

ISEN

ALL IS DIGITAL!

BREST



M1

**Institut Supérieur de l'Électronique et du
Numérique**

Tél. : +33 (0)2.98.03.84.00

Fax : +33 (0)2.98.03.84.10

20, rue Cuirassé Bretagne

CS 42807 - 29228 BREST Cedex 2

FRANCE

**Année scolaire 2017/2018
Stage d'application**

« LPWAN applied to marine technologies »

effectué du 18/06/18 au 18/09/18

à

« UPC Vilanova – Avd. Victor Balaguer, 1 – 08800, VILANOVA I LA GELTRU –
Barcelona »

Relecture du rapport par le tuteur industriel : (date, signature, cachet)



UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Encadré par : M. Joaquin Del Rio, +34 (0)6.09.92.69.66

Correspondant ISEN Brest : Christophe Vignaud

Etudiant : Julien Peudennier

Acknowledgements

First of all, I would like to thank Mr. Joaquin Del Rio, my internship supervisor, for teaching me much knowledge and giving me lots of advices. I am grateful to him for his confidence too. I would also like to thank him for his care, his support, his guidance and his constant encouragement. He has provided a huge amount of his precious time for me.

I would like to express my sincere gratitude to Mr. Matias Carandell, an electronic research engineer who works in the university. He helped me in my project and answered all my questions.

I am also thankful to Mr. Kaloyan Ganchev, a Bulgarian student who worked in the university at the beginning of my internship. He explained me lots of things about Sigfox communication and about his work. His programs helped me in my personal project.

I would like to thank Mrs. Nuria Martinez Rovira and Mrs. Débora Prieto who work in the International Office of the university. They helped me in the administrative tasks before arriving in Spain and when I was there.

Finally, I would like to thank all the employees for their host. They contributed to make this internship a great experience for me.

Table of contents

List of figures	4
1. Introduction	5
2. The university	6
2.1. SARTI Centre	6
2.2. Projects	7
3. Context	8
3.1. Existing work	8
3.2. Initial request	9
3.3. Similar technologies in the market	11
4. Technologies and tools	13
4.1. LPWAN networks	13
4.2. Sigfox	14
4.3. TD1205P module	15
4.4. EFM32 Tiny Gecko	16
4.5. J-Link EDU	16
5. Technical development	17
5.1. Fabrication of an electric cable (for the J-Link Device)	17
5.2. Equipment	20
5.3. Tutorial	21
5.4. Application and tests	22
5.4.1. General presentation	22
5.4.2. GPS Mode	24
5.4.3. Interval T1	25
5.4.5. Timeout	28
5.4.6. Type of message	30
5.5 Additional work	32
6. Project management	34
7. Conclusion	35
8. References	36
9. Summary	38

List of figures

Figure 1 - SARTI logo	6
Figure 2 - OBSEA underwater observatory	7
Figure 3 - Result of the program which retrieves x, y and z data	8
Figure 4 - Result of the program which detects accelerometer events	9
Figure 5 - Ocean drifter	9
Figure 6 - Example of device's track on a map	10
Figure 7 - Chipfox GPS Tracker	11
Figure 8 - Comparison of GPS/GSM with GPS/IoT technology	11
Figure 9 - Oyster LoRaWAN	12
Figure 10 - Some LPWAN networks	13
Figure 11 - Sigfox logo	14
Figure 12 - TD1205P: top view	15
Figure 13 - TD1205P: bottom view	15
Figure 14 - EFM32 Tiny Gecko	16
Figure 15 - J-Link EDU	16
Figure 16 - TD1205P Pin Diagram	17
Figure 17 - Pinout for JTAG (J-Link connector)	18
Figure 18 - Connections between TD1205P and J-Link EDU	18
Figure 19 - TTL-232RG cable	18
Figure 20 - Connections between TD1205P and J-Link EDU	19
Figure 21 - Electric cable for using J-Link device	19
Figure 22 - Parameters for using J-Flash	20
Figure 23 - Possible issue with J-Flash	20
Figure 24 - Uploading a program with the EFM-32 Tiny Gecko	21
Figure 25 - Eclipse error in the step 2 of the tutorial	21
Figure 26 - Sigfox backend	22
Figure 27 - Example of message sent by the device	23
Figure 28 - GPS Power Modes	24
Figure 29 - Exemple of command to change GPS power mode	25
Figure 30 - Command to see the current interval T1	25
Figure 31 - Command for changing the interval T1	26
Figure 32 - Sigfox messages when T1 = 3600 s and movement	26
Figure 33 - Sigfox messages when T1 = 600 s and movement	26
Figure 34 - Command to see the current interval T2	27
Figure 35 - Command for changing the interval T2	27
Figure 36 - Sigfox messages when T1 = 600 s, T2 = 1800 s and no movement	28
Figure 37 - Command to see the current timeout	28
Figure 38 - Command for changing the timeout	29
Figure 39 - Sigfox messages when timeout expires before finding GPS coordinates	29
Figure 40 - Command the see the current type of message sent (no movement)	30
Figure 41 - Command for changing the type of message sent (no movement)	30
Figure 42 - Sigfox messages sent when the type of message is '0' (no movement)	31
Figure 43 - Sigfox messages sent when the type of message is '1' (no movement)	31
Figure 44 - Gantt chart	34

1. Introduction

In this report, I will explain what I did during my internship which took place in the University Polytechnic of Catalonia (UPC). I was in the UPC SARTI centre which is a Technological Development Centre of Remote Acquisition and Data Processing Systems.

Engineers who are in the SARTI centre work on different projects linked to marine and terrestrial technologies, to the automation of complex measurement systems, to electronic design of remote data-acquisition systems and to digital signal processing. They are focused on developing environmental instruments and sensors for industrial and scientific applications.

The subject of the project was linked to these fields. The aim was to deploy a system in the sea, into a buoy for example, which will be able to send GPS data and other information on a specific network. The main constraint is to think about saving power energy, because the device will not be accessible every time. This system has to send data more regularly if it detects movement.

Tasks were focused on the development and the programming of a device using low-power wide-area network (LPWAN). A preliminary work was done in order to know which LPWAN networks and modules are available on the market. The aim of this work was to learn things about these technologies.

After that, the technical work began with the use of the TD1205P module from TDNEXT and the Sigfox network. The TD1205P module integrates GPS, accelerometer, temperature sensor and serial communication. This report will describe the development of a low power application based on this module. The work included firmware development and also some hardware development. The final application is a low power GPS tracker with movement detection.

The report is organized as follows. Chapter 2 will introduce a description about the UPC, and especially about the SARTI centre. Then, Chapter 3 will explain the context of the internship, in particular the existing work and the initial request. Chapter 4 will describe more specifically the technologies and the tools I used during this internship. Chapter 5 will explain the work which was realized, and Chapter 6 will mention the project management. Finally, the conclusion will be given in Chapter 7.

2. The university

In the UPC, there are 20 Research centres which are located on four UPC campuses in Barcelona, Terrassa and Vilanova i la Geltru.

2.1. SARTI Centre

SARTI centre is one of these 20 centres. In 1999, the CIDEM (Information Centre and Business Development), the CIRIT (Interdepartment Commission for Research and Innovation) and UPC signed an agreement for collaboration in Catalonia. One of the specific aims was to deploy a Technology Development Centre of Remote Acquisition and Data Processing (SARTI). This centre was created in 2000, in the Technological Centre of Vilanova i la Geltru.



Figure 1 - SARTI logo

The SARTI centre is formed by a multidisciplinary team. This team includes members from different departments of the UPC. The SARTI centre is associated with the Institute of Earth Sciences Jaume Almera, the Institute of Marine Sciences and Marine Technology Unit of the National Research Council (CSIC). The thematic scope of performances of the SARTI centre is in the development of environmental sensors and instrumentation for industrial and scientific applications. Seventeen persons work in the SARTI centre.

SARTI has as main purpose the scientific and technological development of equipment and systems for remote data acquisition, emphasizing on virtual and oceanographic instrumentation. The development of equipment and systems is done using the latest techniques in electronic design and includes simulation methods and statistical analysis. SARTI transfers these advanced technologies to industry, turning on scientific knowledge of excellence for society.

The main objectives of the SARTI Technological Development Centre are:

1. Technology transfer oriented to entrepreneurs.
2. Training in advanced instrumentation control techniques.
3. Design and implementation of distributed information systems for industrial and environmental process control.
4. Counseling, studies and services providers to businesses and governmental institutions.

To achieve these objectives, the SARTI Technological Development Centre establishes a partnership with some companies like National Instruments for example.

2.2. Projects

There are many projects in the SARTI centre. Many of them are closely tied to the Expandable Seafloor Observatory (OBSEA). The OBSEA underwater observatory is connected with 4 kilometres of cable to the coast of Vilanova i la Geltru and placed at a depth of 20 metres in a fishing protected area.

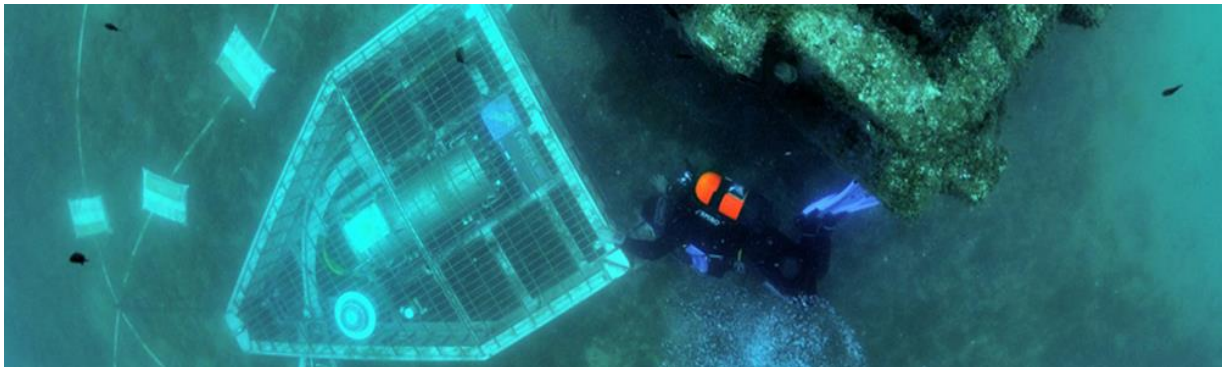


Figure 2 - OBSEA underwater observatory

Seafloor observatories can acquire data with great resolution uninterruptedly during long time periods. With this information, scientific community is able to analyse as well annual tendencies as singular events. It is possible to observe in real time multiple parameters of the marine environment.

This observatory contains a structure which is designed to house and protect several elements from external factors. The structure can also withstand the traction of the marine cable generated by the water currents. There are marine sensors too. There is a buoy which serves as an extension of the OBSEA observatory. It works as a useful platform for oceanographic and environmental parameters measurements.

It is used as a testbed for marine sensors and equipment. It has a video surveillance camera and a weather station with GPS that measures wind direction, wind speed, barometric pressure, air temperature among other parameters.

There is an underwater camera in this observatory which provides real time video allowing the surveillance of the platform, showing images of wildlife in the environment of the platform. There are lots of other sensors in the OBSEA observatory, like a broadband hydrophone, a wave measurement system, or a broadband seismometer for example.

Many technologies are used for marine technologies, but some of them are terrestrial technologies. The industrial applications of its technologies are used in communications, in electronics or in mechanics for example.

3. Context

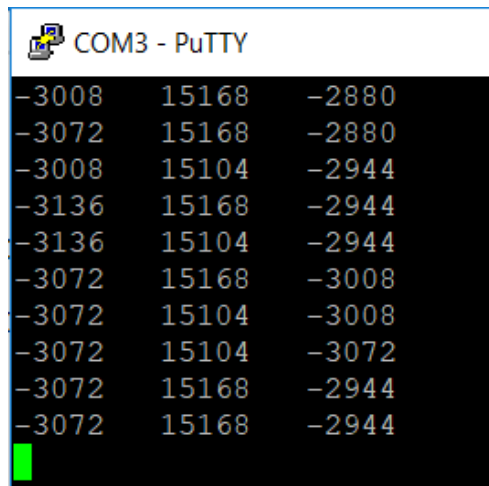
3.1. Existing work

In this part, I will mention more specifically the existing work which is linked to my project. It was made by Mr. Kaloyan Ganchev, a Bulgarian student who did an internship in the SARTI centre from April to July 2018. He worked on a low powered module (TD1205P) using Sigfox communication. He did some programs in C.

In every example, the module sends a 12 bytes message for transceiving information. The aim of these programs is to retrieve GPS data, accelerometer data, battery level or temperature.

He did one program which calculates and prints the minimum, the maximum, the average and the standard deviation of the accelerometer data for x, y, z at given intervals. In this example, it is possible to change some parameters with commands, using serial communication. For example, it is possible to change the frequency or the scale, to turn on/off the accelerometer...

Another program configures the accelerometer and prints the 3 axes (x, y, z) data. These two programs were done in order to retrieve accelerometer data. This is an example of data the user can have when he tests the last program:



```
COM3 - PuTTY
-3008    15168    -2880
-3072    15168    -2880
-3008    15104    -2944
-3136    15168    -2944
-3136    15104    -2944
-3072    15168    -3008
-3072    15104    -3008
-3072    15104    -3072
-3072    15168    -2944
-3072    15168    -2944
```

Figure 3 - Result of the program which retrieves x, y and z data

He did another program which configures GPS and sends messages at given intervals. He added some customs commands to change GPS timeout, GPS fixing interval and GPS fixing mode. The message is composed of 12 bytes (4 bytes for latitude, 4 bytes for longitude, 1 byte for voltage, 1 byte for temperature and 2 empty bytes).

Finally, he created a program which looks like the precedent program, but he saves the minimum value of z axis during the given interval in the 11th and 12th bytes.

I have to mention other programs which were made by Louis Moreau, an intern at Sigfox. I retrieved those programs thanks to a tutorial I will evoke later in this report. One of those programs is called `accelero_event`. With this application, it is possible to see some information when the TD1205P module is in motion.

```
COM3 - PuTTY
Accelerometer event:
x high
z high
Accelerometer event:
x high
y high
Accelerometer event:
x high
z high
Accelerometer event:
x high
```

Figure 4 - Result of the program which detects accelerometer events

3.2. Initial request

In the SARTI centre, lots of projects are closely linked to the sea. The initial request was to think about deploying a system into a buoy which will be able to send GPS data and to detect movements. For example, the system could be applied to a marine target like this buoy:



Figure 5 - Ocean drifter

This buoy is a system which can monitor water currents. It can contain different sensors, like a wave measurement system for example. This type of buoy will move in the sea with the currents, so you could easily understand the utility to retrieve its GPS coordinates and other data, like the battery level for example.

With Sigfox, there is an application which shows the messages position of a device when there are messages. This is example of track map for a device:

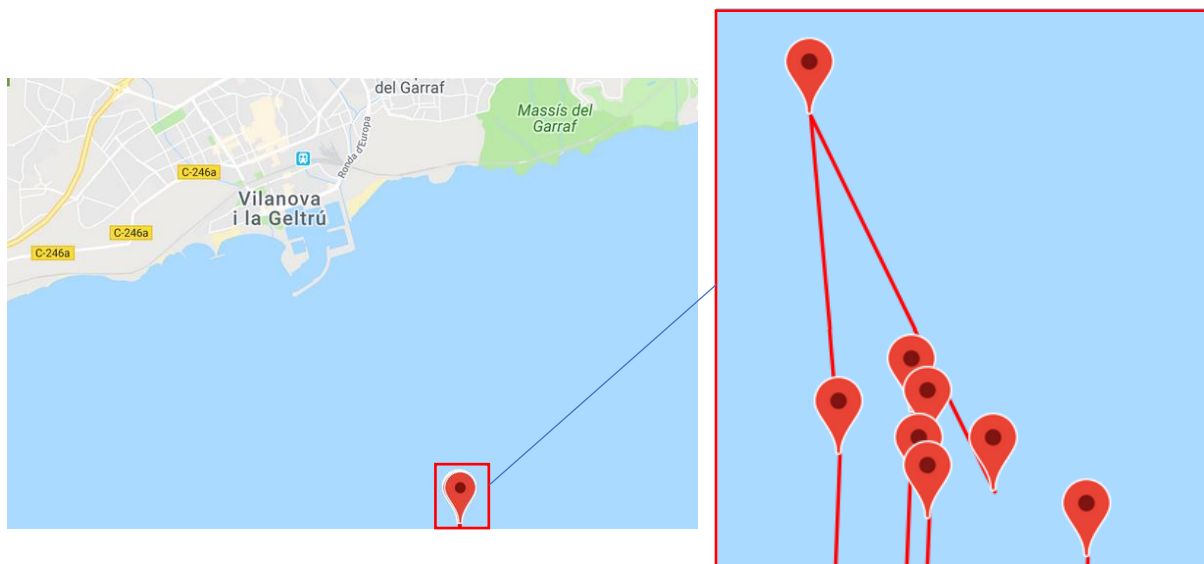


Figure 6 - Example of device's track on a map

This could be an example of track for a device into a buoy. The position is retrieved in this case every 2 hours.

To sum up, it is possible to retrieve GPS and accelerometer data at given intervals with the precedent programs. In certain ways, it is possible to detect movements too.

The idea of the project is now to be able to know if the device is in motion during a specific interval. In other words, the idea is to detect if a buoy is in movement for example, due to the currents. Then, the aim is to send specific messages when the device is in motion, and to send other messages when the device is not in motion.

The objective is to send information at relatively "narrow" intervals when the device moves, and to send data at larger intervals when the device doesn't move, in order to gain electric consumption.

3.3. Similar technologies in the market

Chipfox GPS Tracker

Chipfox GPS tracker is a small battery-operated, highly energy efficient device for tracking and securing of non-powered assets, animals or people. It communicates on Sigfox network. It doesn't need SIM card, and battery lasts for several years. Signal is resistant to jamming. It uses a Telecom Design module.

This tracker has got small dimensions and a low weight. The main advantages are his extreme battery life (up to seven years) and a high sensitivity GPS. There is 3D accelerometer for motion detection. It is possible to use a mobile app and a web interface with this module.



Figure 7 - Chipfox GPS Tracker

This table shows the main differences between GSM (Global System for Mobile Communications) and the technology which is used in this device when you try to use GPS.

	GPS/GSM	Chipfox GPS Tracker
Battery life	days	years
Signal jamming	easy	not possible
Tracker detection	yes, easy	difficult
GPS indoor	no	yes, accuracy 100-5000m
SIM card	yes	no
Operator contract	monthly SIM/data contract	annual subscription, first year included
Roaming	paid	free
Tracker functions	usually not	yes

Figure 8 - Comparison of GPS/GSM with GPS/IoT technology

Oyster LoRaWAN

Oyster is a compact GPS tracking device that has been designed for tracking containers, trailers and other assets where long battery life is required without sacrificing the frequency of updates and performance.

Oyster can operate in ultra-low power modes. With a battery life of up to 5 years, it can be attached to assets and tracked without needing to change batteries. Oyster has built-in antennas for reception and communication, a 3D accelerometer and a high-performance GPS.

This device uses LoRaWAN technology to coverage assets at a low data cost. Compared to Sigfox, LoRaWAN has got a wider band, so there are more interferences.

LoRaWAN looks the better option if you need bidirectional communication. So, if you need command-and-control functionality, it is the best technology. With Sigfox, you could use bidirectional command-and-control functionality, but it is more difficult because of the asymmetric link. To sum up, Sigfox is better for applications that send only small and infrequent bursts of data, like alarms for example.



Figure 9 - Oyster LoRaWAN

You have seen the presentation of two devices, but it is possible to find lots of devices which can tracked objects or persons in the market. There are many technologies, but you have to think about your requirements in order to select the best product.

4. Technologies and tools

4.1. LPWAN networks

LPWAN (Low Power Wide Area Network) technology is a type of wireless telecommunication wide area network designed to allow long range communications at a low bit rate among connected objects, such as sensor operated on a battery.

The table below is a comparison of some LPWAN technologies on the market:








							
	LoRa / LoRaWAN	Sigfox	NB-IoT	LTE-M	RPMA	Weightless-P	Symphony Link
Origin	France	France	USA (Global)	USA (Global)	USA	UK	USA
Proprietary or open	LoRa – proprietary LoRaWAN – open	Net – proprietary Devices – open	Open	Open	Proprietary	Open	Proprietary
Cellular	No	No	Yes	Yes	No	No	No
Spectrum	Unlicensed	Unlicensed	Licensed	Licensed	Unlicensed	Unlicensed	Unlicensed
Range, km	urban: 2-5 rural: 15	urban: 3-10 rural: 30-50	urban: 1-5 rural: 10-15	urban: 2-5	urban: 1-3 rural: 25-50	urban: 2	urban: 2-5 rural: 15
Speed, uplink / downlink	50 kbps / 50 kbps	300 bps / –	250 kbps / 250 kbps	1 Mbps / 1 Mbps	634 kbps / 156 kbps	100 kbps / 100 kbps	100 kbps / 100 kbps
Power consumption	●●●	●	●	●●●	●●	●	●●
Security	●●	●●	●●●	●●●	●●●	●●●	●●●
Availability of devices	●●	●●●	●●	●	●●	●	●●
Price*	●●	●	●●	●●●	●●●	●	●●
Areas of application	Precision farming, manufacturing automation, pipeline monitoring	Predictive maintenance, capacity planning, demand forecasting	Electric metering, manufacturing automation, retail PoS	tracking objects, wearables, energy management, utility metering, city infrastructure	Digital oilfield, connected cities, usage-based insurance, agriculture	Smart grid, healthcare, automotive, smart cities, asset tracking	Industrial control systems, lighting control, alarm systems
Supporting companies	IBM, Semtech, Cisco, HP, Orange, Kerlink, Actility	STMicroelectronic, Texas Instruments, Atmel, Silicon Labs	Huawei, Ericsson, Qualcomm, Vodafone	Verizon, AT&T, Nokia	Ingenu	Accenture, Sony Europe, uniik, ARM, Telensa	Link Labs

Figure 10 - Some LPWAN networks

In the SARTI centre and especially for this project, Sigfox technology is used because of the very low power consumption and the good range compared to the other technologies.

4.2. Sigfox

Sigfox is the world's leading IoT services provider. It is a French company founded in 2009 which builds wireless networks to connect low-power objects. Sigfox uses a global Low Power Wide Area Network (LPWAN).



Figure 11 - Sigfox logo

Sigfox wireless systems send very small amounts of data (12 bytes) using standard radio transmission methods, and it takes very narrow chunks of spectrum. The long range is accomplished as a result of very slow messages. This technology allows the receiver to only listen in a tiny slice of spectrum, which mitigates the effect of noise. It is a good fit for any application that needs to send small and infrequent bursts of data.

Things like basic alarms, location monitoring, are examples of one-way systems that might make sense for this network. In these networks, the signal is typically sent a few times to “ensure” the message goes through.

While this works, there are some limitations, such as shorter battery life for battery-powered applications, and an inability to guarantee a message is received.

Sigfox employs a proprietary technology that enables communication using the Industrial, Scientific and Medical ISM Radio Band which uses 868 MHz in Europe and 902 MHz in the US. It utilizes a wide-reaching signal that passes freely through solid objects, called “ultra-narrowband” and requires little energy. The signal can be used to easily cover large areas and to reach underground objects. The ISM radio bands support limited bidirectional communication.

The existing standard for Sigfox communications supports up to 140 uplink messages a day (messages of 12 bytes, excluding message header and transmission information) and up to 4 downlink messages per day (messages of 8 bytes).

To sum up, the advantages of Sigfox are:

- a low energy consumption
- a low cost
- a complementary technology

Sigfox is rolling out the first global IoT network to listen to billions of objects broadcasting data, without the need to establish and maintain network connections. The Sigfox network offers a high quality of service which is highly resistant to interference. It's perfectly fitted for low cost IoT devices that need to run on extremely low energy consumption levels and need to send data over a very long range.

4.3. TD1205P module

The TD1205P module feature the Sigfox high efficiency Gateway which offers high interference immunity performances on ISM band. This compact geolocation engine includes GNSS (Global Navigation Satellite System), 3-axis accelerometer and sensors (for temperature for example) in order to provide a high efficiency tracking solution.

Highly programmable with TD Next's free Software Development Kit, this module is the most integrated and lowest-power engine for most battery-operated Sigfox objects.



Figure 12 - TD1205P: top view



Figure 13 - TD1205P: bottom view

4.4. EFM32 Tiny Gecko

Silicon Labs EFM32 Tiny Gecko is ideal for low power application. It is a 32-bit Microcontroller. This board can be used for flashing the TD1205P module.



Figure 14 - EFM32 Tiny Gecko

4.5. J-Link EDU

With this device, it is possible to upload a program on the TD1205P module too.



Figure 15 - J-Link EDU

5. Technical development

At the beginning, the aim was to program the TD1205P module with Eclipse. There are two solutions for uploading programs:

- Using the EFM-32 Tiny Gecko
- Using the J-Link EDU

EFM-32 Tiny Gecko

To use this device, you need to follow the instructions on this website:

<https://www.instructables.com/id/Sigfox-GPS-Tracker/>

J-Link EDU

An electric cable was made in order to work with the J-Link device. After this work, programs can be uploaded using the software J-Flash.

5.1. Fabrication of an electric cable (for the J-Link Device)

The first step is to retrieve the pin diagrams of the two components:

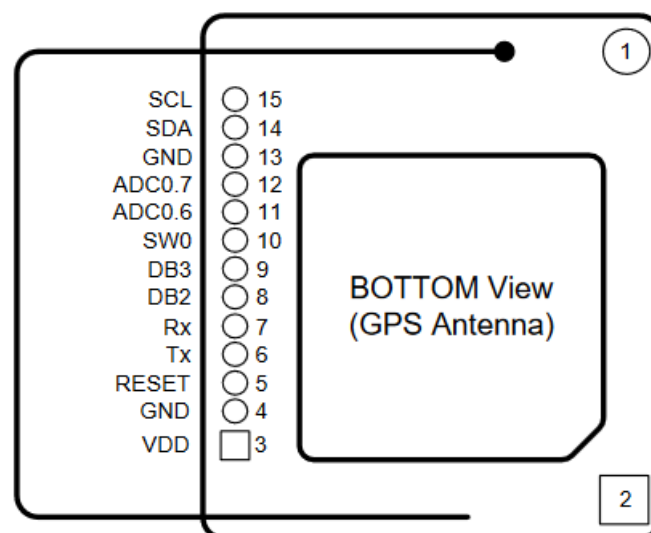


Figure 16 - TD1205P Pin Diagram

VTref	1 ●	● 2	NC
nTRST	3 ●	● 4	GND
TDI	5 ●	● 6	GND
TMS	7 ●	● 8	GND
TCK	9 ●	● 10	GND
RTCK	11 ●	● 12	GND
TDO	13 ●	● 14	GND*
RESET	15 ●	● 16	GND*
DBGREQ	17 ●	● 18	GND*
5V-Supply	19 ●	● 20	GND*

J-Link and J-Trace have a JTAG connector compatible to ARM's Multi-ICE. The JTAG connector is a 20 way Insulation Displacement Connector (IDC) keyed box header (2.54mm male) that mates with IDC sockets mounted on a ribbon cable.

**On some models like the J-Link ULTRA, these pins are reserved for firmware extension purposes. They can be left open or connected to GND in normal debug environment. Please do not assume them to be connected to GND inside J-Link.*

Figure 17 - Pinout for JTAG (J-Link connector)

With the multimeter, a test was done for finding the connections between the two devices. This figure represents the connections which are needed:

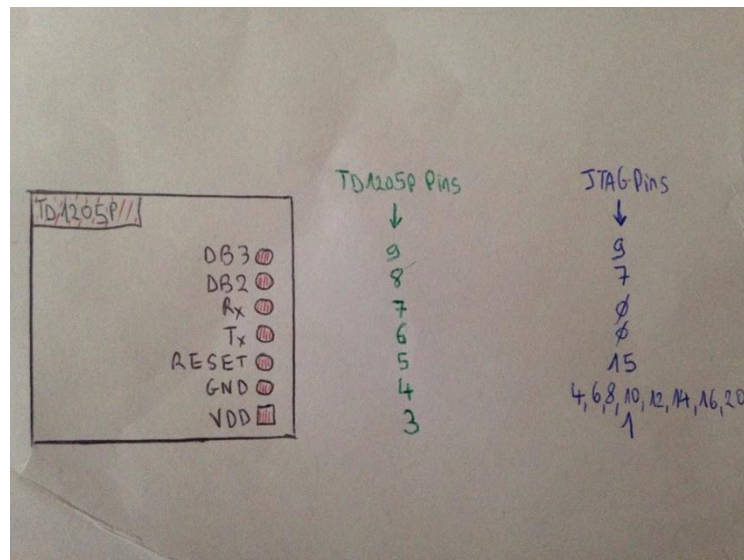


Figure 18 - Connections between TD1205P and J-Link EDU

A TTL-232RG cable, which is a USB to TTL serial UART converter cable, is necessary.



Figure 19 - TTL-232RG cable

Only the black (GND), the orange (TX) and the yellow (RX) cables are necessary. This table shows the connections between the TTL-232RG cable and the TD1205P module:

Colour of the cable (TTL-232RG)	TD1205P pin
Black (GND)	4 (GND)
Orange (TX)	7 (RX)
Yellow (RX)	6 (TX)

Figure 20 - Connections between TD1205P and J-Link EDU

After soldering, the result of this work is below:

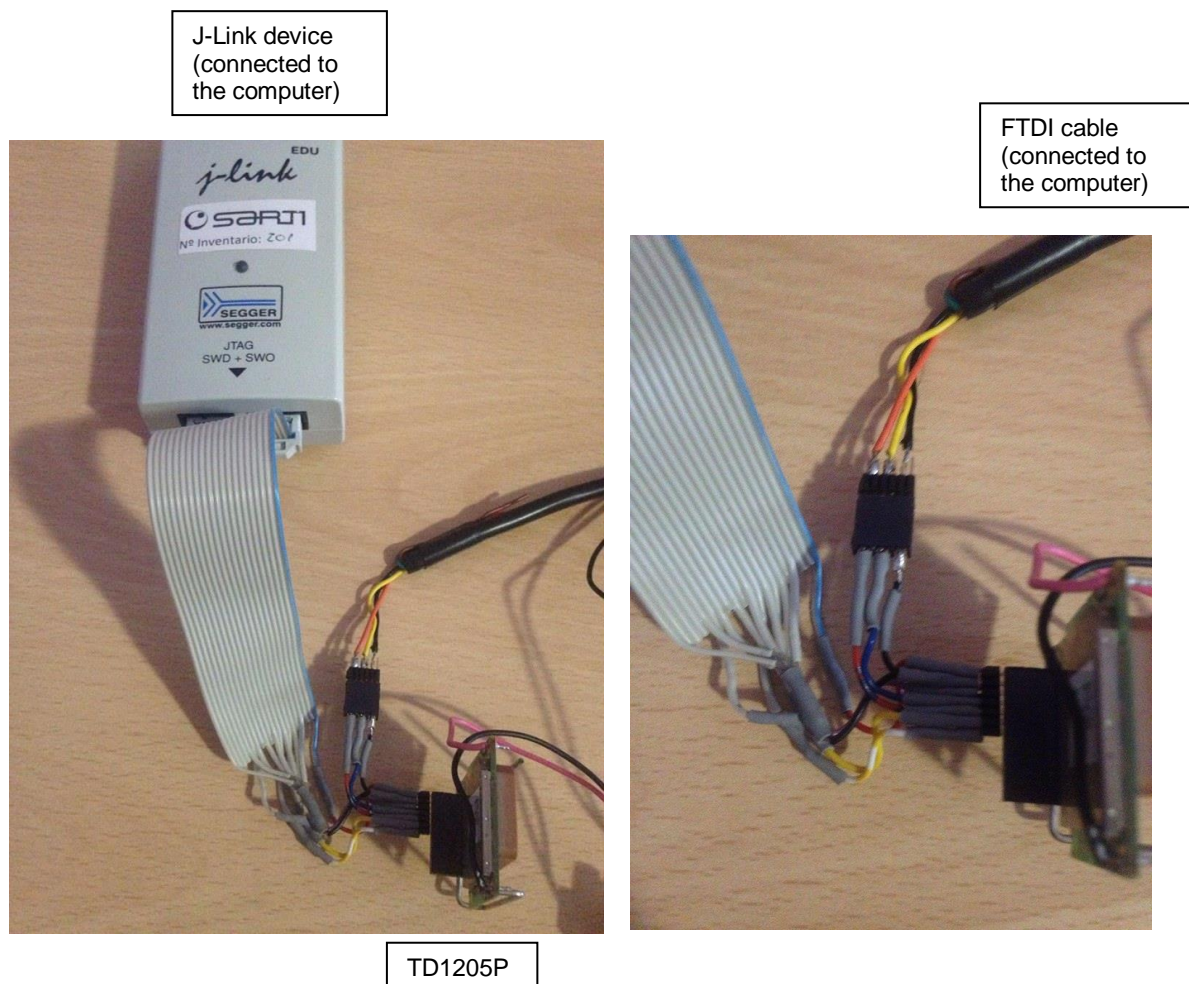


Figure 21 - Electric cable for using J-Link device

5.2. Equipment

In order to upload a program on the module, users have to download the software J-Flash if they decide to use the J-Link device. In the figure below, there are some parameters that users have to type:

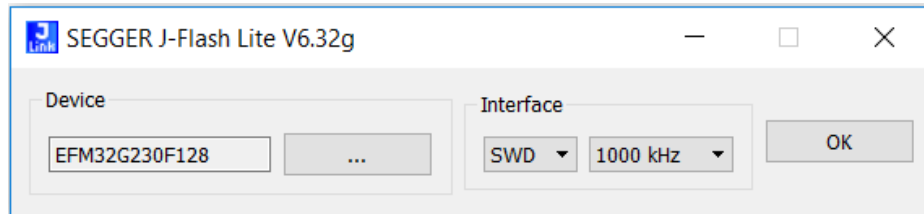


Figure 22 - Parameters for using J-Flash

Then, a new window will appear. Users have to choose the binary file they want to upload on the module, and then click on “Program Device”. During my project, sometimes it worked, but I often got an error (see the figure below):

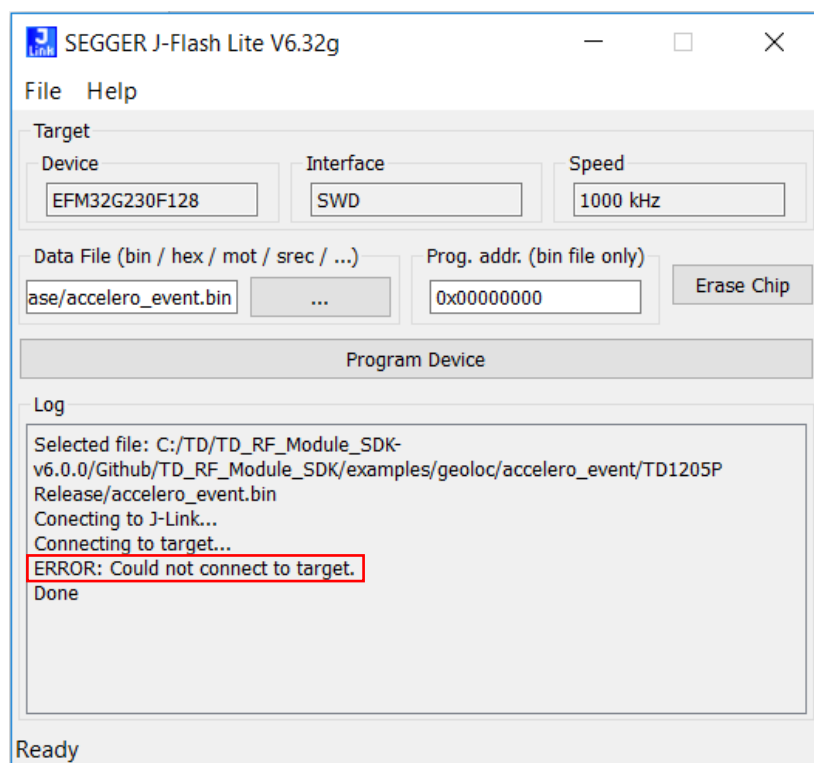


Figure 23 - Possible issue with J-Flash

I didn't find the cause of this problem. For this reason, I used the EFM-32 Tiny Gecko most of the time. To program the TD1205 module with this device, see the tutorial in the next part.

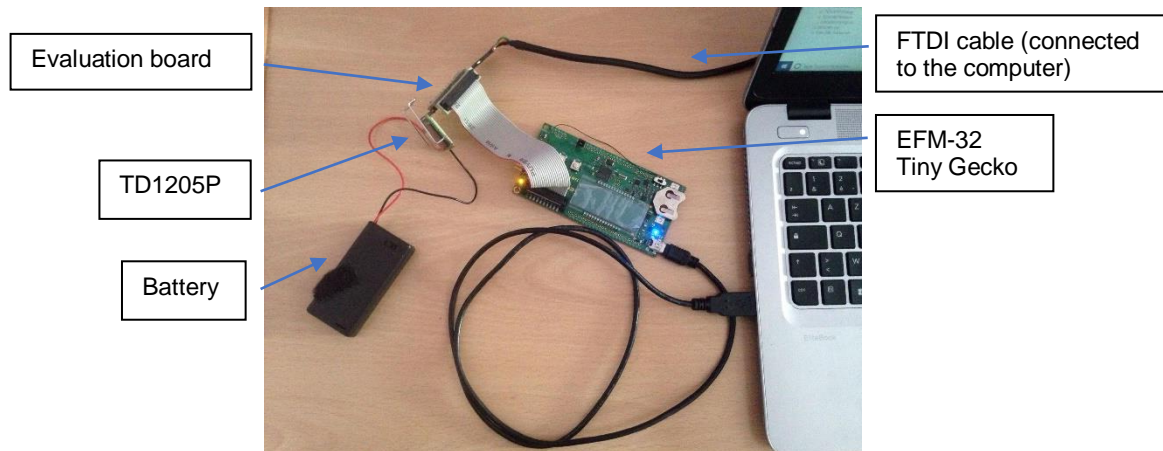


Figure 24 - Uploading a program with the EFM-32 Tiny Gecko

5.3. Tutorial

At this step, it is possible to program the TD1205P module using the EFM-32 Tiny Gecko or the J-Link device (when it works). The best thing to do in order to understand the many possibilities of this module is to follow the tutorial of this website:

<https://www.instructables.com/id/Sigfox-GPS-Tracker/>

Thanks to this tutorial, we can send the GPS coordinates of the module using Sigfox. We are able to send the battery level, the speed and other information too, and to retrieve these data. I will not explain more specifically this tutorial, because the website does it better than me, but maybe it is possible to have issues which are not mentioned. For example, it is possible to have this issue:

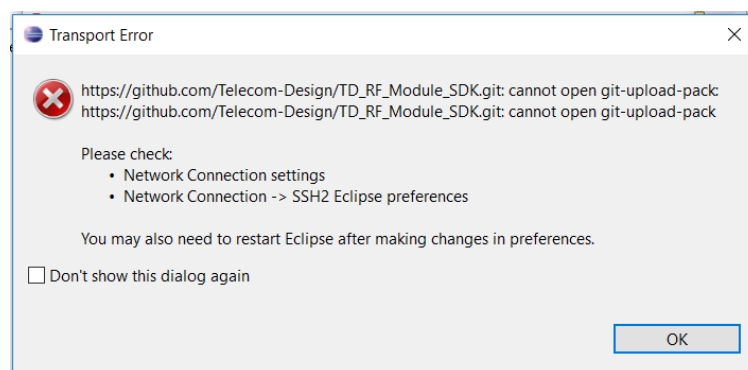


Figure 25 - Eclipse error in the step 2 of the tutorial

Adding this line `"-Dhttps.protocols=TLSv1.1,TLSv1.2"` at the end of this file `"..\TD\TD_RF_Module_SDK-v6.0.0\eclipse\eclipse.ini"` will solve the problem.

5.4. Application and tests

After configuring the environment thanks to the tutorial and testing the existing work, I was able to think about the program I had to do. The code is this program (gps_movement_tracker.c) can be retrieved on this Github account:

https://github.com/JulienPeudennier/sigfox_examples

5.4.1. General presentation

With this program, the TD1205P module detects if there is movement. Then, this device sends some information every T1 period if it is in motion, and other information every T2 period if it is not in motion.

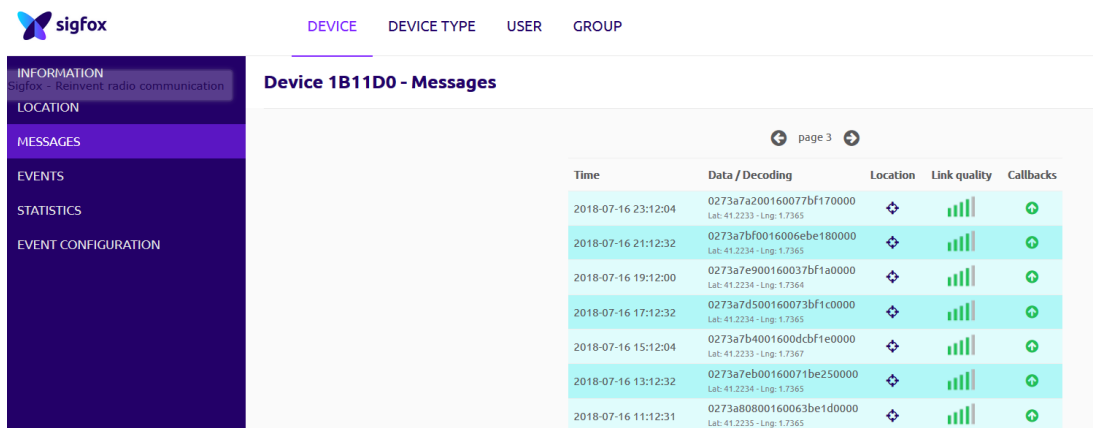
For transceiving information, this program uses a 12 bytes format. Message is formed by using some code from the tutorial:

- 4 bytes: GPS longitude or free
- 4 bytes: GPS latitude or free
- 1 byte: voltage
- 1 byte: temperature
- 2 bytes: free

The baud rate which is used here is 9600 bits/s. A keepalive message is sent at every boot.

Users can change some parameters of the program using serial communication. Indeed, it is possible to communicate with the module thanks to putty.exe, a free and open-source terminal emulator and serial console.

To see the different messages, users can use Sigfox backend. They have to create an account. Once they have access to their account, they can go to "Device" and select their module by clicking on its id. Then, they can click on "Message" to see the messages sent by the device.



Time	Data / Decoding	Location	Link quality	Callbacks
2018-07-16 23:12:04	0273a7a200160077bf170000 Lat: 41.2233 - Long: 1.7365	+		+
2018-07-16 21:12:32	0273a7bf0016006ebe180000 Lat: 41.2234 - Long: 1.7365	+		+
2018-07-16 19:12:00	0273a7e90016003bf1a0000 Lat: 41.2234 - Long: 1.7364	+		+
2018-07-16 17:12:32	0273a7d500160073bf1c0000 Lat: 41.2234 - Long: 1.7365	+		+
2018-07-16 15:12:04	0273a7b4001600dcbf1e0000 Lat: 41.2233 - Long: 1.7367	+		+
2018-07-16 13:12:32	0273a7eb00160071be250000 Lat: 41.2234 - Long: 1.7365	+		+
2018-07-16 11:12:31	0273a80800160063be1d0000 Lat: 41.2235 - Long: 1.7365	+		+

Figure 26 - Sigfox backend

In this part, I will explain the link between the values which are sent by the device, and the values in reality. I will take an example of message to explain this:




Time	Data / Decoding	Location	Link quality	Callbacks
2018-07-16 23:12:04	0273a7a200160077bf170000 Lat: 41.2233 - Lng: 1.7365			

Figure 27 - Example of message sent by the device

As you can see, this is a 12 bytes message. There are 4 bytes for GPS latitude, 4 bytes for GPS longitude, 1 byte for battery level, 1 byte for temperature and 2 empty bytes.

GPS latitude

Hexadecimal data: 02 73 A7 A2.

This value in decimal is: 41 133 986.

To get the decimal degrees, we have to divide the decimal value by 1 000 000.

So here, the result is 41,133986 -> 41°.

The value of the decimal minutes is 13,3986'.

To have the decimal part of the GPS latitude value, we have to divide the decimal minutes value by 60. Here, $13,3986 / 60 = 0,22331$.

So, the GPS latitude value is 41,22331°.

GPS longitude

Hexadecimal data: 00 16 00 77.

This value in decimal is: 1 441 911.

To get the decimal degrees, we have to divide the decimal value by 1 000 000.

So here, the result is 1,441911 -> 1°.

The value of the decimal minutes is 44,1911'.

To have the decimal part of the GPS longitude value, we have to divide the decimal minutes value by 60. Here, $44,1911 / 60 = 0,7365183333...$

So, the GPS longitude value is 1,7365183333°.

Battery level

Hexadecimal data: BF.

This value in decimal is: 191.

To get the value in volts, we have to multiply this value by 0,015.

$191 * 0,015 = 2,865$.

So, the voltage value is 2,865 V.

Temperature

Hexadecimal data: 17.

This value in decimal is: 23.

So, the temperature value is 23°C.

Now you know the meaning of the values which are sent by the module. I will present in the next parts the different commands which can be used in order to configure some parameters. These commands must be typed during the 3 minutes after the launching of the program. Indeed, serial communication is stopped after 3 minutes in order to save power. In that way, the module doesn't wait for serial connection, so it doesn't consume energy for that.

5.4.2. GPS Mode

It is possible to work with different power modes. This figure shows all the possibilities:

```
/** GPS Power mode types */
typedef enum {
    TD_GEOLOC_OFF = 0,
    TD_GEOLOC_HW_BCKP = 1,
    TD_GEOLOC_SOFT_BCKP = 2 ,
    TD_GEOLOC_NAVIGATION = 3,
    TD_GEOLOC_POWER_SAVE_MODE = 4,
    TD_GEOLOC_NAVIGATION_COLD_START = 5,    // Force cold start
    TD_GEOLOC_NAVIGATION_WARM_START = 6,    // Force warm start
    TD_GEOLOC_NAVIGATION_HOT_START = 7,     // Force hot start
    TD_GEOLOC_NAVIGATION_IDLE = 8           // Stop GNSS receive
} TD_GEOLOC_PowerMode_t;
```

Figure 28 - GPS Power Modes

To have more information about those power modes, go to this website:

<http://community.td-next.com/viewtopic.php?f=5&t=55>

In the program, we decided to only use three modes:

- `TD_GEOLOC_OFF`
- `TD_GEOLOC_HW_BCKP`
- `TD_GEOLOC_NAVIGATION`

`TD_GEOLOC_NAVIGATION` is only used by the program when we try to find GPS coordinates. There is a command which allows the user to choose the power mode he wants to use, between the 3 other modes. This command is:

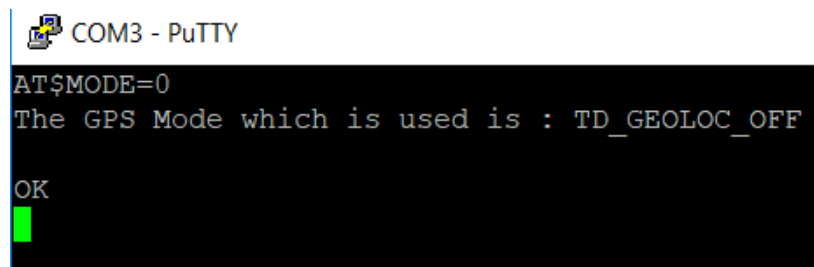
AT\$MODE=

After typing this command, there are two possible values:

- 0 for *TD_GEOLOC_OFF* mode: with this mode, there is no RAM retention, everything is off, there is no consumption. The module doesn't store any information about the fixing and the satellites.

- 1 for *TD_GEOLOC_HW_BCKP* mode: with this mode, there is RAM retention. The module can store the last information about the fixing and the satellites in the RAM. It means that the fixing time for finding GPS data will be significantly shorter the next time the GPS will wake up in order to get data.

The default value in this program is *TD_GEOLOC_HW_BCKP* mode. When the GPS tries to find coordinates data, the mode is *TD_GEOLOC_NAVIGATION*. After that, the mode is the one the user chose (default mode or the other if he typed a command).



```
COM3 - PuTTY
AT$MODE=0
The GPS Mode which is used is : TD_GEOLOC_OFF
OK
█
```

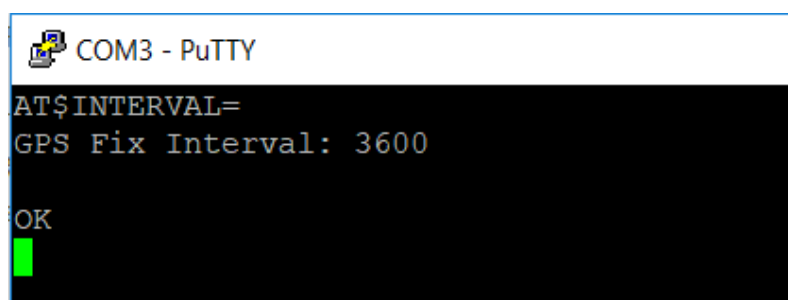
Figure 29 - Exemple of command to change GPS power mode

5.4.3. Interval T1

This interval is the time between two consecutive messages when the module is in motion. Users can change the value of this interval with this command:

AT\$INTERVAL=

When there is nothing written after this command, it prints the current value:



```
COM3 - PuTTY
AT$INTERVAL=
GPS Fix Interval: 3600
OK
█
```

Figure 30 - Command to see the current interval T1

When there is a positive value after this command, this value become the new interval T1 in seconds. In this example, the new interval T1 is 7200 seconds (2 hours). Don't forget that Sigfox supports 140 messages a day! Here is an example when the user decides to have an interval of two hours:

```
COM3 - PuTTY
AT$INTERVAL=7200
OK
AT$INTERVAL=
GPS Fix Interval: 7200
OK
```

Figure 31 - Command for changing the interval T1

The default value of the interval T1 is 3600 seconds. If the user changes the value of this interval and resets the module, its value will be the default value because there is no ROM. The next figure shows the messages on the Sigfox backend when the interval T1 has got the default value (3600 seconds), and the device is in motion:










Time	Data / Decoding	Location	Link quality	Callbacks
2018-09-08 20:46:56	0275240f0020550dbe160000 Lat: 41.3856 - Lng: 2.1982			
2018-09-08 19:46:57	02750f5b00203d71be170000 Lat: 41.3768 - Lng: 2.1881			
2018-09-08 18:46:31	0275197d00201a94be180000 Lat: 41.3811 - Lng: 2.1733			

Figure 32 - Sigfox messages when T1 = 3600 s and movement

The duration between the messages are not exactly the same because sometimes it could take more time to find satellites. The figure below shows the messages on the Sigfox backend when the user chooses a value of 600 seconds (10 minutes) thanks to a serial command.






Time	Data / Decoding	Location	Link quality	Callbacks
2018-08-30 17:59:32	02739c3a0015f5e9c31b0000 Lat: 41.2184 - Lng: 1.7320			
2018-08-30 17:49:29	02739c680015f5c8c31c0000 Lat: 41.2185 - Lng: 1.7320			
2018-08-30 17:39:32	02739c450015f61ac31a0000 Lat: 41.2185 - Lng: 1.7321			

Figure 33 - Sigfox messages when T1 = 600 s and movement

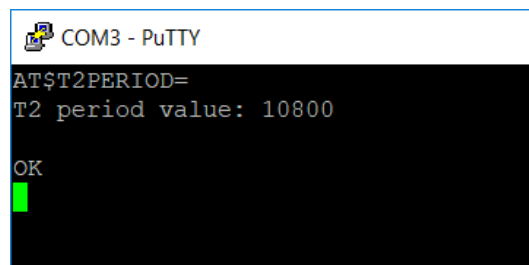
This test looks good in order to check this functionality. I used a value of 600 seconds here because I didn't want to wait too much time! But in the application, we use bigger values (1 hour for example) because we don't want to have values regularly: we want to gain consumption!

5.4.4. Interval T2

This interval is the time between two consecutive messages when the module is not in motion. This interval is larger than T1 because we don't need data regularly if the device doesn't move. Users can change the value of this interval using this command:

AT\$T2PERIOD=

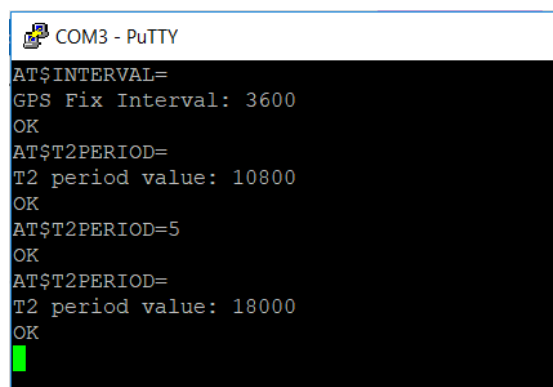
When there is nothing written after this command, it prints the current value:



```
COM3 - PuTTY
AT$T2PERIOD=
T2 period value: 10800
OK
```

Figure 34 - Command to see the current interval T2

Users can change the value of this interval by adding a positive number after this command. Indeed, if someone types a value after this line, the new value of the interval T2 will be the interval T1 multiplied by this number. In this example, the value of the interval T1 is 3600 seconds (1 hour). The default value of the command is 3, so the default value of the interval T2 is $3600 * 3 = 10800$ seconds (3 hours). Thanks to the command, the user decides that the new value of the interval T2 has to be $5 * T1$. In this case, T2 value is $5 * 3600 = 18000$ seconds (5 hours).



```
COM3 - PuTTY
AT$INTERVAL=
GPS Fix Interval: 3600
OK
AT$T2PERIOD=
T2 period value: 10800
OK
AT$T2PERIOD=5
OK
AT$T2PERIOD=
T2 period value: 18000
OK
```

Figure 35 - Command for changing the interval T2

The next figure shows the messages on the Sigfox backend when the interval T1 is 600 seconds (10 minutes), and the command has got the default value 3. So, the interval T2 has got a value of 1800 seconds (30 minutes). Between the two first messages, there was movement. That is why there are 10 minutes between them. Between the two last messages, there was no movement. That is why there are 30 minutes between them. The first eight bytes are 0x00 in the last message because the “type of message” command had the default value (0), so we only sent voltage and temperature. There are other possibilities, but it will be explained later.

Time	Data / Decoding	Location	Link quality	Callbacks
2018-08-30 18:29:26	0000000000000000c41a0000 Lat: 0.0000 - Lng: 0.0000			
2018-08-30 17:59:32	02739c3a0015f5e9c31b0000 Lat: 41.2184 - Lng: 1.7320			
2018-08-30 17:49:29	02739c680015f5c8c31c0000 Lat: 41.2185 - Lng: 1.7320			

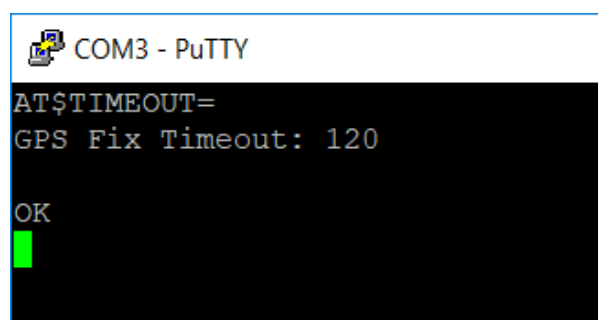
Figure 36 - Sigfox messages when T1 = 600 s, T2 = 1800 s and no movement

5.4.5. Timeout

This data is the duration during which the GPS tries to find satellites in order to have GPS coordinates. Users can change its value using this command:

AT\$TIMEOUT=

When there is nothing written after this command, it prints the current value:



```

COM3 - PuTTY
AT$TIMEOUT=
GPS Fix Timeout: 120
OK

```

Figure 37 - Command to see the current timeout

The default value of the timeout is 120 seconds (2 minutes). If the user types a positive number after this command, this number will be the new timeout. In this example, the new timeout becomes 180 seconds (3 minutes).

```
COM3 - PuTTY
AT$TIMEOUT=180
OK
AT$TIMEOUT=
GPS Fix Timeout: 180
OK
```

Figure 38 - Command for changing the timeout

If the device is outdoors, it will normally find satellites, so there will be no problem for sending GPS data. But if the device is indoors, it can be difficult to find satellites, and sometimes it is not possible to retrieve data. Sometimes, it will possible to find GPS data with a long timeout, but sometimes it will be impossible. If it is not possible, the module will only send temperature and voltage (and 8 bytes 0x00 for latitude and longitude).

This is an example of messages when there is movement, the value of the interval T1 is 600 seconds (10 minutes), the timeout is 120 seconds (2 minutes), and the module is not able to find satellites.








Time	Data / Decoding	Location	Link quality	Callbacks
2018-09-07 19:09:17	0000000000000000c11e0000 Lat: 0.0000 - Lng: 0.0000			
2018-09-07 18:59:17	0000000000000000c11d0000 Lat: 0.0000 - Lng: 0.0000			
2018-09-07 18:47:19	02739c370015f620c2190000 Lat: 41.2184 - Lng: 1.7321			

Figure 39 - Sigfox messages when timeout expires before finding GPS coordinates

With these values, the device has to send information every 10 minutes. But there are 12 minutes between the two first messages because the module cannot retrieve GPS data. Indeed, it started looking for satellites after 10 minutes and during 2 minutes, but without success. For this reason, the device sent only voltage and temperature 12 minutes after the first message.

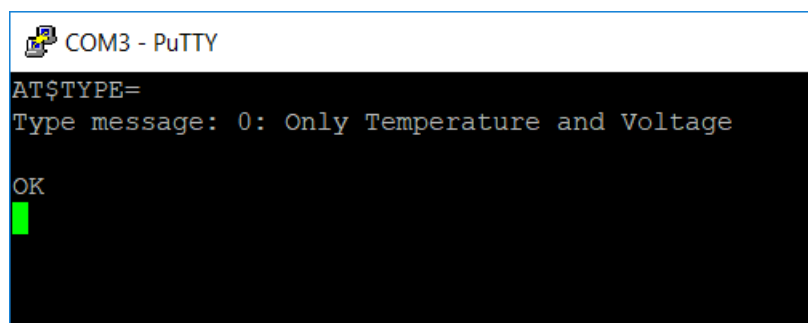
The next message is sent 10 minutes after because the module is not able to retrieve GPS data too. It is looking for satellites until timeout expiration. This message is sent 10 minutes after the precedent message and not 12 minutes after because the timeout is included in the 10 minutes.

5.4.6. Type of message

Users have the possibility to change the type of message sent by the module when there is no movement thanks to a command:

AT\$TYPE=

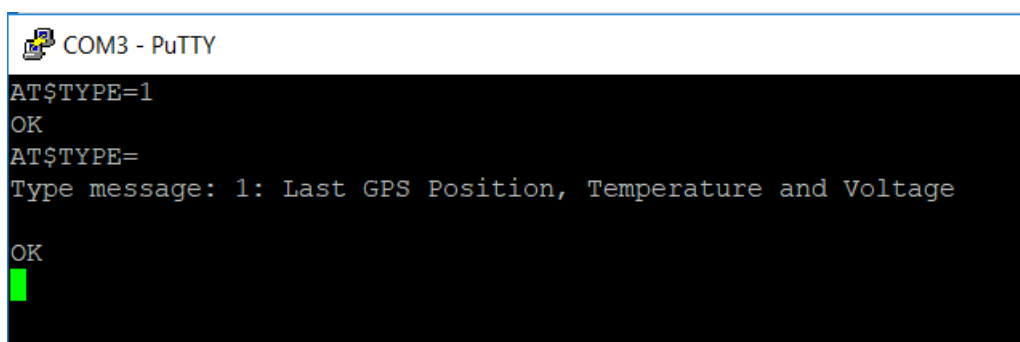
When there is nothing written after this value, it prints the type of message sent in case of no movement:



```
COM3 - PuTTY
AT$TYPE=
Type message: 0: Only Temperature and Voltage
OK
█
```

Figure 40 - Command the see the current type of message sent (no movement)

Users can change the type of message sent by adding a number after this command. There are two possible values: 0 or 1. The default value is 0. It means that, in case of no movement, the device will only send temperature and voltage. The first eight bytes will be 0x00. If the user wants GPS coordinates too, he has to type '1' after the equal sign. The result of this command is below:



```
COM3 - PuTTY
AT$TYPE=1
OK
AT$TYPE=
Type message: 1: Last GPS Position, Temperature and Voltage
OK
█
```

Figure 41 - Command for changing the type of message sent (no movement)

In this case, the device will not do a new measure for GPS coordinates: it will send the last GPS latitude and longitude which were measured when there was movement. If the device doesn't make measurement for GPS since the last reset, the first eight bytes will be 0x00.

This is an example of messages which are sent when the interval T1 is 300 seconds (5 minutes), the interval T2 is 900 seconds (15 minutes - I take small values for the test because I don't want to wait too much, but it is the same with biggest values -) and the type of message is '0' (only temperature and voltage).

Between the two first messages, there was movement. That is why there was 5 minutes between them. However, between the two last messages, there was no movement. That is why there was 15 minutes between them. The type of command was '0', so the first eight bytes were 0x00 in the last message.


Time	Data / Decoding	Location	Link quality	Callbacks
2018-08-31 19:03:13	0000000000000000c3160000 Lat: 0.0000 - Lng: 0.0000			
2018-08-31 18:48:19	0273973b0015fcd4c1160000 Lat: 41.2163 - Lng: 1.7350			
2018-08-31 18:43:19	0273990c0015fa4fc1190000 Lat: 41.2171 - Lng: 1.7339			

Figure 42 - Sigfox messages sent when the type of message is '0' (no movement)

Here, it is the same test, but the type of messages chosen was '1' (with GPS coordinates). In the last message, the first eight bytes were the latitude and the longitude of the last GPS position.








Time	Data / Decoding	Location	Link quality	Callbacks
2018-09-04 10:15:54	0273a8180015fdabc41a0000 Lat: 41.2235 - Lng: 1.7353			
2018-09-04 10:00:57	0273a8180015fdabc31a0000 Lat: 41.2235 - Lng: 1.7353			
2018-09-04 09:55:58	0273a87a0015ff4ac31b0000 Lat: 41.2237 - Lng: 1.7360			

Figure 43 - Sigfox messages sent when the type of message is '1' (no movement)

5.5 Additional work

During the last week of my internship, I did another program that I will briefly explain. The aim of this program is to send information about the device, which are its GPS coordinates, its battery level and the temperature (like in the precedent program). At the same time, it prints information about accelerometer if the user decides to use serial communication. It will print x, y and z values. These data will appear 3 minutes after the launching of the program. A keepalive message is sent at every boot.

For transceiving information on the Sigfox network, this program uses the same format as the precedent program (see part 5.4.1. *General presentation*).

Users can change some parameters using serial communication. Indeed, it is possible to communicate with the module, like in the precedent project. Commands must be typed 3 minutes after the launching of the program.

You will see below the different types of commands and their possible values:

AT\$ACCFREQ=

This command is used for choosing the frequency to retrieve accelerometer data. When there is nothing written after this command, it prints the current value. Users can change this frequency by adding a number after this command. The different possibilities are:

- 1: 1 Hz.
- 2: 10 Hz.
- 3: 25 Hz.
- 4: 50 Hz.
- 5: 100 Hz.
- 6: 200 Hz.
- 7: 400 Hz.
- 8: 1.25 kHz.

The default value is 1 (1 Hz).

AT\$ACCSCALE=

This command is used for choosing the scale to detect movement. This value has an impact on the accuracy of the accelerometer data. When there is nothing written after this command, it prints the current value. Users can change this scale by adding a number after this command. The different possibilities are:

- 1: 2G.
- 2: 4G.
- 3: 8G.
- 4: 16G.

The default value is 1 (2G).

AT\$ACCDATA=

Thanks to this command, users have the possibility to choose if accelerometer data has to be printed on the serial console. When there is nothing written after this command, it prints the current configuration. The two possible values after this command are:

- 0: x, y and z values will not be printed on the serial console.
- 1: x, y and z values will be printed on the serial console. This display will be stopped a few moments when the module sends a message on the Sigfox network.

This command can be used at any time. Indeed, it is possible to change its value during the 3 minutes after the launching of the program, and after these 3 minutes too. The default value is 1.

AT\$INTERVAL=

Like in the precedent program, this interval is the time between two consecutive messages sent on the Sigfox network. When there is nothing written after this command, it prints the current value. When there is a positive number after this value, this value become the new interval in seconds. The default value is 3600 seconds.

AT\$MODE=

This command is used for choosing the GPS mode. The possible values are:

- 0: *TD_GEOLOC_OFF*
- 1: *TD_GEOLOC_HW_BCKP*
- 2: *TD_GEOLOC_POWER_SAVE_MODE*

To have more information about these different modes, see part 5.4.2. *GPS mode*. The default mode is *TD_GEOLOC_HW_BCKP*.

AT\$TIMEOUT=

Thanks to this command, it is possible to choose the timeout, like in the precedent project. This data is the duration which the GPS tries to find satellites in order to have GPS coordinates. When there is nothing after this command, it prints the current value. When there is a positive number, this value will be the new timeout. The default value is 120 seconds.

The code is this program (*acc_data_gps.c*) can be retrieved on this Github account:

https://github.com/JulienPeudennier/sigfox_examples

6. Project management

This is a Gantt chart which represents the project organisation. With this tool, it is possible to see the general tasks of the project and their duration.

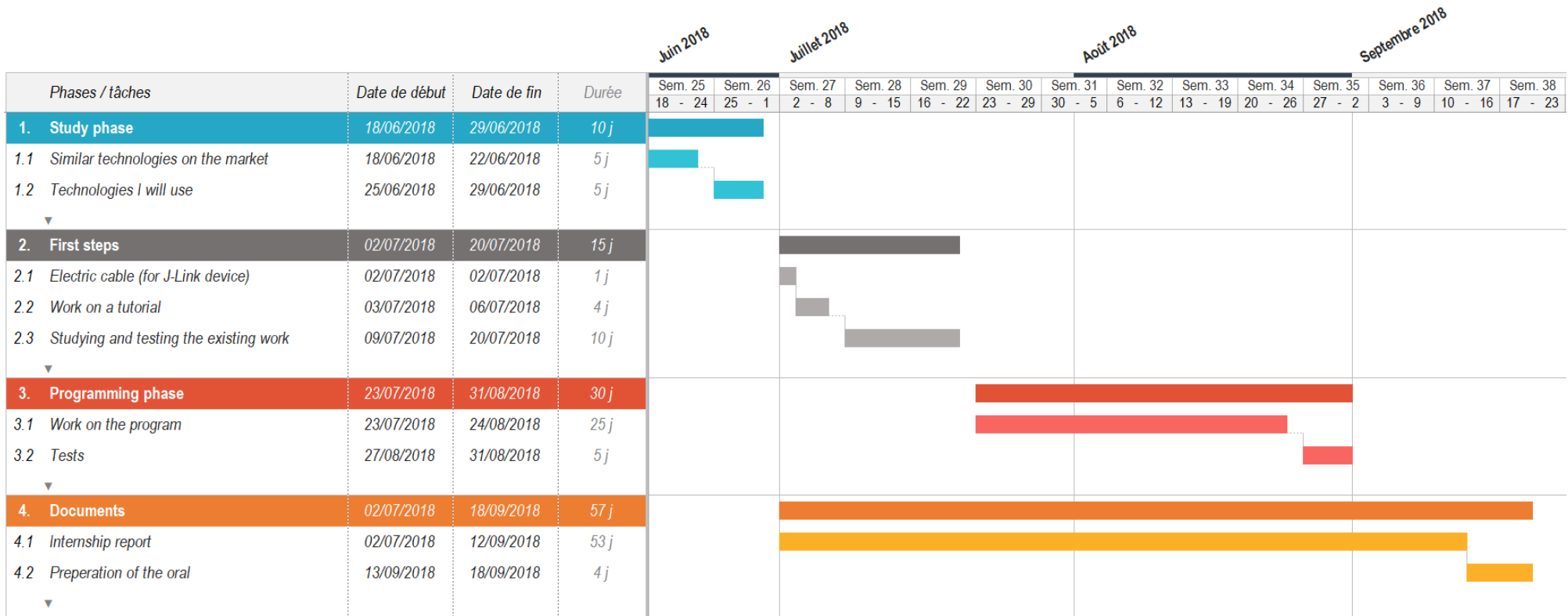


Figure 44 - Gantt chart

7. Conclusion

The final aim of this project was to create a low power GPS tracker with movement detection. For that, the program had to detect if the device was in motion. If there was movement, the module had to send some information every T1 period. If there was no movement, information was sent every T2 period (which was longer than T1 period). The objective of this configuration was to gain consumption. Indeed, it was not necessary to have data every hour for example if the device was not in motion...

This work was done using the TD1205P module and Sigfox communication. A program was realized on Eclipse, using C language, in order to satisfy demand. In accordance with the supervisor, the messages which are sent by the device had to include: GPS coordinates of the device, battery level and temperature.

Users can use serial communication for changing some parameters with specific commands, like the interval between two messages when the device is in motion (T1 period) for example. All the possible commands are defined in this report. Data can be observed using Sigfox backend.

Thanks to this work, we could imagine that the device could be deployed into a buoy. For example, it could be placed into a buoy which contains sensors for monitoring currents. If the buoy moves due to the marine currents, the device will send the location of the buoy more regularly. If it doesn't move, it is not necessary to send data regularly, because it will be the same location. For this reason, it will send data at larger intervals, so it will save energy.

Concerning the perspectives, maybe it could be a good idea to add other commands for configuring the device or printing other information when there is a serial communication. Moreover, I think that it could be great to find interesting other information to send. Indeed, only ten bytes are sent in this application, but Sigfox messages can contain twelve bytes of data.

8. References

Information about UPC and the SARTI centre:

- (1) https://cit.upc.edu/en/upc_centers/place/45/sarti_upc
- (2) <http://www.cdsarti.org/>
- (3) <http://www.obsea.es/>

Kaloyan Ganchev's work:

- (4) https://github.com/kaganchev/sigfox_examples

Sigfox GPS Tracker tutorial:

- (5) <https://www.instructables.com/id/Sigfox-GPS-Tracker/>

Information about Chipfox GPS Tracker:

- (6) <https://partners.sigfox.com/products/chipfox>
- (7) <http://chipfox.cz/>

Information about Oyster:

- (8) <https://www.digitalmatter.com/Devices/Lorawan-IOT/Oyster>

Information about Sigfox:

- (9) <https://www.link-labs.com/blog/sigfox-vs-lora>
- (10) <https://www.sigfox.com/en>
- (11) <https://en.wikipedia.org/wiki/LPWAN>
- (12) <http://www.rfwireless-world.com/Tutorials/advantages-and-disadvantages-of-Sigfox-wireless-technology.html>
- (13) <https://www.link-labs.com/blog/what-is-sigfox>
- (14) <https://internetofthingsagenda.techtarget.com/definition/LPWAN-low-power-wide-area-network>

Information about the TD1205P module:

- (15) <http://rfmodules.td-next.com/modules/td1205/>

- (16) <https://s3-eu-west-1.amazonaws.com/assetstdnext/download/TD1205P+Datashort.pdf>
- (17) <https://www.avnet.com/wps/portal/silica/products/product-highlights/2016/td-next-td1205p/>

Information about the EFM-32 Tiny Gecko:

- (18) <https://www.digikey.com/en/product-highlight/s/silicon-laboratories/efm32-tiny-gecko>

Information about the TTL-232RG cable:

- (19) <https://docs-emea.rs-online.com/webdocs/0f73/0900766b80f73fab.pdf>

9. Summary

This report was done during my internship in the SARTI centre in Vilanova i la Geltru. I realized this internship as part of my studies at ISEN Brest, an engineering school in France. This report deals with the project I worked on.

The aim of this project was to develop a low power application based on the TD1205P module. The final application of this work is a GPS tracker with movement detection. It means that the module sends some information when it is in motion, and other information when it doesn't move. The device sends data on a more regular basis if it moves. These data can contain the GPS coordinates of the device, its battery level and the temperature. The device could be placed into a buoy for example. In this case, the device will send data more regularly if the buoy moves, due to marine currents. It will not send data regularly when the buoy is not in movement, so it will save energy.

This document tries to trace the progression of my work. It mentions the different technologies and tools which were used in this project, as well as the choices which were made. Finally, it evokes the technical development which was necessary to attain the objective set.

Ce rapport a été réalisé pendant mon stage dans le centre de développement technologique pour les systèmes d'acquisition et de traitement de l'information à distance (SARTI), à Vilanova i la Geltru. J'ai réalisé ce stage dans le cadre de mes études à l'ISEN Brest, une école d'ingénieurs située en France. Ce rapport traite du projet sur lequel j'ai travaillé.

Le but de ce projet était de développer une application à faible consommation énergétique basée sur le module TD1205P. Le résultat final de ce travail est la réalisation d'un tracker GPS capable de détecter les mouvements. Cela signifie qu'il peut envoyer des informations différentes selon qu'il est en mouvement ou non. Les données sont envoyées plus régulièrement dans le cas où il est en mouvement, et peuvent contenir les coordonnées GPS du module, son niveau de batterie ou encore la température. Ce module peut par exemple être placé à l'intérieur d'une bouée. Dans ce cas précis, il pourra envoyer des données plus régulièrement lorsque la bouée est en mouvement en raison des courants marins. Les données seront envoyées moins régulièrement lorsque la bouée n'est pas en mouvement, ce qui permettra d'économiser de l'énergie.

Ce document tente de retracer la progression du travail réalisé. Il évoque les différentes technologies utilisées dans le cadre de ce projet, ainsi que les choix qui ont été faits. Enfin, il décrit le développement technique qui a été nécessaire pour atteindre l'objectif fixé.