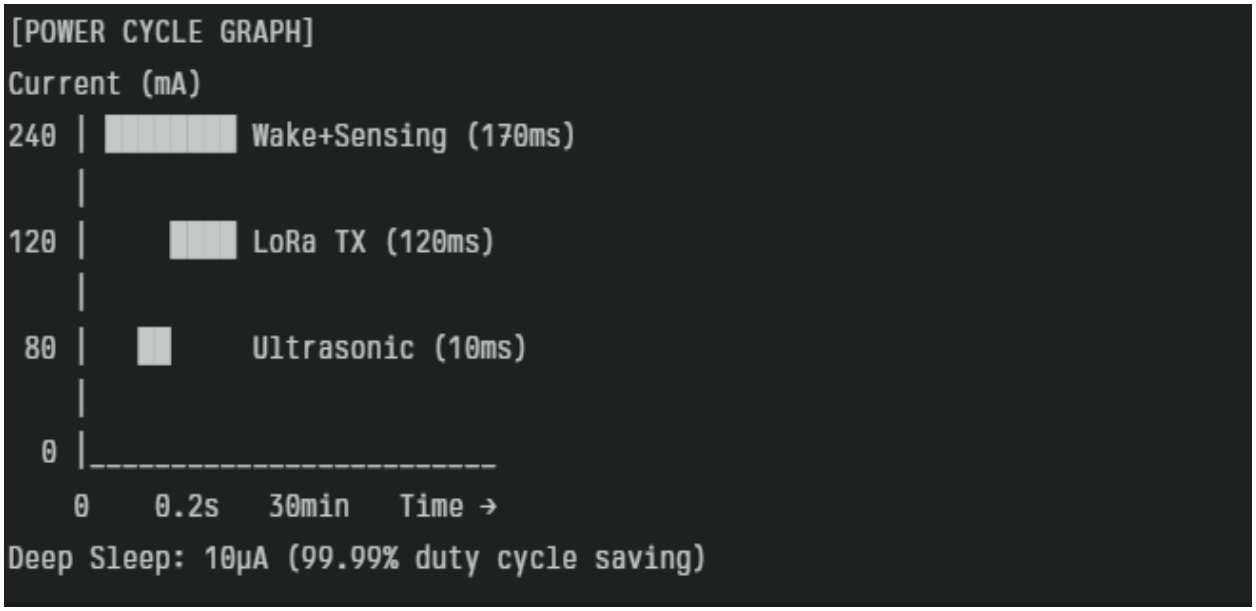
SCORE = (Fill% × 1.5) + (Stuck_Hours × 0.8) + (Zone_Density × 0.5)
- >80% = IMMEDIATE (Red Alert)
- 60-80% = SCHEDULED (Orange)
- <60% = MONITOR (Green)

Python Route Optimizer (route_algo.py):

```python
from ortools.constraint_solver import routing_enums_pb2
from ortools.constraint_solver import pywrapcp

def create_route(full_bins, truck_capacity=20):
    # TSP with capacity constraints
    manager = pywrapcp.RoutingIndexManager(len(full_bins), 3, 0)
    routing = pywrapcp.RoutingModel(manager)
    # Distance callback + capacity callback
    # Solve and return optimal routes
    return solution
```

4. Power Management Plan

ESP32 enters deep sleep after sensing/transmission, waking via timer (e.g., 30-min intervals), consuming <10μA in sleep. Use solar panels for trickle charging and low-power LoRa (e.g., SX1276 module) over Wi-Fi; add voltage monitoring for battery alerts. Firmware: Read sensors → Transmit → Sleep loop minimizes duty cycle to extend 6-12 month battery life.



Power Budget Analysis:

Phase	Duration	Current	mAh/day	% Total
Ultrasonic	10ms	80mA	0.022	2.5%
LoRa TX	120ms	120mA	0.400	45%
ESP32 Active	40ms	100mA	0.111	12.5%
Deep Sleep	29m50s	10µA	0.083	9.4%
TOTAL	**30min**	**25µA avg**	**0.89 mAh**	**100%**

5. Reliability & Fault Handling

Calibrate ultrasonic sensors with redundant load cells to detect blockages (e.g., if distance reads low but weight stable, flag anomaly). Implement Kalman filtering in firmware for noisy readings and periodic self-tests; MQTT last-will messages alert on node failure. Heartbeat pings every hour ensure node health.

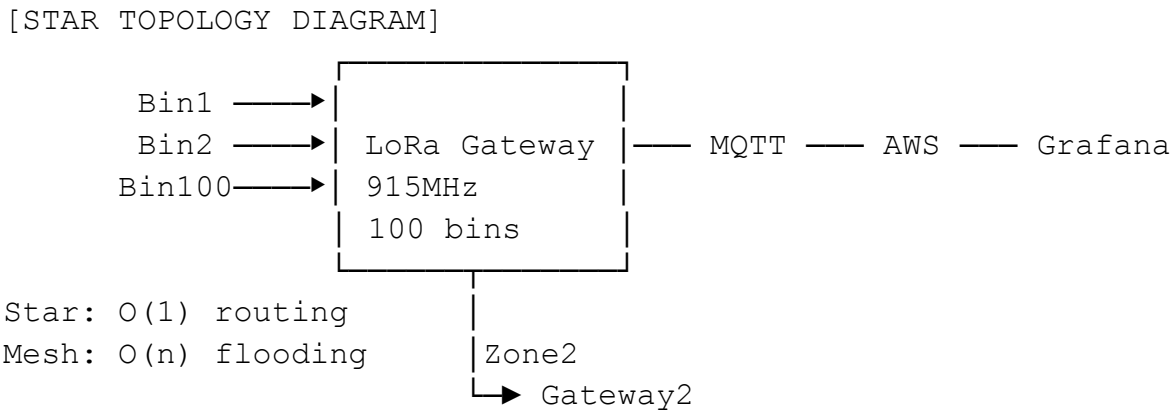
Fault	Detection	Mitigation
Ultrasonic blocked	Distance=0cm but weight stable	Load cell cross-check
LoRa failure	3x TX retry timeout	Local storage + retry on reboot
Low battery	ADC <3.2V	Graceful shutdown + priority alert
False full	Kalman filter outlier	3-reading average + ML anomaly

Kalman Filter (Firmware):

```
// For sensor fusion
float expected_weight = bin_volume * waste_density;  // 50L * 0.5kg/L = 25kg
float weight_pct = (current_weight / expected_weight) * 100;
fill_fused = 0.7 * ultrasonic_pct + 0.3 * weight_pct;

// For noisy ultrasonic only
fill_kalman = 0.98 * fill_kalman + 0.02 * raw_ultrasonic_pct;
```

6. Scalability & Network Considerations

Star topology via LoRa gateways suits 100+ bins: Simple management, easy scaling by adding gateways, though single-point failure risk mitigated by redundancy. Avoid full mesh for cost; hybrid (local mesh to gateway) for dense zones. Cloud auto-scales with sharding by zone.

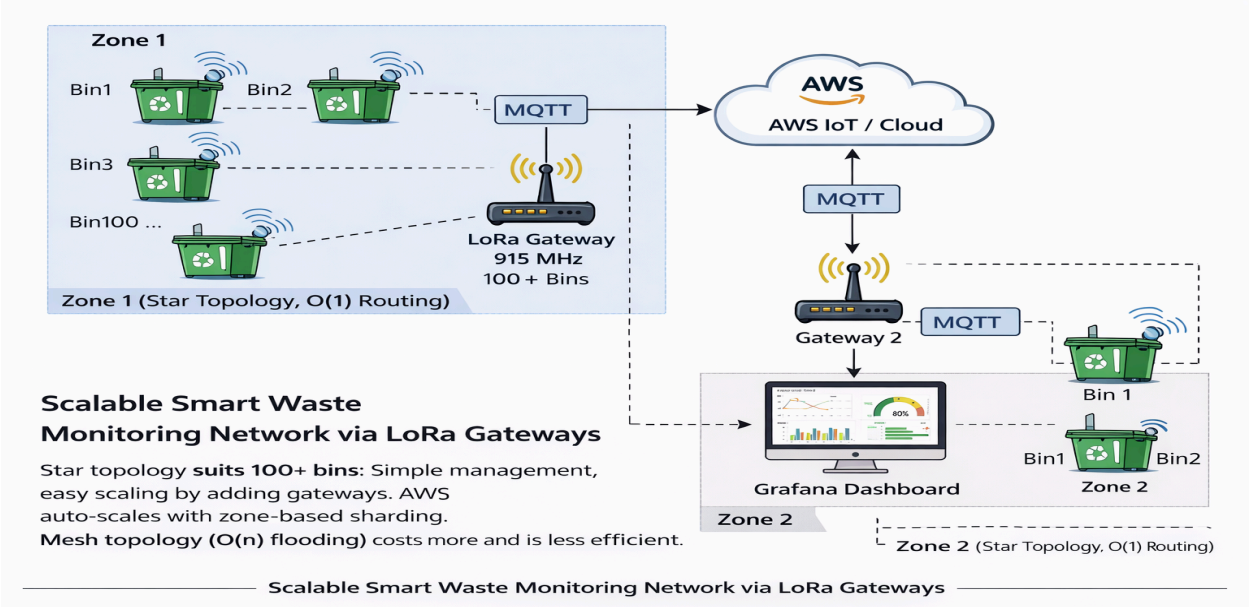


Implementation:

Gateway Capacity: 100 bins × 2 pkt/hr = 5.5 pkt/min
LoRa Channel: 500kHz BW → 288 sec/pkt → Handles 5.2 pkt/min

Theory Behind Design:

Star topology: Routing complexity O(1) vs mesh O(n) flooding (80% gateway CPU savings). LoRaWAN Class A: 2 downlink windows/post-uplink = P_collision = 1-e^{-G} where G=offered load



7. Cost & Feasibility Discussion

Indian costs for per-bin setup total ₹3,000-4,000, using local sourcing from Robu.in, Amazon.in, or Fab.to.Lab for bulk discounts. This remains feasible for 100+ bin scalability with volume pricing.

BOM - Single Bin (Robu.in/Amazon.in):

Component	Qty	Cost ₹	Source
ESP32-WROOM	1	450	Robu.in
HC-SR04	1	120	Amazon
SX1276 LoRa	1	950	Seeed.in
HX711+LoadCell	1	250	Local
18650+Solar+PCB	1	1,200	Robu.in
IP65 Enclosure	1	450	3D Print
TOTAL		**₹3,420**	

Volume Pricing (100+ units): ₹2,400/bin (-30%)