

UltraTeam

Project Report

MOLION Enzo, VALETTE Léo

Tutor: DONSEZ Didier

Teachers: DONSEZ Didier, PALIX Nicolas, RICHARD Olivier

Table of contents

I) Introduction	2
A) Project presentation	2
B) Primary audiences of this document	2
II) Technical explanations	3
A) Used technologies	3
LoRa	3
BLE	3
Ionic framework	3
B) Architecture	4
C) UltraTeam: an application	5
Frontend	5
Backend	7
D) UltraTeam: a protocol	8
LoRa packet management	8
LoRa packet content	9
III) Organisation	11
IV) Conclusion	12
A) Unfinished tasks	12
BLE communication	12
Actual calls to the server API	12
Deporting AES encryption to ESP32	12
B) Issues	12
Multiple instances of Ionic pages	12
C) Possible improvements	13
Offline map loading	13
Add protocol version number in packets	13
Reduction of LoRa packet size	13
Sos via ESP button	14
Better map markers	14
V) Bibliography	15
Report notes	15
Further reading	16

I) Introduction

A) Project presentation

This project consists of an implementation of a solution that should allow a hikers team to localize each other in real time in zones with no-cellular data (further referred to as “white zones”)[1]. The current version of UltraTeam¹ isn’t the first one but restarted this project from scratch.

Such solution should allow new hikers to be really easily added to the current hike (further referred to as “course” as applications of UltraTeam are not bound to hiking). As stated it should be able to operate without access to cellular data hence should mainly rely on LoRa connection, but should be able to connect via cellular network to the server.

Therefore, this project consists of:

- A BLE & LoRa decentralized (“full mesh”) protocol conception and implementation,
- A web application development.

This project is developed in synergy with another UltraTeam 2018 project[5].

B) Primary audiences of this document

As one might understand by reading the current report, the whole project has not been completed fully, thus this project report is not only aimed at professors for the purpose of its supervision and scoring but also for potential continuators of UltraTeam.

Indeed, this year’s version of UltraTeam was built as a baseline for future years development.

Curious users of UltraTeam application might as well want to learn how but also on which principles and technologies it was originally built.

Thus, this project is written such that anyone with minimal IT knowledge can understand it.

¹ See [Bibliography](#) for further reading about previous years’ UltraTeam versions^{[2][3][4]}.

II) Technical explanations

A) Used technologies

LoRa

The LoRa technology is the way for members of a hike to exchange data.

LoRa is a patented spread-spectrum radio modulation[6]. It allows data to be sent on long range. In order to maximize the emission efficiency and frequency, the size of the sent packets should be as small as possible.

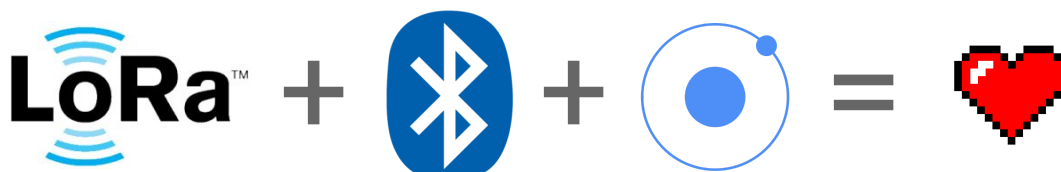
The boards on which was implemented the described protocol are ESP32 equipped with a LoRa module, from Heltec[7]. Its development was based on Arduino language which is close to C++, with the Heltec's LoRa library[8].

BLE

The Bluetooth Low Energy (further referred to as BLE) is used here in order to limit the energy consumed by the cellphone and the embedded board. BLE connection was developed using the ESP-IDF IDE, using C language.

Ionic framework

Ionic is an Angular framework to ease the development of hybrid multiplatform mobile application (relying on web technologies such as HTML5, CSS and Sass)[9]². It is based on Typescript that is a typed superset of JavaScript which compiles into plain JavaScript³.

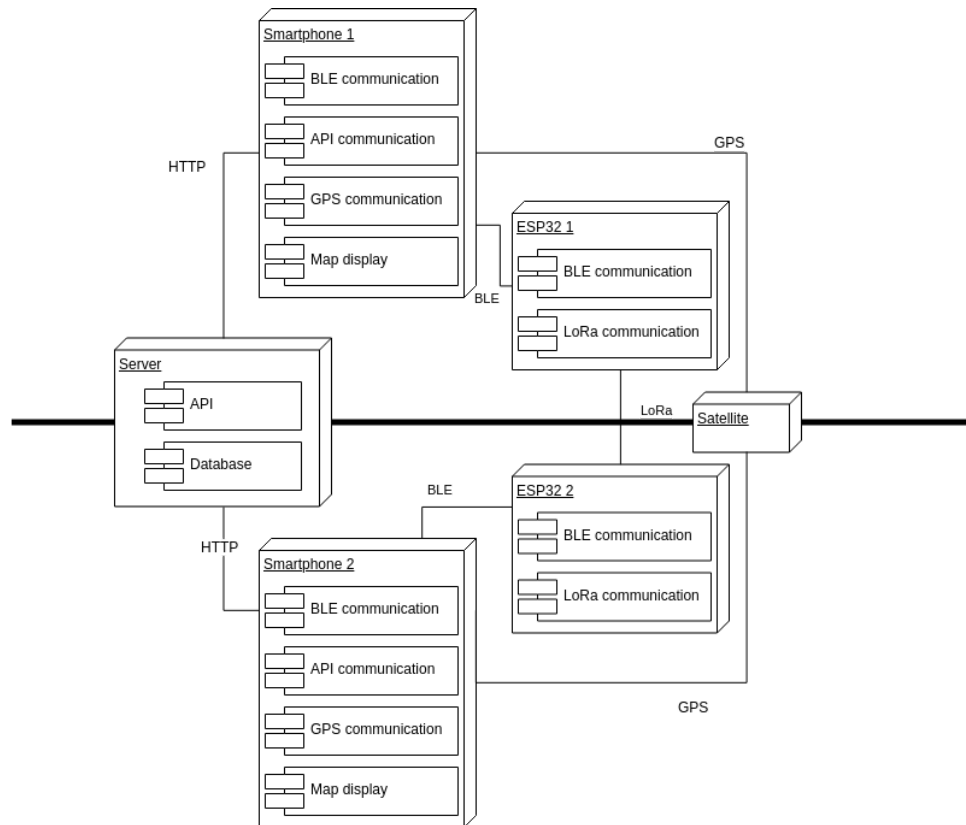


² This framework was chosen because of its simplicity, its ability to produce beautiful applications but also because it was part of the MMDO course in RICM4. This course's teacher, Sebastien Pittion has been an invaluable resource to us to master this framework.

³ <https://www.typescriptlang.org/>

B) Architecture

The project has the following architecture:



Several entities are shown on the diagram: a server, a smartphone, an ESP32 and a satellite (another smartphone and ESP32 are shown to illustrate that there are quite a few of them while there is only one server and satellite system on the project's architecture).

A server was deployed by the other UltraTeam project which communicates with the smartphone via HTTP and more precisely through an API. This smartphone communicates to its user's ESP32 via BLE and gets its geolocation via the Global Positioning System. ESP32s communicate between them via LoRa.

The reader is here advised to keep in mind that most of the time the HTTP connection which is carried through by cellular network is not available. Hence, the architecture is simpler: it might be seen as a full mesh network smartphones equipped with two antennas: a GPS one and a LoRa one, the latter being actually deported (via BLE) in the ESP32.

C) UltraTeam: an application

The application should be able to display a map with a marker on every hiker, given the received positions via BLE or API packets.

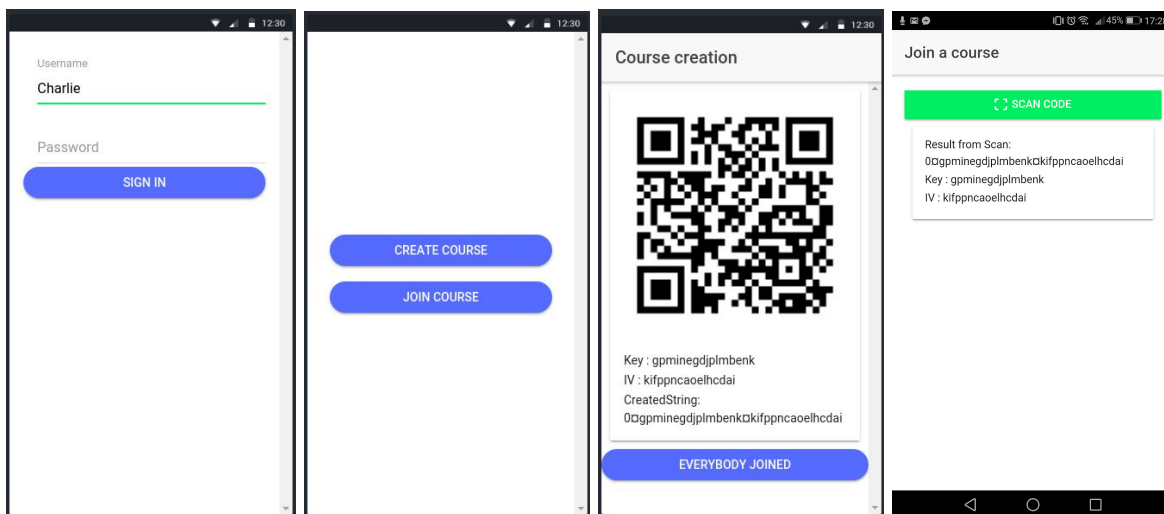
It should also provide the ability for a user to signal that he is in a distress situation.

Frontend

The aim of this the application development part was to produce a simple and user-friendly solution for non technical experts. Ionic helped this development by its already widespread visual components.

The application interface is thus quite standard. It can be divided in two phases: a initialisation one and a normal one.

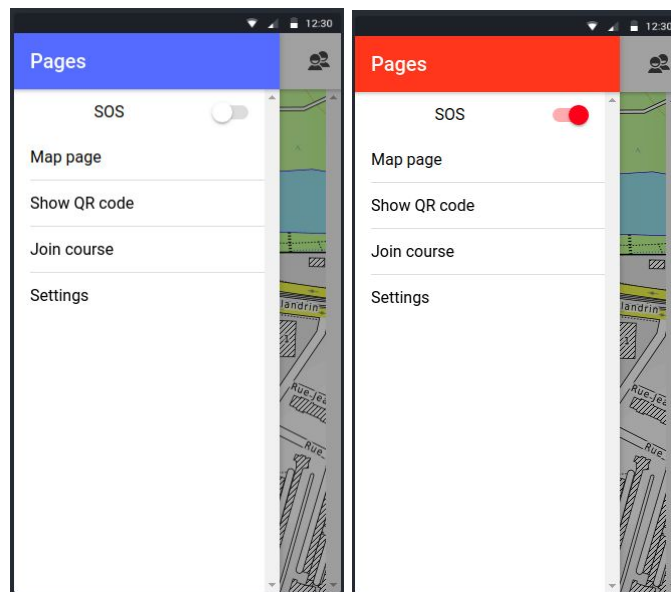
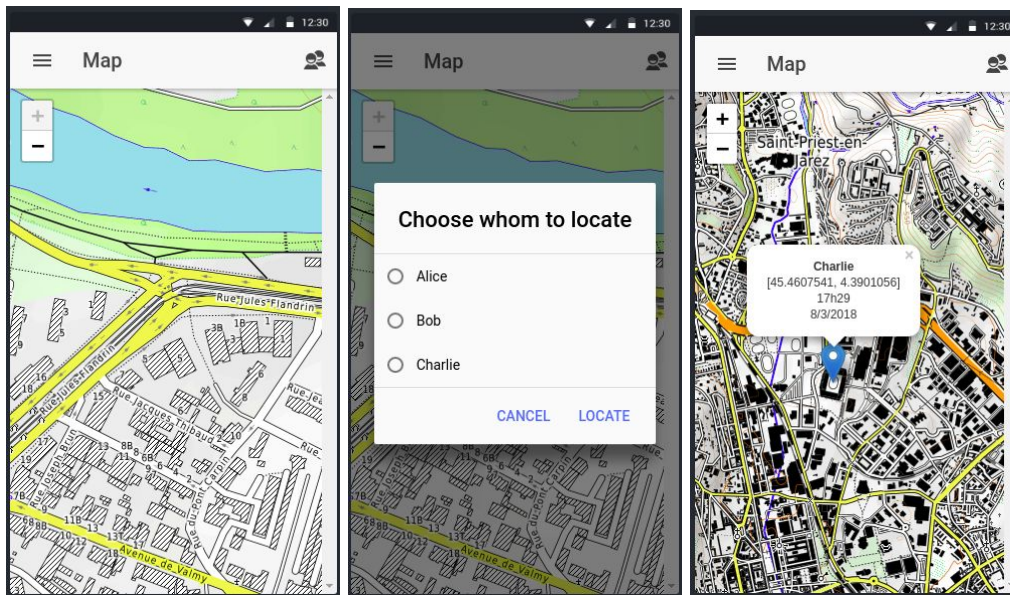
Let us depict the first : after a login page, the user is asked to decide if he wants to join a course or to create a new one. Depending on its choice, the user will be faced with a QR code or a camera page. Once the joining users all flashed the QR code, the creating user clicks the “Everybody joined” button and the initialisation is over.



Such an initialisation phase was coded thanks to Ionic standard Ionic input components (input, button,...) but also the ngx-qr-code-2 Angular component library for the QR Code generating and scanning[10].

The second phase is the actual use of the application. It displays a map with markers on other hikers, with the ability to select a user to center on via a list.

It also allows the user to open a menu to toggle the distress state signal on and off as well as navigating to pages such as Settings or QR code displaying.

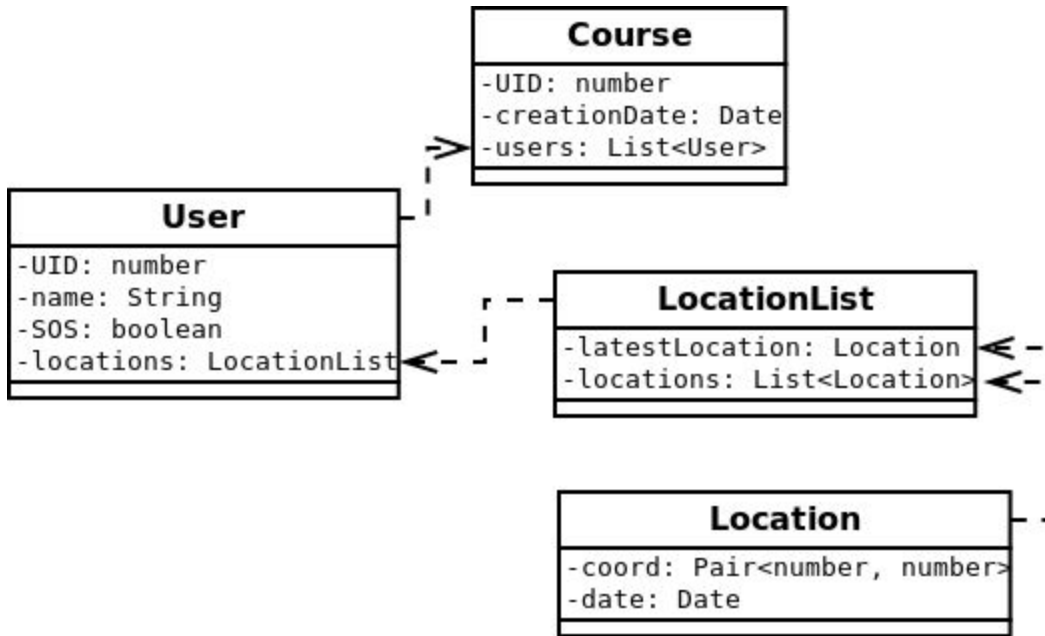


This phase was coded thank to Ionic components (for the hikers list and menu) and mostly thanks to leaflet, an open-source JavaScript library for mobile-friendly interactive maps^[11].

Backend

The current section will depict how data is structured in the application such that it can produce the previously described frontend.

The data structure is composed as follows:



One might be surprised by its simplicity but should then remember that a more complex database is stored in the server. Here a light structure is a plus and is sufficient as the only info to be shared is the location of users.

In every page is injected a provider which allows the page to access the cross-pages data. This data is composed of:

- A Course which is the hike currently taking place,
- A User which is the hiker owner of the smartphone,
- A boolean which is the distress state of the owner of the smartphone.

Another interesting aspect of backend is the AES-128 encryption. Indeed, as ESP32 to ESP32 are carried out by LoRa which is based on radio, data can not be sent unencrypted.

Thus, an encryption process has been developed. It works as such:

The course creator generates an AES object⁴, which will allow data encryption and decryption. Such AES is parameterized by a Key and an IV which are 128 bits long numbers.

As the AES is created, the creator asks the server for a course unique identifier via the API. Then the creator displays a QRCode containing the course UID, AES Key and IV, such that other users joining the course can inform the server that they engage in said course but also can encrypt and decrypt course's LoRa messages.

A sharp-eyed reader may have noticed that using this process, AES encryption key are not transmitted to the server, avoiding a massive security flaw. The only way for someone to get them would be to flash the QR code without the creator seeing it.

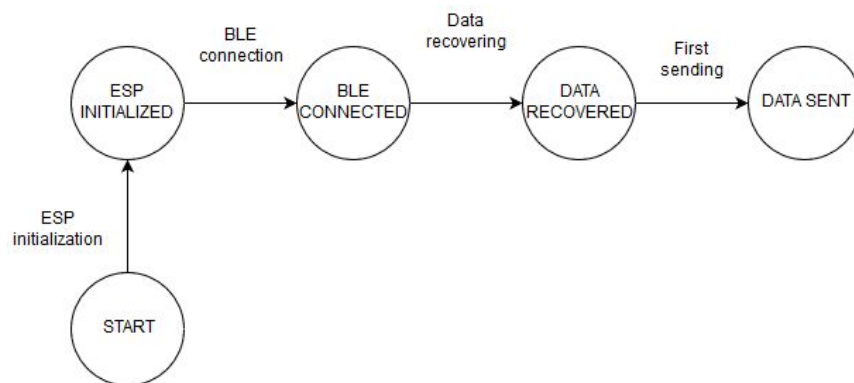
D) UltraTeam: a protocol

LoRa packet management

The LoRa packet management is divided into two sections:

The setup phase, which starts when the ESP is powered on. During this step, the cellphone must send its data to the ESP via BLE.

Firstly the ESP is configured, then it should be connected to the cellphone to recover data. Finally it sends its data to the world !



The second phase is the loop phase. This is what will be done until the ESP is powered off.

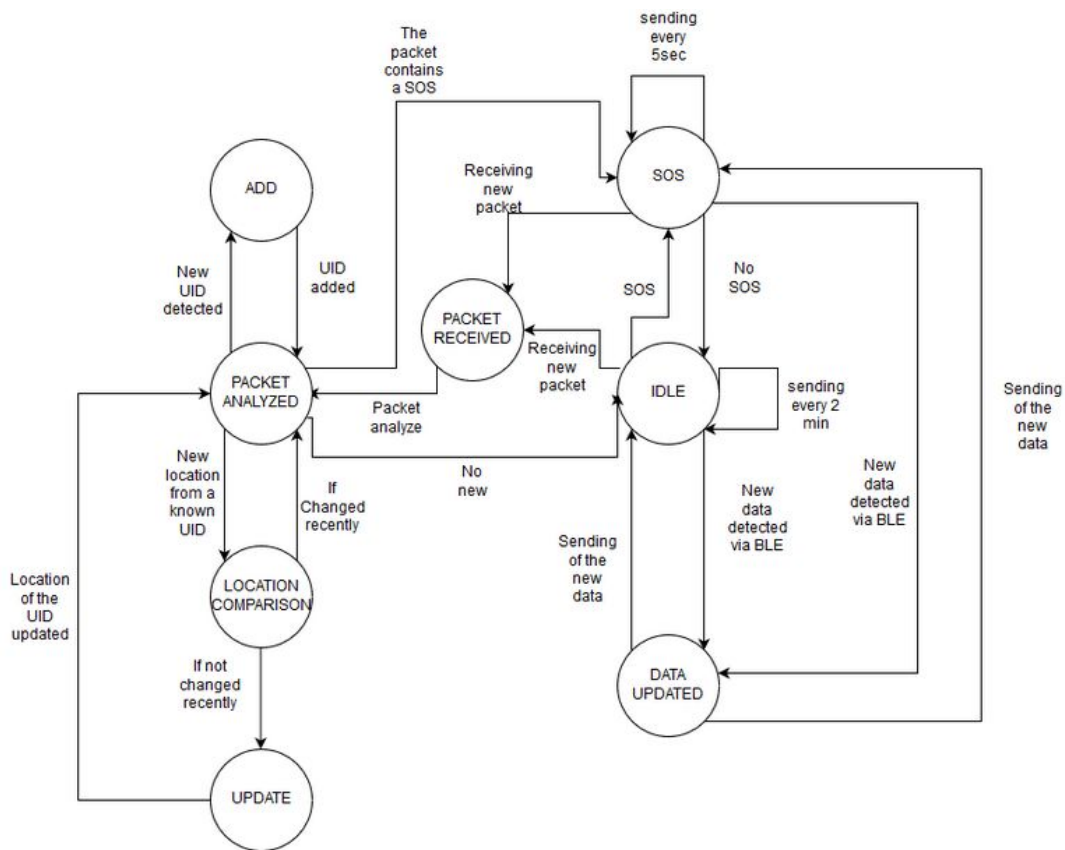
The ESP32 send packets every two minutes, or five seconds if a SOS is sent by any

⁴ Which is defined by the *aes-js* JavaScript library[12].

member of the course (the content of the packets is explained in the next section).

The loop phase is described on the automata below. The base statement is IDLE (where the ESP is communicating every two minutes). It goes out of this statement by means of: receiving data, switching to SOS statement or updating location.

If a packet is received, it is analyzed in order to allow the ESP to extend or update its database.



A whole protocol specification document^[13] has been written and is available online on the AIR page of the project. The reader is advised to look at this paper for a better understanding of the management of received LoRa packets.

LoRa packet content

As stated before, the packets are sent every two minutes (except for SOS statement). The length of a packet is variable. It depends on what has been recently updated in local database. It can be:

- A new member of the course (identified by its UID),
- A new location from a member of the course
- The variation on the SOS value of a member of the course.

So, in a packet, there are information about the member who is sending data, about users that have send changes and about, obviously, users that are on a distress situation. A packet is composed of subpackets with each user's data.

Each subpacket looks like this:

User ID (1 byte)	Geolocation (8 bytes)	SOS Flag (1 byte)
---------------------	-----------------------	----------------------

III) Organisation

A “standard” reader might not be interested by this section as it is about project development organisation. In this case, such reader can directly jump to the [Conclusion](#) section of this paper.

This section will depict the way the project went from nothing but the theoretical instructions (as it started from scratch) to its current state. It will be written using the first person plural and will depict who did what and how it happened.

We used a “pseudo-scrum” method. Despite the fact that we did not know about this Agile methodology[14], we applied by ourself a lot of its principles:

- We splitted the project into huge tasks. Then as we were moving forward those tasks splitted into smaller tasks, short enough to be achieved in one or two weeks (one may want to call it “sprints”),
- We used Todoist⁵ to keep track of those tasks, assign them a “length - difficulty” score and assign it to the one responsible for it,
- We made “pseudo-stand up meeting” by informing each other of what we previously did et what we were about to do, and this before every session.

This methodology was really efficient at the beginning of the project, during the specification phase. Then, we spread the tasks and as a result, we worked on separate tasks for the whole project : Léo worked on BLE communication and then on an implementation while Enzo worked on the application development.

As we went further in the specification and implementation, we realized that the project could not be correctly completed on a single year, that it would require more work. Thus, we decided to build a solid baseline for future continuators of the project. Then the documentation of every task was reinforced.

At every moment, each member of the project knew what the other was doing and how far in this achievement of the task he was. Every decision was taken after a “whole team” meeting.

⁵ <https://en.todoist.com/>

IV) Conclusion

A) Unfinished tasks

BLE communication

The pairing has been developed, using example from ESP - IDF. But no data was exchanged. Many examples of BLE are findable on the dedicated branch of the UltraTeamMV repository^[15].

Actual calls to the server API

As stated, this project was developed in synergy with another UltraTeam 2018 project. The software interface was defined by the two group who made the agreement of having an API module implemented by Gros-D-Aillon, H. and Terrier, B. while Molion, E. and Valette, L. called this module's methods in the application code.

Deporting AES encryption to ESP32

Currently, AES encryption is possible via TypeScript methods in smartphone application's code. It should be done by the ESP32 (otherwise it couldn't be possible for it to determine what to do).

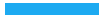
B) Issues

This section was made available by the Issues system of GitHub to future contributors on the public repository, issues section⁶.

Multiple instances of Ionic pages

Regarding the Ionic web app, the main issue is that when a page is popped out of the navigation stack (via `pop()` or `setRoot(page)` methods), the instance of the page is

⁶ <https://github.com/ultratrail/UltraTeamMV/issues>



deleted and one another is created when pushing it in the navigation stack. As a consequence, attributes of pages are reinitialised and that's not the wanted behavior.

A simple fix exists:

Every data that needs to be kept between two views of a page has to be saved in an injectable provider. This injectable then must be accessed to get the needed page info.

For example, for QRCodeGenerationPage: create an injectable which as an AES from which access to recreate the QRCode to display to another user.

C) Possible improvements

Offline map loading

One of the major needed improvement would be to be able to preload[\[16\]](#) maps for a hike as it is currently needed to have a cellular connection to see the map (!).

Add protocol version number in packets

The protocol is bound to be enhanced, thus a version number should be included in LoRa packet in order for the ESP32 to be able to correctly interpret sent data. This is really simple to implement, and might be the first enhancement to be.

Reduction of LoRa packet size

Currently, the size of a packet is absolutely not optimized with 10 bytes per user-info sent. It can be reduced from 10 bytes to 6 bytes as shown below:

Version number	User ID	User ID	User ID	User ID	User ID	User ID	User ID
User ID	User ID	User ID	User ID	User ID	User ID	User ID	SOS Flag
Location	Location	Location	Location	Location	Location	Location	Location
Location	Location	Location	Location	Location	Location	Location	Location
Location	Location	Location	Location	Location	Location	Location	Location
Location	Location	Location	Location	Location	Location	Location	Location

(every cell is 1 byte)

This solution also has the benefit to increase the maximum number of participants to a course from 256 to 16384 (= 2^{14}).

It relying on the fact that the location can be encoded on 4 bytes[17] against the current 8 bytes.

Sos via ESP button

A physical button is available on ESP32 boards. It might be interesting for a user to be able to signal a distress state via this button.

Better map markers

When a lot of hikers are in the same zone, a “one marker per hiker” can make the map really messy, a marker cluster system[18] might be better for readability.

V) Bibliography

Report notes

[1] Molion, E., Valette, L. April 2018. *RICM4 2017 2018 - UltraTeamMV*, in *AIR*. [Online] Available: https://air.imag.fr/index.php/RICM4_2017_2018_-_UltraTeamMV [09/04/2018].

[2] Donsez, D., Palix, N. February 2017. *UltraTeam*, in *AIR*. [Online] Available: <https://air.imag.fr/index.php/UltraTeam> [09/04/2018].

[3] Geourjon, A., Rouquier, C. April 2017. *Projets-2016-2017-UltraTeamBest*, in *AIR*. [Online] Available: <https://air.imag.fr/index.php/Projets-2016-2017-UltraTeamBest> [09/04/2018].

[4] Donsez, D., Ferrera, A., Gallier, R. April 2017. *Projets-2016-2017-UltraTeam*, in *AIR*. [Online] Available: <https://air.imag.fr/index.php/Projets-2016-2017-UltraTeam> [09/04/2018].

[5] Gros-D-Aillon, H., Terrier, B. April 2018. *RICM4 2017 2018 - UltraTeam 7.1*, in *AIR*. [Online] Available: https://air.imag.fr/index.php/RICM4_2017_2018_-_UltraTeam_7.1 [09/04/2018].

[6] Prajzler, V. August 2015. *LoRa, LoRaWAN and LORIoT.io* [Online] Available: <https://www.loriot.io/lorawan.html> [09/04/2018].

[7] supamoe25. September 2017. *ESP32 Board - WiFi LoRa 32*. [Online] Available: <https://hackaday.io/project/26991-esp32-board-wifi-lora-32> [09/04/2018].

[8] Heltec-Aaron-Lee, nishushu. *WiFi_Kit_series*, GitHub repository. [Online] Available: https://github.com/Heltec-Aaron-Lee/WiFi_Kit_series/blob/master/esp32/libraries/LoRa/APl.md [09/04/2018]

[9] Drifty Co. 2016. *About Ionic*. [Online] Available: <https://ionicframework.com/docs/v1/overview/#about> [09/04/2018]

[10] Grimm, S. August 2017. *Ionic QR Code Generator & Reader*. [Online] Available: <https://ionicacademy.com/ionic-qr-code-generator-reader/> [09/04/2018].

[11] Agafonkin, V. January 2018. *Leaflet - a JavaScript library for interactive maps*. [Online] Available: <http://leafletjs.com/> [09/04/2018]

-
- [12] ricmoo. October 2017. *aej-js*, GitHub repository. [Online] Available: <https://github.com/ricmoo/aes-js> [09/04/2018]
- [13] Molion, E. Valette, L. *ESP32-Smartphone via LoRa & BLE communication protocol Specification*. [Online] Available: <https://air.imag.fr/images/d/d8/UltraTeamMVProtocolSpecification.pdf> [09/04/2018]
- [14] Schwaber, K. February 2004. *Agile Project Management with Scrum*. Microsoft Press. ISBN 978-0-7356-1993-7.
- [15] Molion, E., Valette, L. April 2018. *UltraTeamMV*, GitHub repository. [Online] Available: <https://github.com/ultratrail/UltraTeamMV> [09/04/2018]
- [16] ismyrnow. October 2017. *Leaflet.functionaltilelayer*, GitHub repository. [Online] Available: <https://github.com/ismyrnow/Leaflet.functionaltilelayer> [09/04/2018]
- [17] arjanvanb. January 2014. *Best practices when sending GPS location data*. [Online] Available: <https://www.thethingsnetwork.org/forum/t/best-practices-when-sending-gps-location-data/1242> [09/04/2018]
- [18] ghybs et al. February 2018. *Leaflet.markercluster*, GitHub repository. [Online] Available: <https://github.com/Leaflet/Leaflet.markercluster> [09/04/2018]

Further reading

- Miller, R. March 2016. *LoRa Security Building a Secure LoRa Solution*. [Online] Available: <https://labs.mwrinfosecurity.com/assets/BlogFiles/mwri-LoRa-security-guide-1.2-2016-03-22.pdf> [09/04/2018]
- Jaguar Network. April 2017. *IoT : LoRaWan, ABP and OTAA connections..* [Online] Available: <https://www.jaguar-network.com/en/news/lorawan-in-a-nutshell-2-internet-of-things-iot/> [09/04/2018]