UNIVERSITÉ Grenoble Alpes

Project report

Ultratrail

Ferrera Alexandre - Gallier Romane

Table of content:

Introduction:

The project's goal is to produce an app capable of transmitting and receiving data about the member of a group.

What makes the whole point of the application is the fact that it should be able to do that, even in areas without internet access.

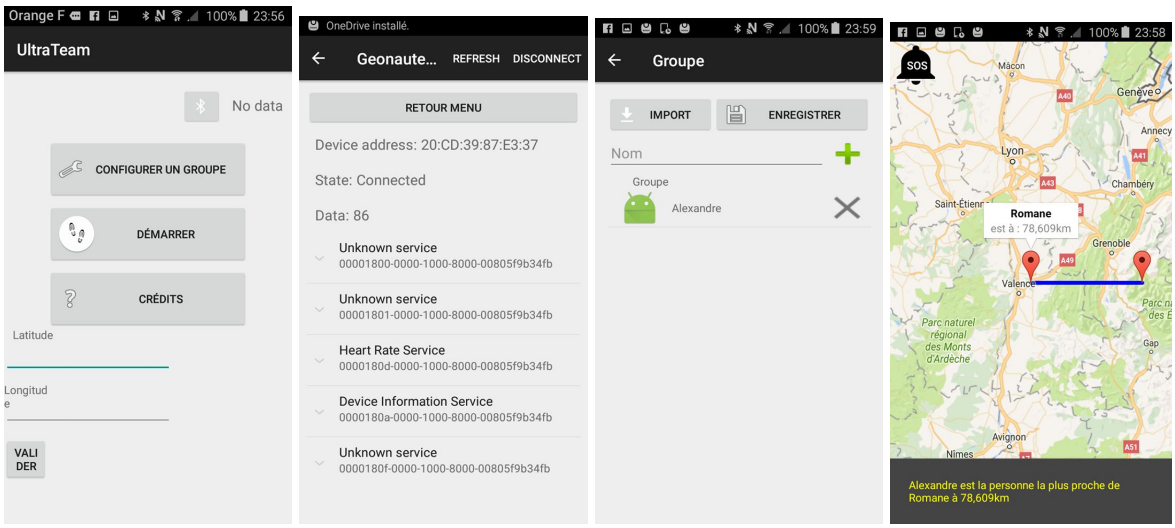
Out of internet network cover, the communication is made through the Lora network. This is made possible thanks to an antenna, using the RN2483 module.

When a user has an internet connection, he should be able to communicate with others by using a mosquitto message broker (using the MQTT protocol).

Transmitted information are provided by the phones sensors, (such as the position) and by low energy Bluetooth item's owned by the user.

One of our application's goals is to minimize our needs in energy, in order to maximize our autonomy. This should give the possibility to fully use our app, even in wild areas. To do this project, we developed an application in android java (with android studio).

Here is a quick overview of the application :



<u>Groupe:</u>

Thanks to the "Config Group" button, we can define the id of the users we wish to follow (get the data they send, such as their position).

Once we have added someone :

With internet access: We have data in real time, over the followed users (as long as they also have an internet connection too). This is made thanks to the mqtt connection (explained later)
Without internet access: If the user has the necessary equipment, he will be able to receive on his USB port the LoRa messages sent by other users (module or LoRamote).
As soon as data is received, the map is updated in order to show the current position of the members of the group. The map gives additional information about the users when we click on their marker.

Map's management:

We chose to use the API of google maps in order to display a map.

Thanks to it, users appear on the map as markers, giving access to additional information over them by tapping their marker. Currently, the displayed information are the nearest user and the distance between that separates you.

Bluetooth:

The app looks for a bluetooth connection. It is designed to connect to equipment that provide a heart rate. (For example a heart rate belt). The connection is handled by a service.

Class Personne:

This class represents a user. To do so, it contains its ID, its position, and the marker that will represent him on the map. (this class could possess other attributes such as an sos boolean, and the heart rate of the person).

Singleton ultraTeamApplication

UltraTeamApplication is a singleton that saves the group as a hashTable of persons and ids. It also contains the base's location, the mqtt client and an adapter that is used to display the users in the group configuration activity.

Mqtt_client: In order to use the mqtt service, there is a mqtt client that contains attributes to describe the server's uri and the client ID. This class manages Listeners on the mqtt connection events.

Lora:

Lora is a network that is designed to be used in the IOT. It provides a very wide network cover and can be used for free. However, in order to be reliable, it has some restrictions to respect.
It uses a very low frequency to provide messages for a minimum quantity of energy.

Use:

Lora's restriction:

We are not allowed to emit using Lora's network, for more than 1% of the time. The emission time is actually the time on air of the message, which means that we should send messages as small as possible, in order to wait the shortest time possible between 2 communications. In our current situation, we are allowed to send 1 message every 30 seconds. (source : https://docs.google.com/spreadsheets/d/1voGAtQAjC1qBmaVuP1ApNKs1ekgUjavHuVQIXyYSvNc/edit?usp=sharing)

Lora communication's principle:

-Reuse of open source code that handles communications through USB port, using a terminal.
A connection to an antenna is made through the USB port. A service is used to manage the reception of data.
-Use of Call back on data reception from USB port
-Every time the position has changed more than a given distance (since the last message sent). This way, we can avoid sending useless messages. Thanks to this, we can respect the Lora restrictions as long as the app is used by trailers.

We have to ways to communicate through LORA:

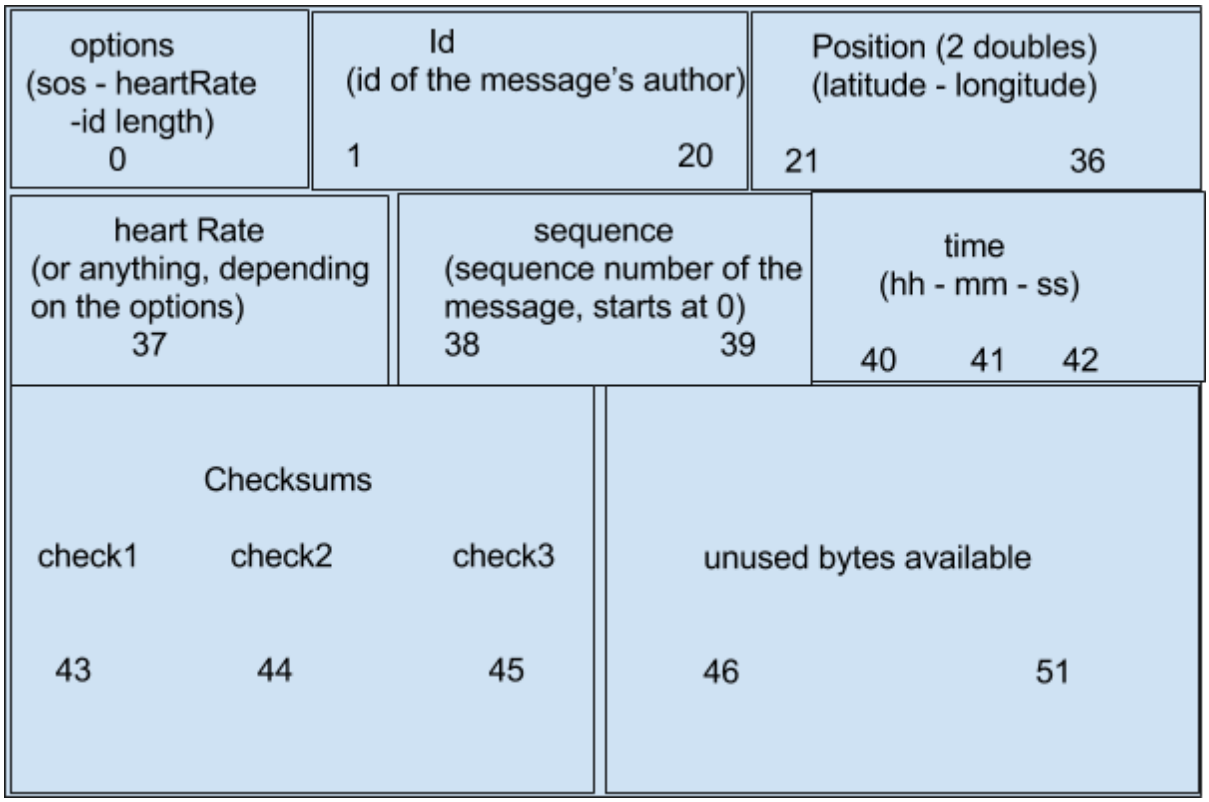
The first one is the module RN2483 which can emit but more importantly, can receive data sent through Lora. These data are transmitted to the USB port of the phone, thanks to an adapter. The phone can also use this port to ask the modem to send a message.
The messages are sent Broadcast.
When a message is received, the app analyse it.

The second way of communication is the LoRamote :

This device was flashed so it can send regularly many information such as the position, altitude, battery level, pressure, temperature, etc… Here, we are mostly interested by the position and altitude.
The messages are also crypted so we are the only ones that can read them.

Message format

Regarding the format of the messages sent with the modem RN2483 :



The options indicate if the message is a SOS and if it contains the heart rate of the sender. It also contains the size of the Id.
Here's how it's composed :
-First bit : SOS
-Second bit : Contains heart Rate
-third to seventh bit : length of the Id (which is the user's name)
Then there is the ID of the sender, a string that is a unique identifier of the user.
The position is composed of two doubles which represent the latitude and longitude of the senders.
The heart rate is coded with 1 byte. This field can be ignored depending on the options. We could add the possibility to replace use this field for another purpose, that would be defined by the options.
Then we send a sequence number of the message and the a time stamp.
We also use 3 cheksums to detect possible mistake.

Regarding the messages sent by the LoRamote, we rely on the bits 8 to 15 in order to retrieve the position and altitude. In order to see the full structure of the messages : see how the variable AppData is created on this page :
https://github.com/Lora-net/LoRaMac-node/blob/master/src/apps/LoRaMac/classA/LoRaMote/main.c

Way of creation of a message :
The attribute of a message are :
-the name (or id) (String)

-the gps coordinates (LatLng)
-the time (3 shorts : hour, min, sec)
-the heart rate (short)
-the options (byte)
-the sequence number (short)
-the transformed message : loraPayload (byte [])

A message can be generated from a personne. in this case the sequence number is 0 and the time is the current one. To do that, one can use the method :
new Message(Personne personne)

Then, the message can be transform into a byte array to be send with the method:
message.loraPayload();

Way of decoding a message :
To decode a message it is possible to simply use the constructor :
new Message (byte[] payload);
Once this constructor is used, the message can be analysed through getters.

Bluetooth:

Utilisation

We use the low energy bluetooth to connect to a heart rate belt through our application. This way, we can provide the user's heart rate to the members of the team.

android implementation

The bluetooth is used thanks to a service.
The application Bluetooth LeGatt was used to manage the connection.

Mqtt:

When a network such as 3G is available the application is able to communicate with other users through messages, using the Mqtt protocol. To do so, we implemented a service that used to be available on the github repository : https://github.com/eclipse/paho.mqtt.java

Protocole mqtt:

The users regularly publish their position while they can access the MQTT broker, which is running on the ovh server.

Message format:

The format used to send messages is the same the ones sent through Lora. We made this choice because even though these messages are limited in size, they contain all the necessary data about the user, and we already implemented a way to construct and analyze such messages.

Publish/subscribe principle:

The MQTT protocol is based on the publish/subscribe principle.

What this means is that a user can subscribe to a topic (for example : ultratrail/user_id). Once the user has successfully subscribed to this topic, he will be notified every time something is published on this same topic.

Every connected user can also publish a message on the topic of their choice.

In our case, if there are 2 users : userA and userB, they can simply subscribe to each others (on the topics : ultratrail/userA and ultratrail/userB), when userA will publish his position and whatever other information, userB will be notified of userA's published information.

Limits of the project

-current issues:

Currently, our app isn't based on user accounts. which means that there can be a duplication of the user's ID, which will be a problem when connecting to the broker.

The Loramote's messages aren't interpreted yet. The application isn't capable yet to bind the Loramote to the users and so to identify who the Loramote's messages belong to. This being said, the application is ready to decode the payload of these messages. (it can read the position and the altitude).

An internet connection is necessary to load the map. Which means that in some areas, the map should be loaded before starting the trail. This also means that the app should never be turned off during the trail in such areas.

We didn't clearly implement the notion of group yet. A user should add the id of every user he wants to follow.

The heart rate isn't available yet in the person's attributes. (Even though it is sent and received through the network). A few others features such as the sequence number, or the SOS encounter the same issue.

-Possible solutions:

In order to manage users' accounts, a server running with meteor code could store and manage data in a collection. In order to update this collection, the meteor app should be connected to the Broker through MQTT, and update users' information, as they are communicating with the Broker. A necessary condition to this issue is to find an API that would run the meteor part of the application.

In order to bind Loramotes with their user, we should find a way to simply get their mac address.

Users should have the possibility to store a map in their application so the internet connection is no longer required to load the map.

In order to manage the groups, we can simply rely on meteors collections. If meteor isn't implemented, the topics can also be easily manipulated to create groups. (separate ultratrail/users/user_id and ultratrail/groups/group_id, and then, subscribe to ultratrail/groups/group_id/# to collect every message published by the members of the group.)

Conclusion

What this project taught us:

This project taught us to use different popular technologies here are a few ones:
-android studio: We learnt to create an app and services, but also handle versions issues and implementing open source features by adapting the gradle file to the situation.
-Meteor: Even though we didn't implement any meteor code, we spent time learning to use it. We now are more familiar to javascript and know a way to develop app's efficiently in javascript.
-github: We worked as 2 groups of 2, on the same app, which forced us to use the branches/forks in order to merge our applications. We learnt to use git kraken
-LoRa: We learnt this efficient way to manage IOT.

Another important aspect that we improved during this project was our capacity to use open source code and application to benefit from their features without having to learn to use them. For example, we've had to flash the LoRamote with code using an old version of the IDE keil, we used an example of paho mqtt that wasn't functional because of it's gradle, and in general, we learnt to be autonomous when learning to use a technology from scratch.

What difficulties we encountered:

What made the project beneficial to us, also provoked a lot of difficulties that slowed our progression down. We encountered difficulties such as :

-Using android studio and creating an application. We had to create a complicated app, using many different features even though we never made any app by ourselves before this project. In addition to that, we didn't have computers fitting with the requirements of android studio (very long build, sometimes resulting in a failure due to a lack of RAM, or a crash of the computer)
-Following a pattern that used many aspects that we didn't know. (message broker, mqtt protocol, meteor, LoRa)
-Working as 2 separated groups on the same application, which led to troubles with sharing our work

What are the good points of our app:

The aspects that we managed to achieve within the app are the following :
-Establishing a bluetooth connection (low energy)
-Connecting and subscribing to a broker's topic with the MQTT protocol
-Subscribing to a list of users
-Communicating a position to the other users (the ones that are subscribed to you)
-Displaying a map with markers placed at the position of the users you are subscribed to.
-Updating the map with the positions given by the others users.