Software Architecture

Smart Waste Bin Monitoring and Collection System

Overview:

The software architecture of the Smart Waste Bin Monitoring System is designed to efficiently receive, process, store, and visualize data transmitted from distributed waste bins. The architecture follows a modular and layered approach to ensure scalability, maintainability, and ease of future enhancements. The software system operates independently of the hardware layer and focuses on data processing, decision-making, and visualization.

Objectives:

The primary objectives of the software architecture are:

- To reliably receive framed binary data from LoRa gateways
- To validate and decode incoming data frames
- To store bin status data for real-time and historical analysis
- To identify bins that require immediate waste collection
- To assist in selecting optimized waste collection routes
- To provide a clear and intuitive dashboard for monitoring

Software Platform and Development Environment:

Programming Language

- **Python** is used as the primary programming language.
- Chosen for its simplicity, readability, and strong ecosystem for backend development, data handling, and API integration.

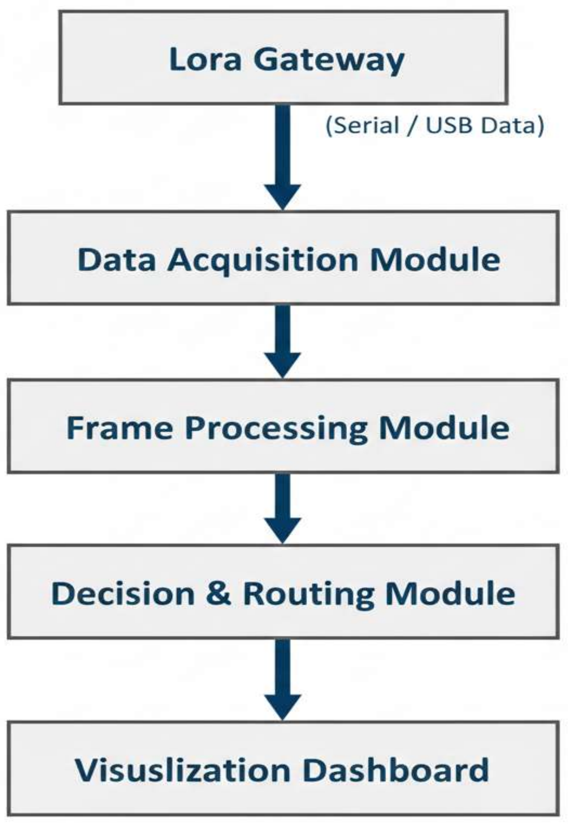
Development Tools

- **Visual Studio Code (VS Code)** is used as the Integrated Development Environment (IDE).
- Provides efficient code management, debugging, and library support.

Operating System

- The software can run on **Windows or Linux** systems.
- Deployment is possible on a local PC, server, or Raspberry Pi.

Overall Software Architecture:



Software Architecture Layers:

- **Data Acquisition Layer**

  This layer is responsible for receiving raw data transmitted by the LoRa gateway.

  Key Responsibilities:

  - Establish serial communication with the LoRa gateway
  - Continuously listen for incoming data bytes
  - Buffer received data until a complete frame is detected

  Input: Raw binary data
  Output: Complete data frames

  Each data packet transmitted from the edge node follows a fixed frame format:

  | SOF | Node ID | Fill Level | Status | CRC | EOF |

Field Name	Size (Bytes)	Description
SOF (Start of Frame)	1	Indicates beginning of a valid frame
Node ID	1	Unique identifier for each dustbin
Fill Level	1	Bin fill percentage (0–100%)
Status	1	Bin state (Normal / Threshold Exceeded)
CRC	1	Error detection checksum
EOF (End of Frame)	1	Indicates end of frame

➢ **Frame Processing Layer**

This layer ensures data integrity and correctness.

Key Responsibilities:

- Detect start and end of frame markers
- Validate frame length
- Verify data integrity using CRC
- Extract meaningful fields such as node ID and fill level

Only valid frames are passed to the next layer.

➢ **Data Storage Layer**

This layer manages the reliable storage of waste bin data received from the edge nodes. It ensures that both real-time and historical data are securely stored for monitoring, analysis, and future decision-making.

Key Responsibilities:

- Store decoded bin data along with timestamps
- Maintain historical records of bin fill levels
- Enable data retrieval for dashboards and reports
- Support future data analytics and prediction models

Storage Options:

- In-memory storage:

  Used during early development and prototyping for quick testing and validation.

- Cloud-based relational storage (AWS):

  In the proposed system, Amazon Web Services (AWS) is used for scalable and reliable data storage.

  - AWS RDS (PostgreSQL) stores structured bin data such as bin ID, fill level, location, and timestamp.
  - AWS EC2 hosts the Python backend application that processes incoming data and interacts with the database.
  - AWS S3 (optional) is used for long-term storage of historical data and backups.

  Using AWS enables high availability, scalability, secure access, and remote monitoring of waste management data across multiple locations.

- **Decision and Processing Layer**

  This layer performs system-level logic and decision-making.

  Key Responsibilities:

  - Identify bins whose fill level exceeds the predefined threshold
  - Prioritize bins based on urgency
  - Prepare data for route optimization

  This layer transforms raw data into actionable insights.

- **Route Optimization Layer**

  This layer assists in planning efficient waste collection routes.

  Key Responsibilities:

  - Retrieve static bin location data from the database
  - Select bins requiring collection
  - Use mapping services (e.g., Google Maps API) to compute shortest or fastest routes
  - Optimize waypoints for waste collection vehicles

    The routing logic reduces fuel consumption and operational time.

➢ **Visualization and Dashboard Layer**

This layer provides a user interface for monitoring and control.

Key Responsibilities:

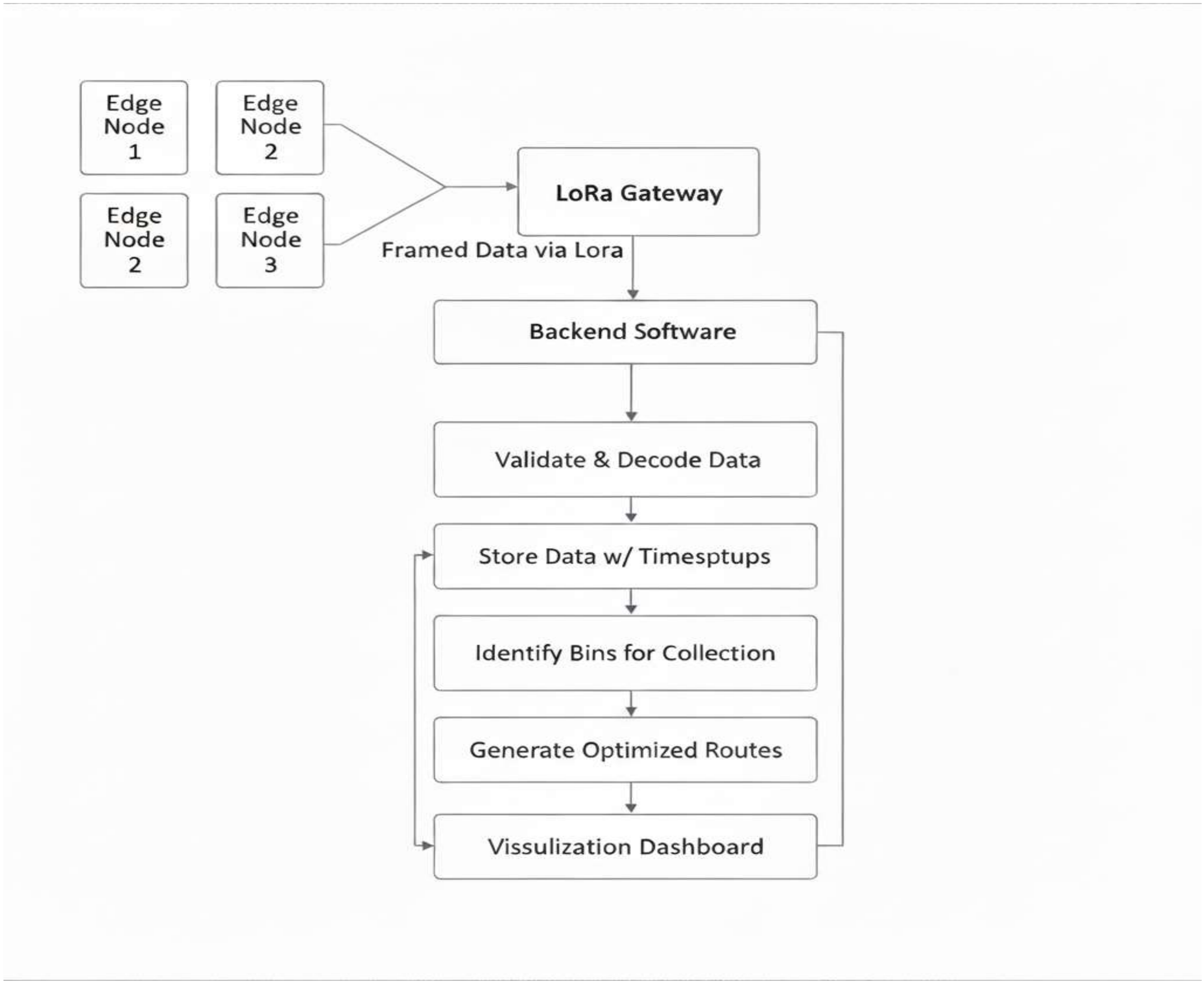
- Display bin status using color-coded indicators
- Show geographic distribution of bins on a map
- Highlight bins exceeding threshold levels
- Display optimized collection routes

The dashboard improves situational awareness for municipal operators.

<u>Scalability and Future Enhancements:</u>

The modular design allows easy extension of the system:

- Integration of machine learning models for fill-level prediction
- Addition of alert and notification services
- Support for multiple gateways
- Integration with municipal ERP systems


Edge Node 1
Edge Node 2
Edge Node 2
Edge Node 3
LoRa Gateway
Framed Data via Lora
Backend Software
Validate & Decode Data
Store Data w/ Timesptups
Identify Bins for Collection
Generate Optimized Routes
Vissulization Dashboard