Smarter Rail: Application of IoT for Route and Track Management Software Products

Karan Asthana
B.Tech in Information Technology,
JSS Academy of Technical Education, Noida.

I. ABSTRACT

Indian Railways is the 4th largest Rail network in the world, encompassing a total of 68,442kms. Managing such a huge network is a nightmarish task. Especially when a discrepancy in just, say 10 cm of the track can cause fatal mishaps. The magnitude of this can be realised by the fact that the bandwidth for error is less than 1.461e-8 km. Thereby, managing such a huge amount of rail network requires immense efficiency and precision. Doing these tasks manually is a lot of work, is slow and usually inefficient. Thereby, through this paper, I propose a universal data preparation system to facilitate the inter-compatibility of data received via different products working towards rail tracks security and an oscillation monitoring device. These, in effect with other similar products, would lead to a better, more efficient railway system. The basic idea of the execution of this project is that we will create a universal data system. A database that'll store data about the rail lines, that is, the GPS location of every reference-able point on the rail tracks, its distance from the previous major station. Further, I also propose a product that'll monitor and note the oscillation of the rail wheels using an accelerometer. The reading of these oscillations will be instrumental in deciphering the rail tracks' quality. The lesser the peak values of an accelerometer, the safer the track is to run trains on. In the case of high peak values, GPS locations would be noted down and reported for inspections and repairs.

II. INTRODUCTION

Rail tracks often wear out or get damaged due to excessive usage or over-exposure to extremes of temperatures. Thereby, an inspection of these tracks is a mundane, tiresome but essential work that needs to be carried on daily. Faulty or damaged tracks can lead to huge train accidents, which could, in turn, lead to loss of myriad of human lives or even a lot of materialistic things. Entrusting manual labour with such important tasks is a conundrum, ranging from the tiresome nature of the job to amount of precision required for it, it poses a lot of ifs and buts.

It would have a great positive impact if new technologies like Artificial Intelligence or the Internet of Things (IoT) to perform or automate these tasks with precision.

Usage of the Raspberry Pi along with different sensors has already been instrumental in automating some of these tasks. As stated in the paper by Kakoli Bannerjee et al, they have been able to automate the measurement of the temperature of the rail tracks or the measurement of the speed of trains, which, earlier, was done by the railway engineers or linesmen.

Raspberry Pi is a brilliant tool which can easily be coupled with different sensors to take data from the surroundings. It can then be used for all sorts of data computation, storage, manipulation etc. The Raspberry Pi is a kind of handy low-cost computer readily available anywhere. Being the size of a palm, it can fit in anywhere and is immensely mobile. Replacing traditional devices which were difficult to use because of their heavyweights is a lot easier due to such capabilities of the Raspberry Pi.

The system proposed by this paper involves basically 2 components -

1. Digital mapping of rail route data.
2. Monitoring of the Oscillation of the train

The rail definitely needs to have a centralised database of all its components. This will be vital in getting all the new devices to work on similar terms and to increase the speed for the location and then fixation of faults. Example of this can be seen in the second component of the project, which measures the peak values of accelerations across the X and Z axes. This measurement is done by detecting voltage changes in the accelerometer. Encountering such a peak Values signifies the bad quality of the track there. We store the GPS coordinates of the peak value location on the device. Next, on inspection of the data produced, we can then map the location to the nearest feature present on that route and easily find the fault and repair or replace it.

Fig 1. Shows a train derailed due to faulty rail tracks

III. LITERATURE SURVEY

A major cause of train derailments is the bad quality of railway tracks. These bad quality tracks cause the trains to oscillate from their mean positions in the X or Z axes. When these oscillations increase more than a fixed limit, it signifies the track to be unusable or unsafe. Presently, railway engineers carry out random inspections of the tracks and find out the possible pain points / weak sections of the track and replace or repair them. As discussed heavily in [2,3,4] the requirement for a proper management system is inevitable, the safety of trains and its passengers should be of foremost importance and it is an all-cost need. The measurement of track quality is to play a vital role in the safety of rail and its passengers.

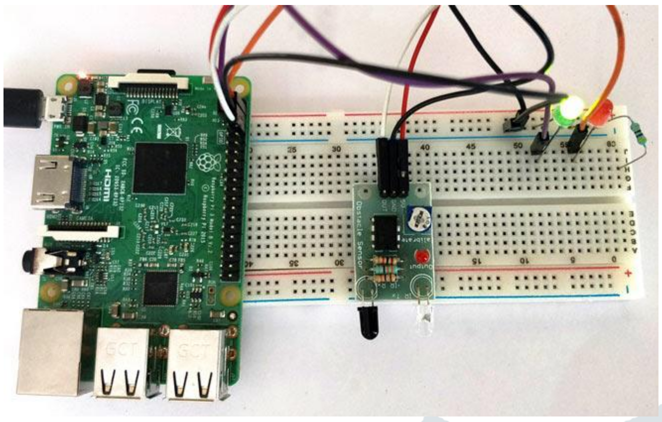
Fig 2. A Raspberry Pi connected to a sensor

Fig 3. A Tachometer sensor

IV. APPROACH

To fetch the distance of the track covered from the previous station we used a tachometer sensor [Fig. 3], and which was fit on the wheel and then the data sent by tachometer (number of revolutions done) was transferred to Raspberry Pi [Fig. 2] through GPIO pins for further computation.

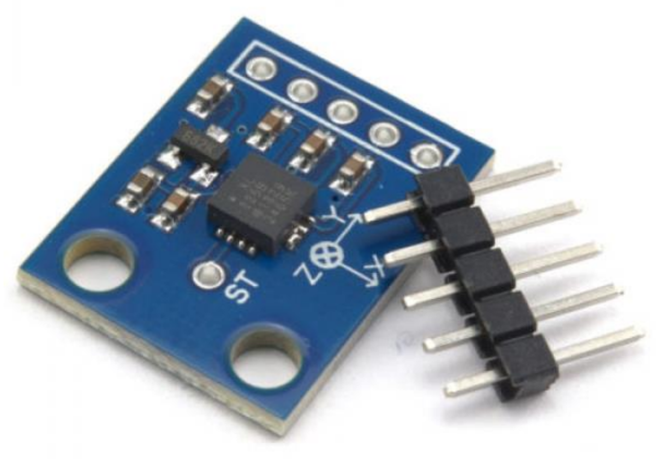

Fig 4. An Accelerometer Sensor

The Raspberry Pi had the measure of wheel diameter and thereby, corresponding distance travelled by the wheel was calculated using the formula :

Distance = 2 * π * radius(m) * number of revolutions

At every Km Post, there had to be manual intervention to change the present Kilometer. This was done since the present Km Posts aren't accurate, and a typical rail kilometre ranges from 800m to 1200m. Thus, the distancing was divided into two parts

- Number of Km Posts from the source station, and
- Distance (in m) travelled from the previous km.

At the occurrence of every feature, the engineers clicked corresponding buttons to feed the present distance and present location into the database for the corresponding feature.

Along with this, we had to measure the oscillation occurring in the wheels about X and Z axes. For this, the accelerometer was used, attached to the Raspberry Pi via GPIO pins. The accelerometer used to give the acceleration values in the 3 axes, of which X and Z were used and in cases of the values being greater than the permissible peak values, entries were made to a separate database system and alarms were beeped.

V. IMPLEMENTATION

STEP1*:*

The process begins with connecting all the sensors with the raspberry pi with correct connection points such that it facilitates the communication between the two.

STEP *2:*

The sensors are used to fetch the data from the outside world and transferred to the pi for calculations and use. The sensors are placed at the wheels to fetch the data.

STEP 3:

The sensor includes an accelerometer and a tachometer and a buzzer. The buzzer is blown when the safe limit of data is crossed.

STEP 4:

The Raspberry Pi creates a hotspot, the android app is connected to the hotspot to facilitate 2-way socket communication between them.

STEP 5:

As the wheels keep rotating, the tachometer keeps increasing its revolution count, which are instantly transferred to the Raspberry Pi, where distance travelled is calculated. The accelerometer also keeps sending its values to the Raspberry Pi which are maintained and manipulated using a different program. The Raspberry Pi then sends these data to the Android side using two different communication channels.

STEP 6:

The data is made visible in the GUI with touch input designed for best efficiency and smoother results. The flow is designed such that it is natural and yet diversified with options.

STEP 7:

The GUI is made on an Android App. The android app is developed using XML and Java language, it makes use of some open-source libraries.

STEP 8:

The Gpio pins interact with Raspberry Pi using a Python program. This python program has callbacks which keep listening to the changes in outputs of the GPIO pins connected to the sensors. These callbacks, in turn, perform computations and then subsequently transfer these values to the Android side in real time.

STEP 9:

The Android program keeps listening for the data to be communicated via the TCP connection. On receiving a packet we decipher the type of data sent (accelerometer or tachometer) and accordingly show it up in the UI. The local variables storing the present distances also keep getting updated. Thereby, whenever a feature button was clicked, the latest distance was fed to the system.

STEP 10:

The package is then bundled in an executable file with "rimraf" bundler and code is uglified and minified to make it inaccessible to anyone else. This is done for the security of the code.

STEP 11:

Attaching sensors to the wheels. To attach sensors to the trains' wheel we mounted them with tough mechanical containers. And fixed them to the wheels. The package as a whole work and connects with all the components with close bindings of software and hardware.

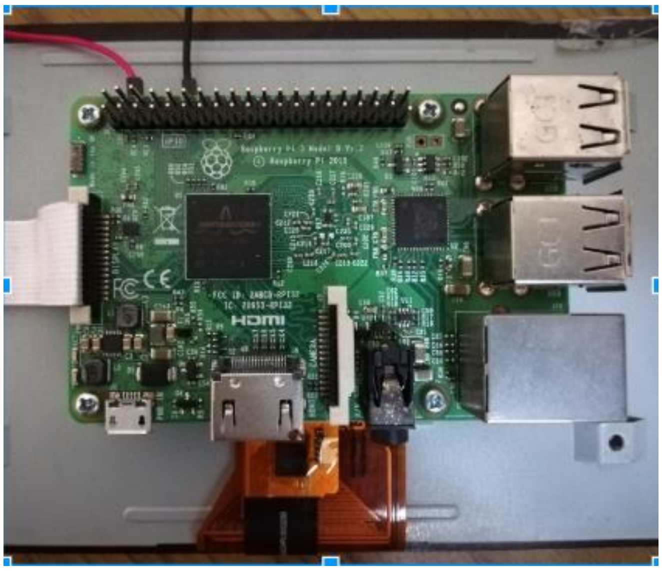

Fig 5. Raspberry Pi has 20 GPIO pins

VI. WORKING

All the required components such as the Raspberry Pi, tachometer, accelerometer, etc are connected together for the proper functionality to get accurate results, along with virtual connectivity with the Android Application via TCP communication after having created a socket between the Raspberry Pi and the Android App, all these connections are automated and done by automated scripts that run on the start of the Raspberry Pi and wait until a connection is made with the Android app.

The main outcome of this setup was to achieve accuracy in readily mapping railway features on a virtual database and accurately measuring the oscillation accelerations along the X and Z axes. To achieve this accuracy, a major roadblock was to not miss any output sent out by the sensors, whilst detecting it from the Raspberry Pi or while communicating it to the Android App.

It was observed that the data was mostly accurate, but a problem arose, when the device was made to oscillate at a frequency of 6Hz, that is 12 half-cycles per second were generated. At this speed, there was a loss in data from the accelerometer, which could be harmful to the successful deployment of the product. The problem was that the baud rates of the accelerometer had to match with that of the Raspberry Pi, without which the Raspberry Pi was missing certain signals. Hence, resulting in loss of data.

Every kilometre measured was further divided into blocks of 200m and the peaks incorporated in this block were then used to calculate a mean value for the block. This signified the quality of track for that block. At the distance of 200m (calculated via the tachometer), we programmatically changed the block for the track.

The Root Mean Square Value of Peaks was calculated for every block using the following formula:

Root Mean Square = $(\Sigma (pi^2)/ n) ^ {1 / 2}$

Where pi = Peak value of half-wave i;
n = number of peaks in the block

The Ride Index for a much longer section of rail tracks was calculated using the following Sperling index formula:

Ride Index = 0.896 X (Σ((bi^3) X F(fi) / n) ^ 1 / 10

Where n = no. of completed half cycles;
bi = Peak value of the half wave i;
fi = Frequency of the half wave i = (1/2Ti);
Ti = Time of the half cycle i;
F(fi) = Correction factor for the half wave i

Table 1 below shows the correction factors for X and Z axis

Frequency	X-axis	Z-axis
F < 0.5Hz	0	0
0.5<= f <5.4Hz	0.8f*f	0.325f*f
5.4<= f <=20Hz	650/(f*f)	400/(f*f)
F >20Hz	1	1

The above-used formulas and their usage to measure track quality for the Indian railways have been talked about in the paper [10] by K. V. Gangadharan, C. Sujatha and V. Ramamurti.

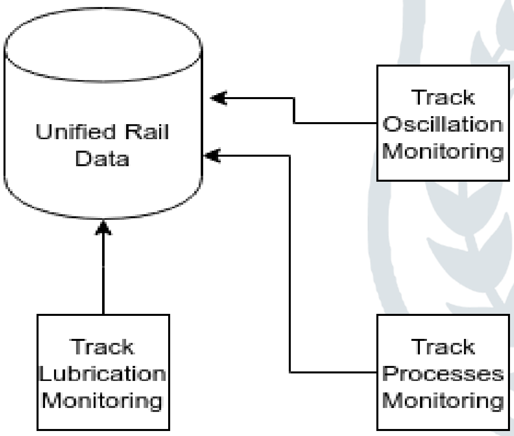
Unified Rail Data
Track Oscillation Monitoring
Track Lubrication Monitoring
Track Processes Monitoring

Fig 6. Centralized database system

Advantages of one format of files over all products used are mainly **data integrity** maximisation as all the files are inter-compatible. This leads to better collaboration between different products and/or analysis of the results of these products

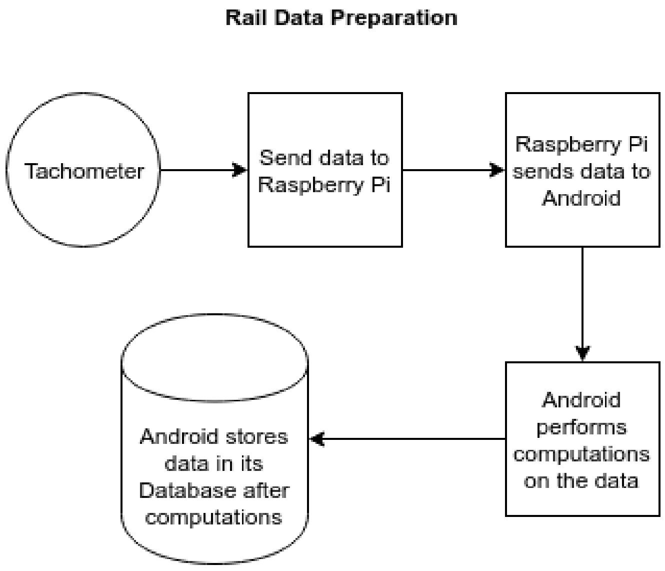
Rail Data Preparation
Tachometer
Send data to Raspberry Pi
Raspberry Pi sends data to Android
Android performs computations on the data
Android stores data in its Database after computations

Fig 7. Data Flow Diagram for the preparation of Rail Data

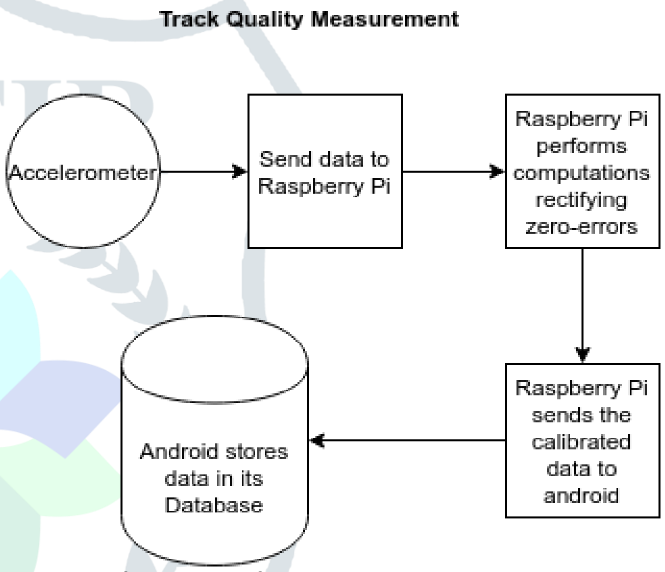
Track Quality Measurement
Accelerometer
Send data to Raspberry Pi
Raspberry Pi performs computations rectifying zero-errors
Raspberry Pi sends the calibrated data to android
Android stores data in its Database

Fig 8. Data flow diagram for the Oscillation Monitor

VII. RESULTS

Both the modules are successfully installed and are capable of achieving the work for which they are designed. First of all the data preparation module keeps calculating the distance from the previous station with minimal human intervention. The GPS module embedded in the Android device also enables recording the features' GPS coordinates effectively. Shown in Table 1 below are some results captured from the Data preparation system.

Km (km)	Distance (m)	Latitude (N)	Longitude (E)	Feature
1055	0000	26.42.33	80.59.0	New KM Start
1055	0007	26.42.34	80.58.52	Post (TP)
1055	0028	26.42.35	80.58.53	Gradient Start
1055	0159	26.42.30	80.58.17	Post (TP)

1055	0364	26.42.23	80.58.54	TP
1055	0609	26.42.15	80.58.55	TP
1055	0700	26.42.12	80.58.55	Signal Post
1055	0899	26.42.06	80.58.56	Bridge

TABLE 1. Sample data for route preparation

For the oscillation monitoring module, the major recordings were as shown in table 2. The readings mentioned in the table show the readings for a span of 4kms starting from km 1045 to 1055. This module keeps getting the distance from the other module of data preparation. The accelerometer readings are also fed into the system as soon as a peak value above the predefined threshold value occurs. Then, at the time of report generation, we also map the nearest TP to the peak reading.

Both these modules which can be used as standalone products or can be incorporated into one entity have a huge future lying ahead and can play a pivotal role in changing the face of how the Railways function, making it more efficient and safer to travel on.

Distance	Vertical Peak	Lateral Peak	Nearest Post
60.98	0.21	0	TP220
1016.8	0.21	0	TP220
191.31	0.16	0	TP398
520.08	0	0.16	TP408
198.26	0.16	0	TP490
579.3	0.18	0	TP492
921.85	0.31	0	TP502
21.35	0.19	0.16	TP766

TABLE 2. Sample data for oscillation measurement

VIII. CONCLUSION

This project aimed at consolidating data for a railway system and then automating things like measurement of track quality using the oscillatory acceleration from the mean position in the X and Z axes. And this will definitely help prevent major train accidents that occur due to inferior track quality. The consolidated system being developed using the tachometer will serve as a common point of reference for every project. Every project can have a mapping, which singles out the railway feature present nearest to the said location of the anomaly. Automatic alerts using SMS/email will also enhance and expedite the maintenance and repair of the affected tracks/parts of the rail ecosystem.

IX. REFERENCES

[1] https://en.wikipedia.org/wiki/List_of_countries_by_rail_transport_network_size

[2] http://www.astanadigital.com

[3] https://www.circuitdigest.com

[4] https://www.alexlnd.com

[5] https://pdfs.semanticscholar.org/671e/4c731e0ec2b4da82b0482373d61531730a9d.pdf

[6] https://drive.google.com/file/d/1tKiPztLoUMjAStUmfeiUobq9VZpuJLCy/view?usp=sharing

[7] http://www.jetir.org/papers/JETIR1906018.pdf

[8] https://pdfs.semanticscholar.org/671e/4c731e0ec2b4da82b0482373d61531730a9d.pdf