CS-684-2018 Final Report

Next Jan Transport System

Abhijit Divekar, 163079036

AjayKumar Maurya, 16307R009

Punit Jain, 17307R016

Table of Contents

1. Introduction 3
2. Problem Statement 3
3. Requirements 3
    3.1 Hardware Requirements 3
    3.2 Software Requirements 4
4. System Design 5
5. Working of the System and Test results 6
6. Discussion of System 8
7. Future Work 8
8. Conclusions 9
9. References 9

1. Introduction

In a Metropolitan city like Mumbai, majority of population depends on public transport i.e. Local Trains and Buses.

Out of the two, buses are the one that reaches each and every corner of the city.

But, buses are very unpredictable and in case if there is any delay/cancellation, people are not informed, which leads to formation of large queues at the Bus stop and people are then forced to take other expensive means of transport.

Next Jan Transport System (NJTS) uses GPS and EspLoRa32 network to locate the exact position of buses and Google API's to calculate the Estimated Time of Arrival (ETA) at the bus-stops.

The Information regarding location of buses with their respective Bus IDs and timestamp is stored at AWS (Amazon Web Services)-DynamoDB Database.

This data from DynamoDB is accessed by a webpage in which location of bus-stops (currently hard-coded) can be used to calculate ETA using Google Maps API.

This implementation will help the public to efficiently manage their time and money, and also it will improve the standards of the service providers which would increase their number of users.

This is sort of a hybrid (RF + IoT) solution which ensures to be a cost effective solution for bus service providers.

2. Problem Statement

Public transports are considered as an important feature of city as it ensures proper functioning of city. Public transport services like buses are important ad they are cheap and can reach each and every corner of the city. However, this service is also highly unpredictable which leads to unnecessary inconvenience to passengers and this reduces their interest in using the service. The reduction here chooses private means which firstly, are expensive and secondly, leads to huge traffic on roads.

Ensuring the predictability of such a service can increase the number of passengers. This in-turn leads to unclogging of roads and ensures a cheaper travel.
The current Implementation uses GPS-GPRS that sends the just the location data whenever requested through an SMS.
With Next Jan Transportation system, we think of building a EspLoRa32 network that can provide real time update of location and Estimated Time of Arrival of that bus on a particular bus stop. This ensures free info update to passengers (website or App) as well as statistical data to bus service providers that can help them to take decisions over route and bus frequency.

3. Requirements

3.1 Hardware Requirements

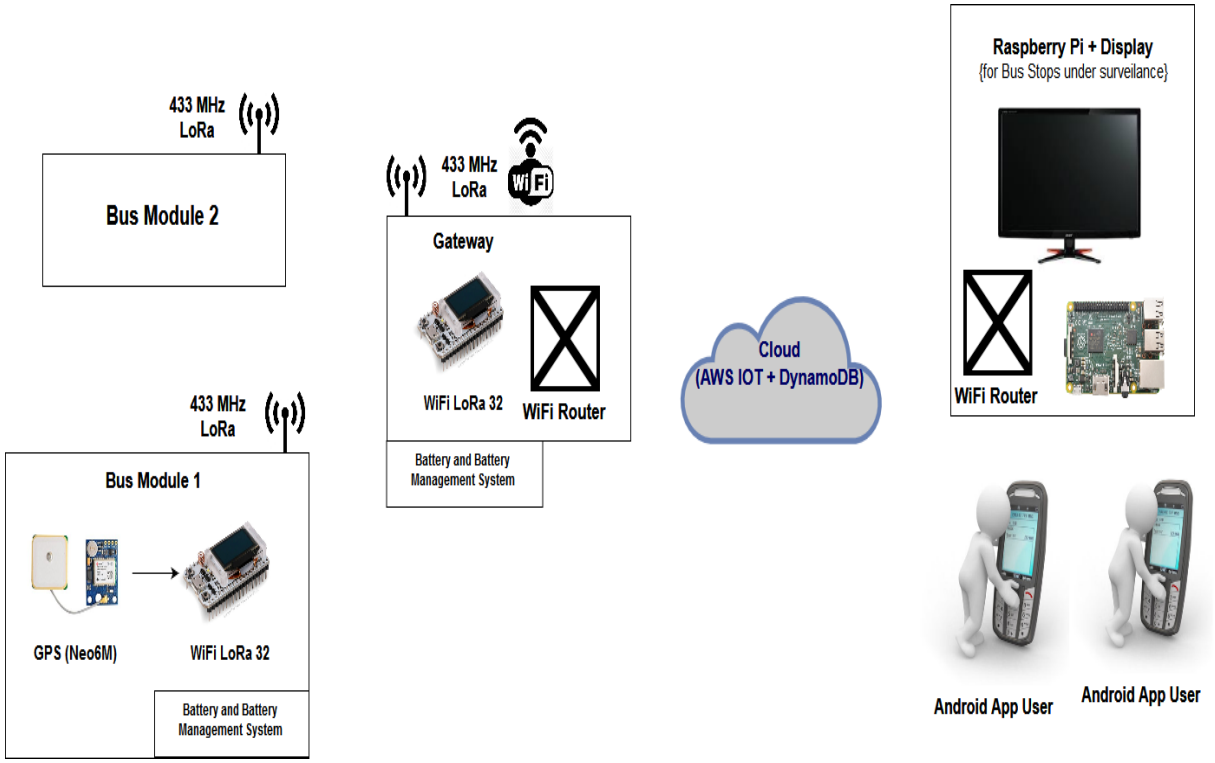
1. **Heltec Wi-Fi LoRa 32(EspLoRa32):** For Sending data to another EspLoRa32 using RF and to cloud using internet.
2. **NEO 6M GPS:** For getting current location (latitude, longitude) of the bus.

3.2 Software Requirements

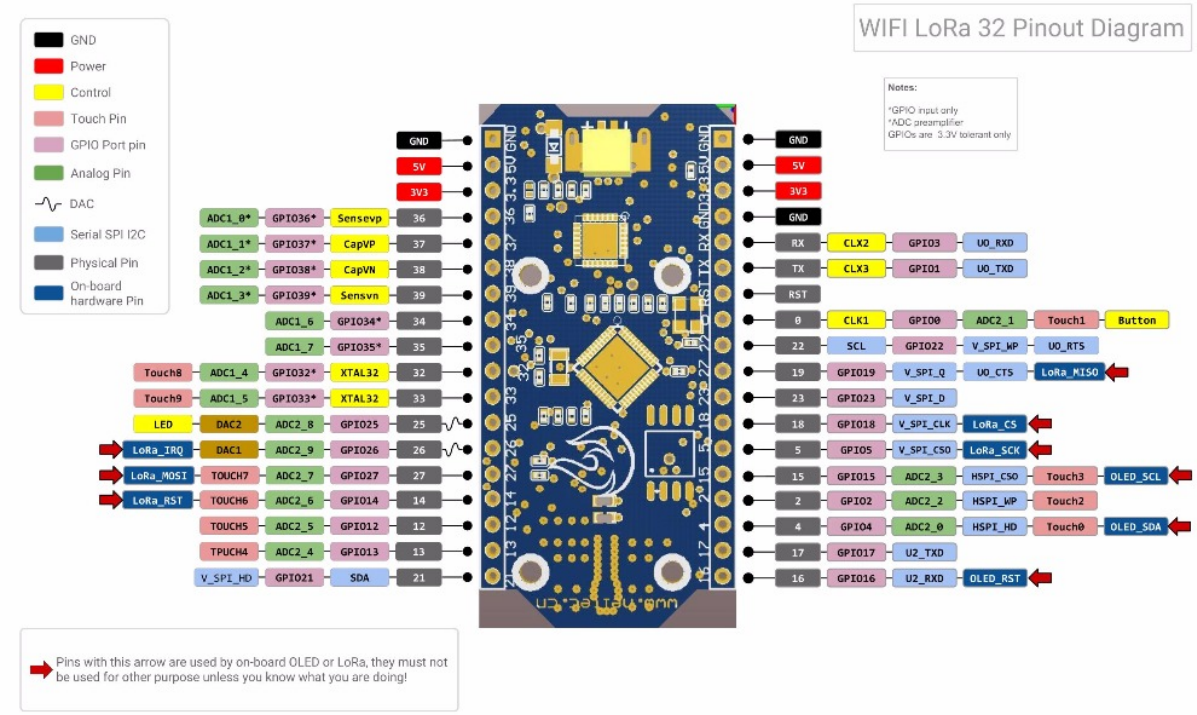
1. **Arduino IDE:** For compiling and uploading program on EspLoRa32.
2. **AWS-IoT:** For Secure communication between EspLoRa32 and Cloud.(Protocol-MQTT)
3. **AWS IoT and DynamoDB services:** For secure communication with device and Storing Information/Data (Bus ID, gps coordinates, Timestamp) related to bus.
4. **Google Maps API's:** For Estimated Time of Arrival calculation, for reverse geo-coding (making a human readable address out of gps coordinates).

4. System Design

Block Diagram



Pin-outs and Connection:



- On Transmitter Side.
  - UART communication between Neo-6M and EspLoRa32 at 115200 baud rate.
  - Connect VCC, GND to +5V Supply and ground respectively.
  - Connect RX of NEO- 6M to TX of EspLoRa32 (GPIO Pin 17).
  - Connect TX of NEO- 6M to RX of EspLoRa32 (GPIO Pin 16).
- On Receiver/Gateway Side.
  - Wi-Fi required for sending data to cloud.
  - AWS – Mutual Authentication Using Certificates is done by registering a “thing” in AWS_IOT.
  - Certificates for the “thing” are generated by AWS. Download these certificates.

- The Certificates are then uploaded in “aws-certificate.c” file in EspLoRa32 library.
- The Certificates helps AWS in authenticating the device. The device can then create a topic and AWS can then subscribe to that topic to obtain the message sent by the device.
- This is Message Queuing Transport Telemetry (MQTT) protocol.

➢ On AWS Side.

- A DynamoDB Rule is created.
- A Rule is something which is triggered when an event happens. A Rule also describes Action to be performed when such an event occurs.
- The DynamoDB Rule in our case triggers when it receives anything over MQTT and Action is “Inserting into DynamoDB”.
- Create a Table in DynamoDB that can store this data.

5. Working of the System and Test results

The whole System consists of a Transmitter (GPS+EspLoRa32), Receiver/Gateway module (EspLora32-Registered on AWS-IoT), and AWS IoT and DynamoDB services.

Heltec Wi-Fi LoRa 32 (EspLoRa32) is based on the ESP32 chip and has onboard Wi-Fi, Bluetooth, a 0.96 OLED display and a CP2102 USB to serial interface. It also works with the Arduino IDE.

On transmitter side, EspLoRa32 (abbreviated as Esp) is connected with **Neo-6M GPS** module. The GPS transmits data serially over UART at 115200 baud rate. GPS data is of NMEA format.

This data is then parsed at transmitter side on Esp and information regarding the bus like Bus ID is appended. This data is then broadcasted over **RF 433 MHz.**

Range tests were done to determine maximum distance at which Receiver/Gateway module could be to successfully obtain packets sent by transmitter.

Range Test Results

Line of Sight	**3.6 km**
Non Line of Sight	**1.3 km**

However this range was obtained with small antenna (frequency: 433MHz, Gain: 3dBi) with power amplifier and a better antenna (a dipole) this range could be enhanced till **10kms.**

Receiver/Gateway module (EspLoRa32), in the range, receives this data and modifies it into JSON format so as to be stored in **DynamoDB database**.

For communicating with AWS, receiver/gateway module first has to be registered on AWS and certificates generated by AWS for that have to be uploaded on receiver module.

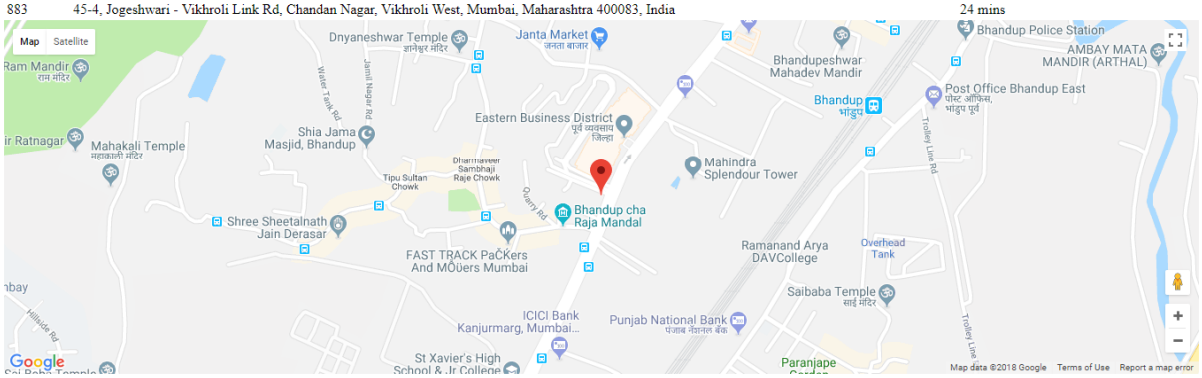
Once registered the module can publish the formatted data over MQTT by creating a “topic” and AWS can listen to this by subscribing to it. This data is then stored in DynamoDB table by creating a Rule in AWS that triggers on receiving data over that topic and takes action by storing it in specified DynamoDB table.

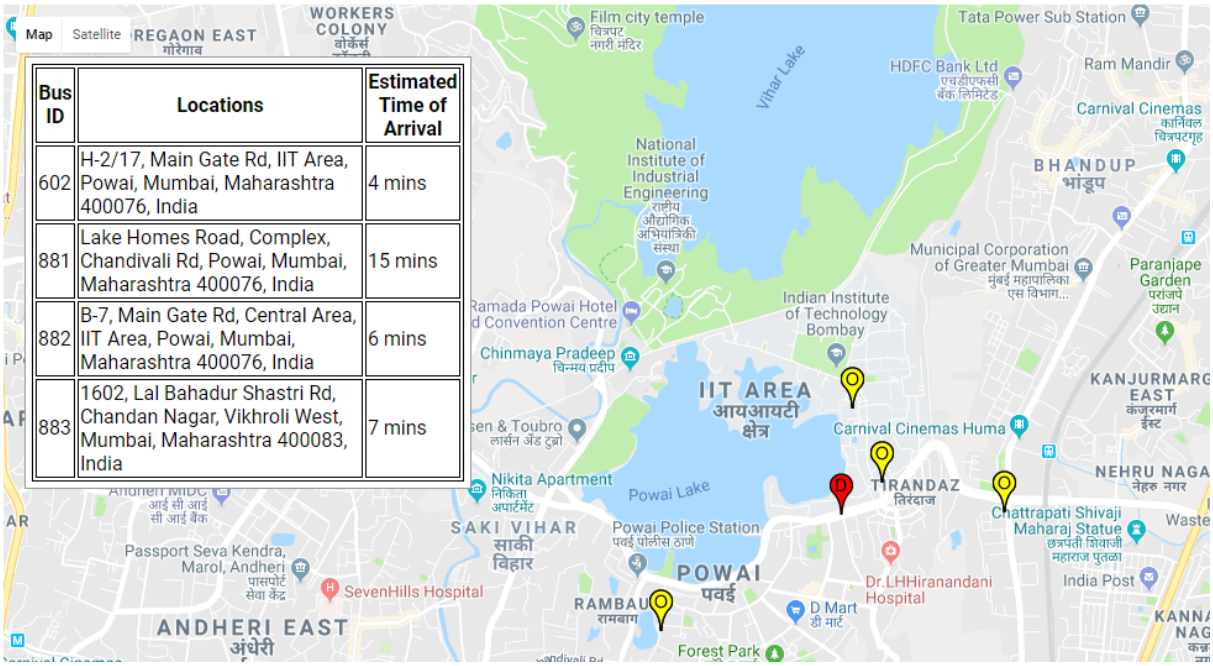
A webpage is created that accesses this DynamoDB table and Queries Google Maps API with source address as Bus location and Destination address as Bus-stop Location to calculate Estimated Time of Arrival (ETA) at the destination. This webpage queries DyanmoDB and Google API's at an interval of every10 seconds. The Location thus obtained is tagged in Google Maps and below is the screenshot of that webpage.(An alternative output that displays destination with "D"(red) marker and Buses with "O"(yellow) marker is also attached).

NextJan Transport System

Bus Stop: Sakinaka Bus Stop, Andheri East

Bus ID	Location	Estimated Time of Arrival
602	45, Lal Bahadur Shastri Rd, Ganesh Nagar, Bhandup West, Mumbai, Maharashtra 400078, India	30 mins
881	Jaltarang Apartments, Rambaug, Chandivali, Powai, Mumbai, Maharashtra 400072, India	14 mins
882	B-7, Main Gate Rd, Central Area, IIT Area, Powai, Mumbai, Maharashtra 400076, India	24 mins
883	45-4, Jogeshwari - Vikhroli Link Rd, Chandan Nagar, Vikhroli West, Mumbai, Maharashtra 400083, India	24 mins





Bus ID	Locations	Estimated Time of Arrival
602	H-2/17, Main Gate Rd, IIT Area, Powai, Mumbai, Maharashtra 400076, India	4 mins
881	Lake Homes Road, Complex, Chandivali Rd, Powai, Mumbai, Maharashtra 400076, India	15 mins
882	B-7, Main Gate Rd, Central Area, IIT Area, Powai, Mumbai, Maharashtra 400076, India	6 mins
883	1602, Lal Bahadur Shastri Rd, Chandan Nagar, Vikhroli West, Mumbai, Maharashtra 400083, India	7 mins

6. Discussion of System

a) What all components of your project worked as per plan?

Everything worked as per plan, only issues faced were in limits on number of Queries made per second that could be made to Google API. As we were using free plan there was a limit on number of Queries, but this could be mitigated using Premium plan.

b) What we added more than discussed in SRS?

In earlier discussions, plan was display the location information in just list format indexed by Bus ID. Tagging this information (using Marker) with Google Maps made it more usable.
Also Multiple Markers are added (one per bus) that tags location of each buses.
This can now be used for single bus info (by passenger), for multiple (selected) buses info (by Bus stops) and for all buses info (by Bus service providers) depending upon query that was made.

c) Changes made in plan from SRS:

In earlier discussions, plan was to build a full IoT solution. That was changed to a Hybrid (RF+IoT) because full IoT implied internet availability on every bus this could be very expensive from Bus service providers perspective. Instead, a network of Gateway modules and transmitter module could be created that receives broadcasted info by transmitter modules (buses) and push that to cloud. Since this network would have a very less number of Gateway modules as compared to large number of buses the number of internet hotspots needed would be very less this could reduce the charges faced by bus service providers dramatically.

7. Future Work

1. The Gateway module is currently programmed for receiving info from single transmitter (due to unavailability of hardware). This could be extended to receiving info from multiple hardware using carrier-sense multiple access with collision detection (CSMA/CD).

2. An Android Application can be built that can access similar info from DynamoDB and display it to user.

3. The data generated could be logged and provided to Bus service providers. The statistics can give them peculiar insights about a particular route like, traffic density on the route, at what time of the day is traffic very dense. This statistics can help them to efficiently manage the frequency of buses on that route/ divert it to another route in near future. Also, it can help them in making a prediction system with higher accuracy of estimated time of arrival.

4. The network thus built can be used for others applications where data from different points are to be collected and processed.

8. Conclusions

Next Jan Transport System thus built can improve the standards of public transport by respecting the time and money of passengers and bus service providers.
This is a small step towards smart transportation which in turn can be thought of as an integral part towards building smart cities.

9. References

1. Registering Thing with AWS-IOT and MQTT pub-sub.
   https://www.exploreembedded.com/wiki/Secure_IOT_with_AWS_and_Hornbill_ESP32
2. AWS-IOT Tutorials
   https://docs.aws.amazon.com/iot/latest/developerguide/iot-gs.html
3. AWS-DynamoDB Tutorials
   https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GettingStarted.html
4. Google Maps API
   https://developers.google.com/maps/