# **IoT Project Report**

# **GPS & RFID-Based Train Location Monitoring & Collision Avoidance System**

# **(Team 15)**

Anagha Prajapati (2024101007)

Arushi Shukla (2024101075)

Navneet Kumar Gupta (2024111030)

**Introduction**

Indian Railways faces significant challenges with train collisions and derailments, causing loss of life and infrastructure damage. While the Kavach system, an ATP (Automatic Train Protection) solution, aims to address these concerns, its extensive retrofitting requirements make implementation time-consuming and resource intensive.

This project aims to develop a proof of concept for an IoT-based Train Collision Avoidance System that reduces the need for retrofitting by using GPS for real-time train location tracking and RFID for track identification. The goal is to provide a scalable, cost-effective interim safety solution that bridges the gap until the full deployment of the Kavach system.

**Objectives**

* Develop a handheld prototype using ESP32, GPS module and RFID reader representing the onboard hardware unit to be installed in trains.
* Design and implement a cloud-based software stack to store and process real-time train data.
* Implement and test real-time alert generation for the following scenarios:
  + Collision Alerts: Detection and resolution of potential train collision risks.
  + Route Deviation Alerts: Notifications for route deviations and their resolution.
  + System Status Alerts: Indications for train halts, resumptions, and other system messages.
* Simulate multi-line rail traffic scenarios, demonstrating the system’s ability to distinguish between safe parallel train movements and actual collision risks.

**System Overview**

Hardware Components

* ESP32 Microcontroller: Core processing and communication unit for data acquisition and transmission.
* GPS Module (NEO-6M): Provides real-time geolocation data to track train positions.
* RFID Reader (RC522): Identifies track location by reading strategically placed RFID tags.

Software Stack

* Database (MongoDB Atlas): Stores train data, route data, logs, and alerts.
* Backend (FastAPI): Handles API requests, processes database interactions, and executes alert logic. Deployed on [Render](https://iot-project-c3wb.onrender.com/).

Communication

* The ESP32 transmits data via Wi-Fi using HTTPS requests to the FastAPI backend hosted on [Render](https://iot-project-c3wb.onrender.com/). Since the backend is cloud-hosted and always online, there is no need for any local server setup.

**Implementation Details**

GPS Fix Quality

**Horizontal Dilution of Precision (HDOP)**, measures satellite geometry's impact on GPS accuracy. Lower HDOP values indicate better geometry and higher accuracy, while higher values signal weaker accuracy due to poor satellite alignment. We are using HDOP values reported by Neo-6M module along with the number of satellites, to classify GPS fix quality.

| HDOP Range | Minimum Satellites | Category | Description | Estimated Error (m) | Colour |
| --- | --- | --- | --- | --- | --- |
| ≤ 1.0 | ≥ 6 | Excellent | Ideal GPS fix | < 5 |  |
| 1.0 – 2.0 | ≥ 5 | Good | Strong and accurate fix | 5 – 10 |  |
| 2.0 – 5.0 | ≥ 4 | Moderate | Acceptable, some errors | 10 – 25 |  |
| 5.0 – 10.0 | ≥ 3 | Poor | Weak GPS fix | 25 – 50 |  |
| > 10.0 or N/A | < 3 or None | Very Poor/Invalid | Very high error or no fix | > 50 or No Fix |  |

A diagram of a structure with Crust in the background

AI-generated content may be incorrect.

^References: [GIS Geography: [GPS Accuracy - HDOP, PDOP, GDOP & Multipath]](https://gisgeography.com/gps-accuracy-hdop-pdop-gdop-multipath/) and [Novotech: [Understanding Horizontal Dilution of Precision (HDOP)]](https://novotech.com/pages/horizontal-dilution-of-precision-hdop)

Hardware Details

* Both GPS modules and RFID readers were tested individually before integration.
* Two hardware prototypes were assembled.
* The hardware prototypes sent the logs every 6 seconds in in\_service\_running mode.
* The hardware included a push button- acting as a start/stop actuator for the train.
* It also included an LED indicator, which-
* turned on only when ESP32 was connected otherwise remained off.
* blinked for each successful https request, so it blinked every 6 seconds when the esp32 sent the logs
* blinked continuously, if the train encountered any risk

System Details

* The backend runs a Monitoring Task on every active train, every 10 seconds.
* Collisions are detected when the distance between two trains is less than 15 meters on the same route.
* Route deviations are detected when a train deviates from its assigned route by more than 10 meters.
* Train stopped alerts are generated when an in\_service\_running mode train moves less than 5 meters between consecutive logs.
* For every kind of alert, a resolution alert is generated if the problem has been resolved.

Hardware Configuration

**A circuit board with wires

AI-generated content may be incorrect.**

Data Flow

A black background with white text

AI-generated content may be incorrect.

System Details

* The backend runs a Monitoring Task on every active train, every 10 seconds.
* Collisions are detected when the distance between two trains is less than 15 meters on the same route.
* Route deviations are detected when a train deviates from its assigned route by more than 10 meters.
* Train stopped alerts are generated when an in\_service\_running mode train moves less than 5 meters between consecutive logs.
* For every kind of alert, a resolution alert is generated if the problem has been resolved.

Route Map

A map of a city

AI-generated content may be incorrect.

**Data & Analysis**

* Develop a handheld prototype simulating GPS-based train tracking and RFID-based track identification.
* Detect and alert collision risks and track deviations in real-time.
* Provide a scalable, low-cost alternative to existing solutions like Kavach.
* Simulate multi-line rail traffic to demonstrate conflict detection and resolution.

**Conclusion**

Lessons Learned

* Database (MongoDB Atlas): Stores train data, route data, logs, and alerts.
* Backend (FastAPI): Handles API requests, processes database interactions, and executes alert logic. Deployed on [Render](https://iot-project-c3wb.onrender.com/).

Lessons Learned

* Database (MongoDB Atlas): Stores train data, route data, logs, and alerts.
* Backend (FastAPI): Handles API requests, processes database interactions, and executes alert logic. Deployed on [Render](https://iot-project-c3wb.onrender.com/).

Communication

* The ESP32 transmits data via Wi-Fi using HTTPS requests to the FastAPI backend hosted on [Render](https://iot-project-c3wb.onrender.com/). Since the backend is cloud-hosted and always online, there is no need for any local server setup.

**References**

* Develop a handheld prototype simulating GPS-based train tracking and RFID-based track identification.
* Detect and alert collision risks and track deviations in real-time.
* Provide a scalable, low-cost alternative to existing solutions like Kavach.
* Simulate multi-line rail traffic to demonstrate conflict detection and resolution.