LPWAN

Software Design Overview Document

Group Members

Ali Şimşek

Bilal Özlü

Murat Kara

Emrah Kösen

Advisor

Cevat Şener

Table of Content

Table of Figures

1 - Product Description

The end-product

The project LPWAN (IoT Platform for Asset Tracking with Low-Power Wide-Area Network) aims to develop a system that continuously receives data from Lora sensors, transfer it to cloud/remote server via gateways and store data in it. Then, the system detects the probable location of the objects (traffic lights, signs etc.) by using time difference of arrival of the signals received by gateways. In addition, to check that the object is at the expected place or not, there is a dashboard to list and see the location and specifications of all the objects. A notification is sent in case of a displacement of any asset. Also, we use cloud system since it is reliable and cost-effective.

Which need it will serve?

Displacements of traffic lights or signs may cause serious problems in traffic. And these traffic objects may not be followed easily by the local authorities. There must be a way to follow these objects easily. Using Bluetooth and Wi-Fi is not so effective because of their low-range connectivity. On the other hand, using connections like GPS or 4G is too costly in terms of power and money. At this point, we develop an IoT project LPWAN, since it has some advantages:

- Long-range connectivity
- Low-powered
- Low cost.
- Secure
- Operates on free frequencies

2 - High Level System View

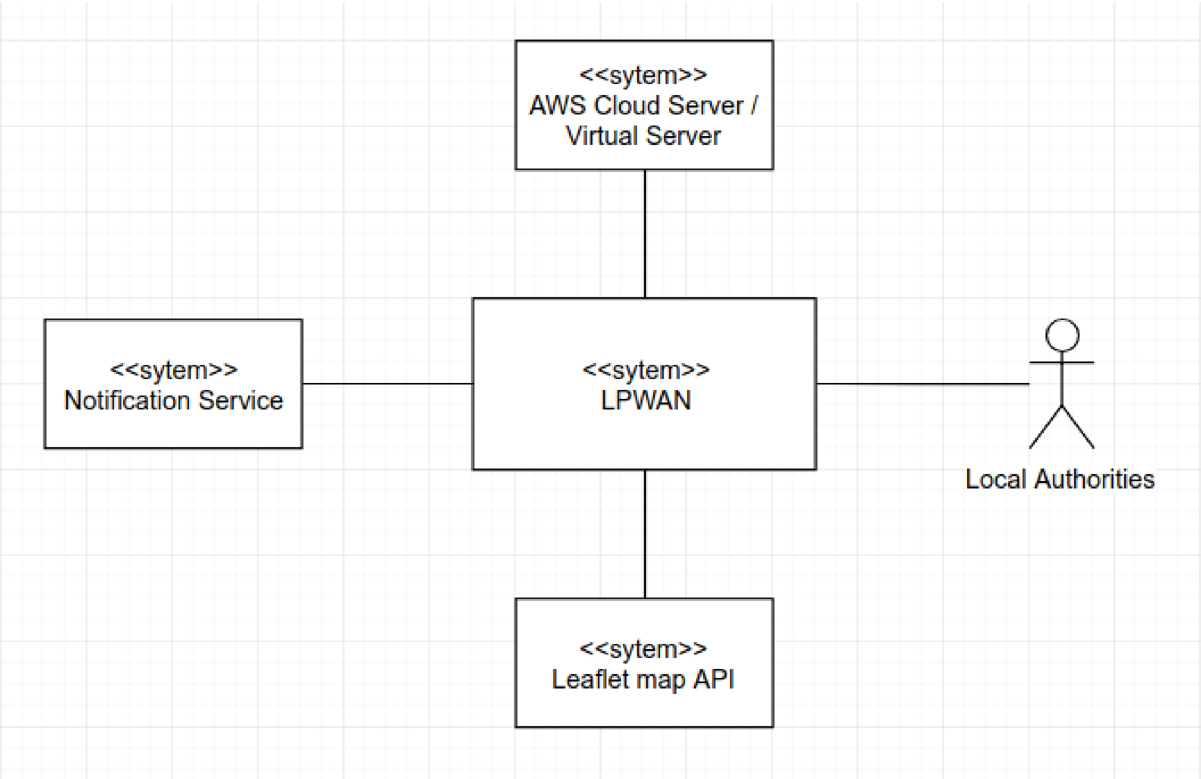

Figure 1: Context Diagram

LPWAN system consists of assets with LoRa sensors , gateways and a dashboard application. LoRa sensors transmit data to gateways via wireless communication LoRa protocol. Gateways transmit data to the cloud/remote server.

Cloud/remote server is where all data is stored. By using time difference of arrival of the signals received by each Gateway, location of the assets are detected with multilateration algorithm. A dashboard application shows the locations of all inventories in the system.

Leaflet map API provides a map for dashboard. All the end-nodes and gateways are shown on Leaflet map with their locations.

A notification e-mail is sent in case of displacement of any item by using a notification service.

3 - Overall Design

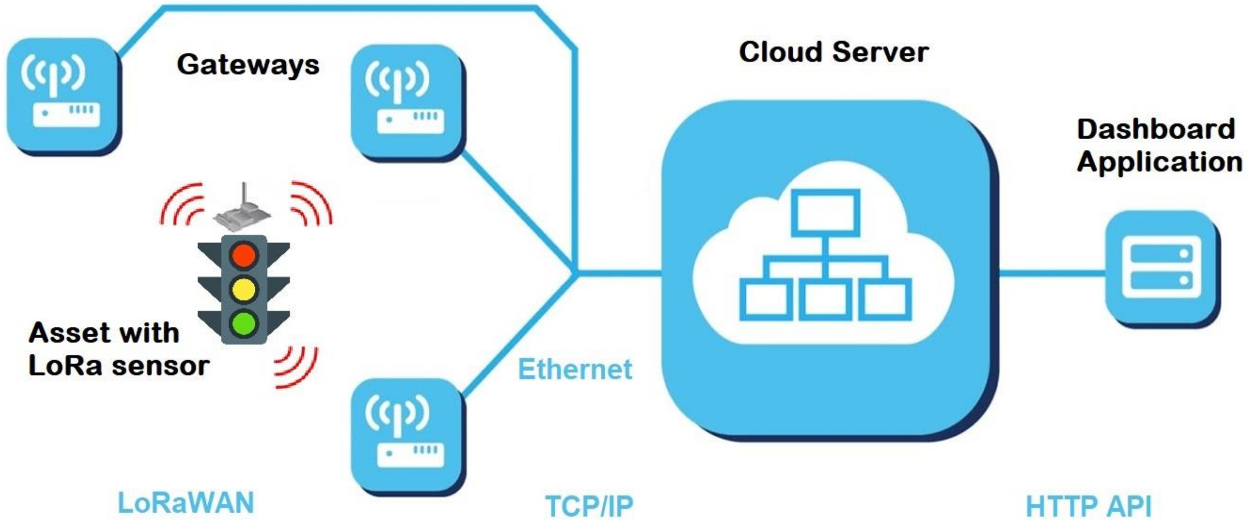

Figure 2: Overall Architecture of the system

LoRa sensors transmit unique id to gateways via wireless communication LoRa protocol . Whenever gateways receive the data transmitted from the LoRa sensors, they save the time of arrival and add this timestamp and unique gateway id to the sensor data, then send it to the cloud/remote server. By getting the difference of timestamps of each gateway, we apply multilateration algorithm to calculate approximate position of the end node. A dashboard application shows the locations of all assets in the system.

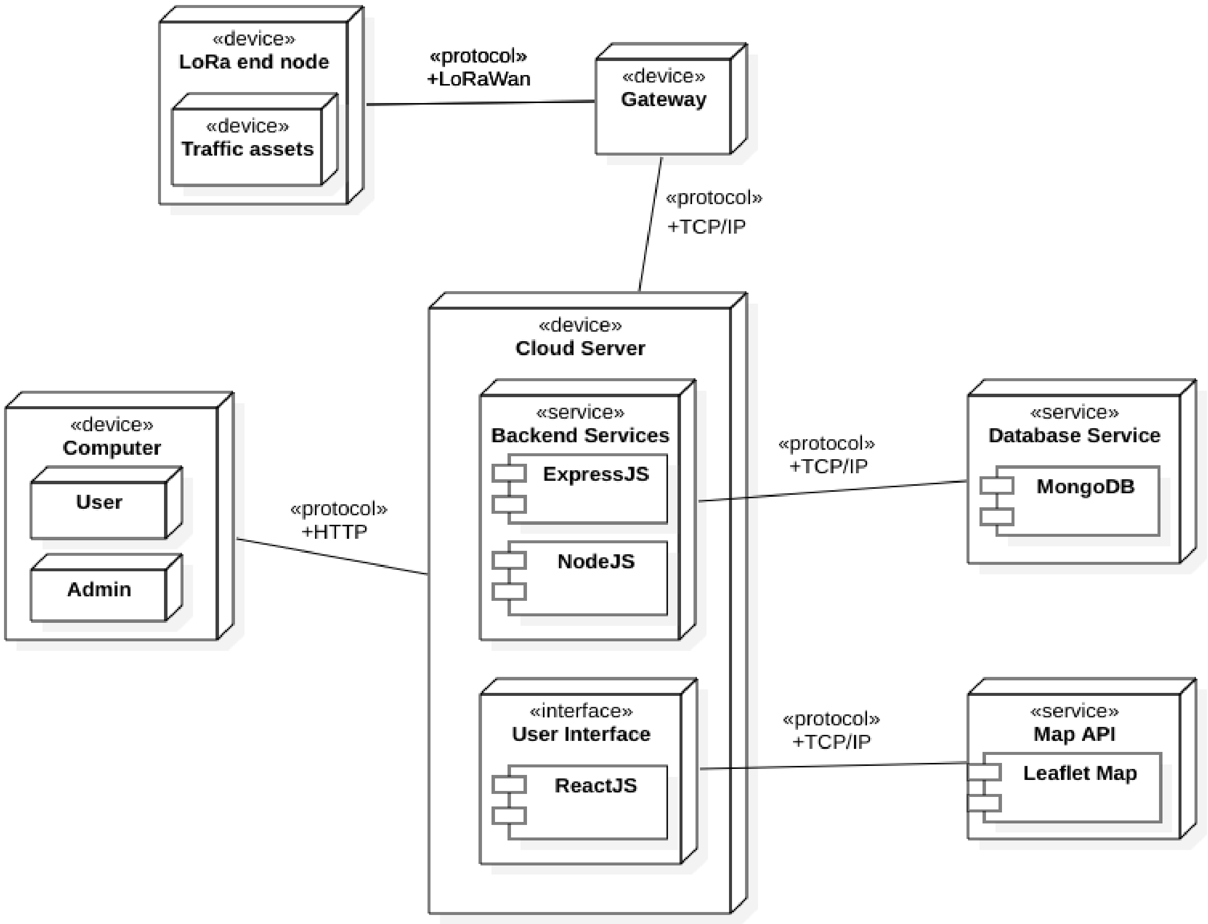

Figure 3: Deployment Diagram

Figure 3 illustrates the overall structure of our systems and tools that we used to built it.

We used ExpressJS and NodeJS to create the backend services. We created services with ExpressJS to handle HTTP requests. Then based on those requests, we manipulated the MongoDB database. Other than that, we used various NodeJS frameworks like mocha, axios, chai etc. to create unit tests for the services that we created. We also used POSTMAN to create HTTP requests and tested our services with those requests.

For the frontend of our project, we used ReactJS. We will use Leaflet Map API to show positions of assets on the map. This map will help user to track assets. There will also be a dashboard list that will show information about every gateway and end node on our system. For admins, there will be another list that shows the registered users and admins can add or remove users via this list.

LoRa end nodes will communicate with gateways using LoRaWAN protocol which will improve the battery life and allows us to build a long lasting system.

Since the system dedicated to local authorities' usage, not everyone can signup and use it. Therefore, we divided users as "admin" and "user". Administrator can define users and other admins with their username and initial password, also can do everything that a user can. In order to illustrate this, we drew two sequence diagrams. One is for admin's authorization and an example task. (Figure 4) The other one is for user's authorization and an example task. (Figure 5).

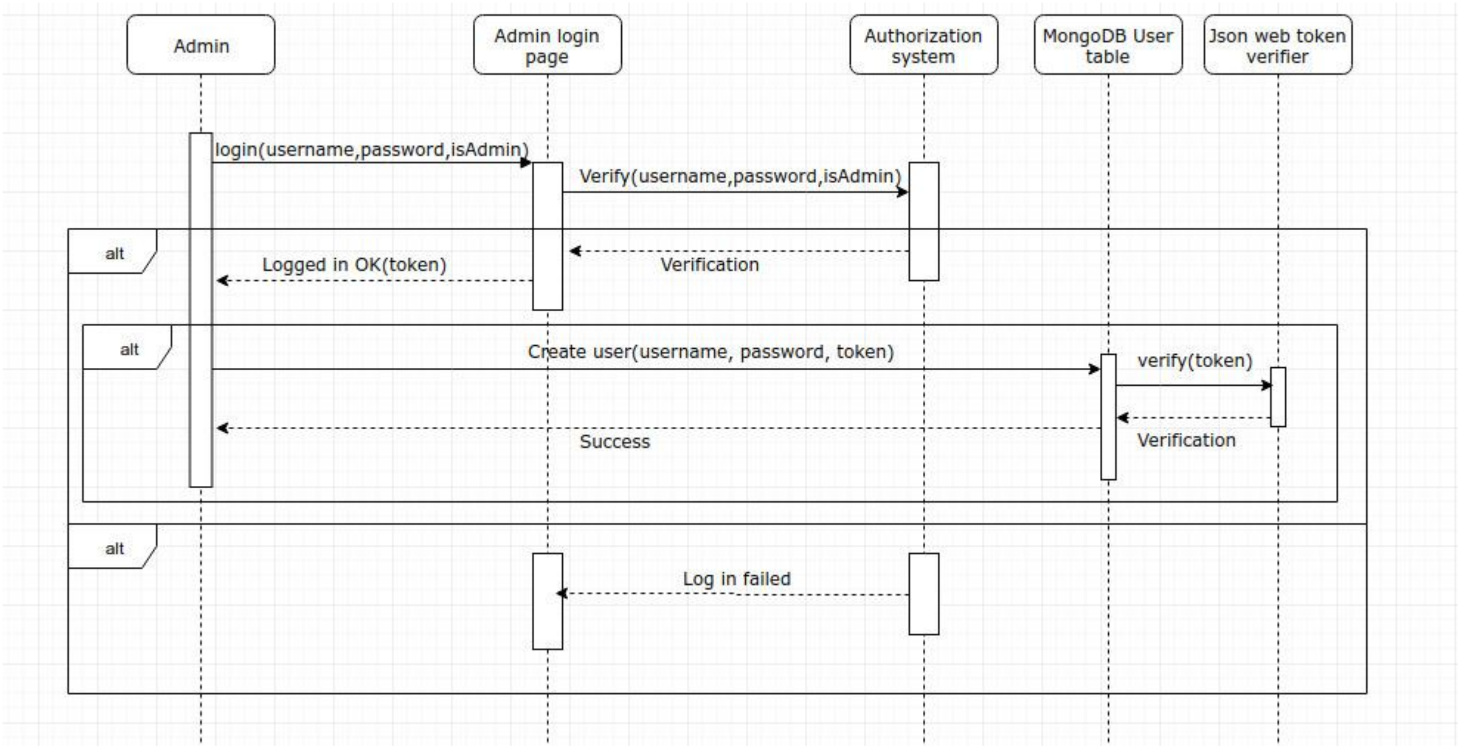

Figure 4: Sequence Diagram for administrator

When we release the project, there will be only one admin in the system, which can add other users and admins. Figure 4 illustrates how admin functionalities work by using create user function as an example. We have created two different login pages. One for admins and one for regular users. By using login page for admins, admin enters its username and password for verification. If authorization system can verify provided information, it gives a token back to admin. Now whenever admin tries to use any other function, he needs to send this token along with other information. We built such a system to increase security. In this diagram, we call create user function with username and password along with token. We create the user with given information only if our system verifies the provided token.

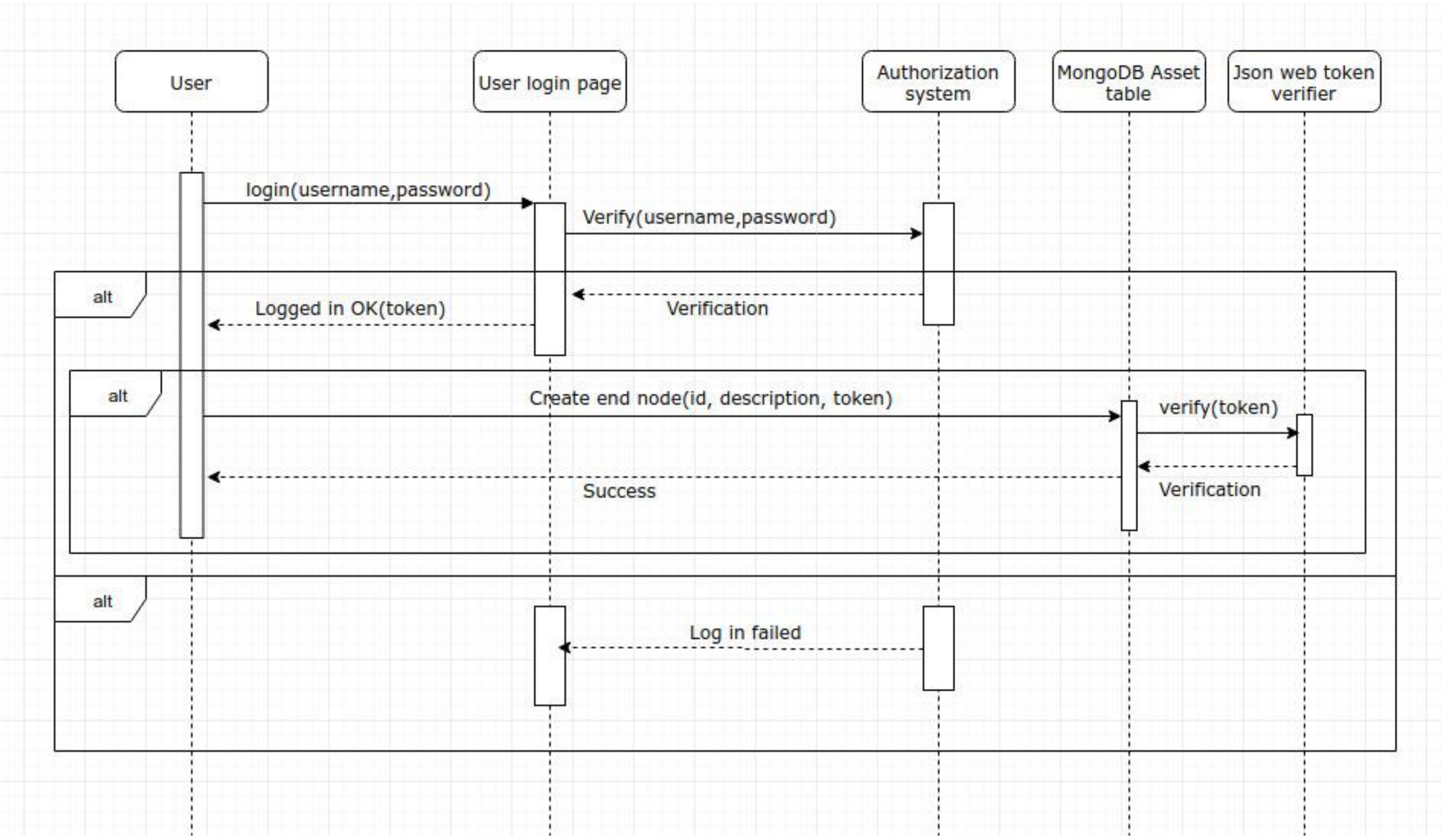

Figure 5: Sequence Diagram for user

User login mechanism is the same with the admin login that we explained previously. Only difference is, login process is made from different pages. After logging in, user is provided a token. After placing end nodes and gateways, user can create an entry for them in the database. Figure 5 illustrates how a user creates an end node entry. User needs to give id, description and initial position of the end node as parameters along with the token. Each end node should be assigned with unique id that will be used while creating the database entry. User can enter the asset name that end node is attached to as a description. Same as before, if our system can verify the provided token, the function call will return success.

4 - Alternative design options

- Among triangulation, trilateration and multilateration algorithms, we use multilateration since it gives more accurate data. Instead of signal strength it uses time difference of arrival.

- We use Cloud server instead of a remote server since it is more secure and has additional features that we may need to implement manually on virtual server.

- At the beginning of the project we were expecting to use a complete gateway but after checking prices of gateways, we decided to use Raspberry Pi along with a gateway model because it is a lot cheaper way.

- We use Leaflet map API for the dashboard instead of Google maps API. Because, Google maps API is not free completely, on the other hand, Leaflet is an open-source map.

- For front-end, we use React instead of classic HTML & CSS because it will make our job easier when we make mobile application with react Native.