

POLYTECH UNIVERSITY SCHOOL OF MONTPELLIER

**Industrial Project of the End of Studies
Electronic, Robotic and Industrial Informatic
Departement**

YEAR 2018-2019

PROJECT

Paul LELOUP

**THE IOT LORA PROJECT
The Smart Sensor**



ECOLE POLYTECHNIQUE UNIVERSITAIRE DE MONTPELLIER
UNIVERSITE MONTPELLIER II SCIENCES ET TECHNIQUES DU LANGUEDOC
Place Eugène Bataillon 34095 MONTPELLIER CEDEX 5
Tél. : 04 67 14 31 60 – Fax : 04 67 14 45 14
E-mail : scola@polytech.univ-montp2.fr



1. Summary

2.	Acknowledgements	2
3.	Introduction	3
4.	Specifications	4
4.1.	Suject / Goal	4
4.2.	System	6
4.3.	Hardware.....	8
4.4.	Software	9
5.	System.....	10
5.1.	Design of the card.....	10
5.2.	LoRa.....	11
5.3.	Pmod Sensors	11
6.	Hardware (FPGA).....	12
6.1.	System	12
6.2.	Processor	12
6.1.	UART, SPI, I ² C Interfaces	13
7.	Software (Flow chart)	14
7.1.	GPS	14
7.2.	HYGRO	15
7.3.	ALS	16
7.4.	LoRa	17
7.5.	Smart Sensor	18
8.	Result	19
8.1.	Flex Node.....	19
8.2.	Terminal	19
8.3.	Test of consumption.....	20
9.	Conclusion	22
10.	Reference	23
11.	Appendix	24

2. Acknowledgements

I thank the Laboratory of Computer Science, Robotics and Microelectronics of Montpellier (LIRMM) for having welcomed me during this Industrial End of Study Project (PIFE).

I would especially like to thank Pascal Benoit, an university lecturer at Polytech and LIRMM, who gave me his confidence to carry out this project and who supervised me every week, as well as Guillaume Patrigeon, PhD at the LIRMM, which helped me greatly in the installation of the workstation and throughout these 200 hours of work.

To do my PIFE at the LIRMM was a pleasure, I could learn a lot thanks to you and to validate my acquired of my curriculum in engineering school.

3. Introduction

The number of connected devices (Internet of Things: IoT) in circulation in the world in 2018 is 23.14 billion [1] and in the next few years the number of devices could reach between 50 and 80 billion. It is estimated that connected devices worldwide in 2013 consumed 616 Twh [2], which is greater than the consumption of France in one year (eg 544 Twh in 2015 [3]). The saving of electric current represents billions of euros if the consumption of these objects was decrease.

The project is part of an industrial end-of-course project which aims to put the student in the position of a project manager responsible for finding solutions to a concrete problem that is posed to him. The title of PIFE is "Study & Design of a Smart Sensor on FPGA".

The IoT project purpose is very interesting to me, and designing a smart object as well as programming and participating in its improvement is relevant because I want to work in this field.

The Internet of Things (IoT, Figure 1) is the extension of the internet to things and places in the physical world. The connected Internet of Things represents the exchange of information and data from real-world devices with the Internet.

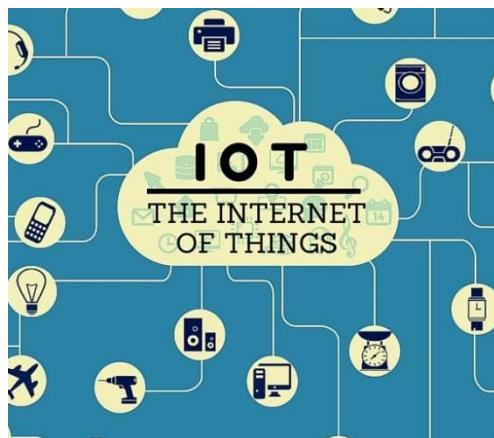


Figure 1 Illustration IoT

The place of study is the Laboratory of Informatics, Robotics and Microelectronics of Montpellier (LIRMM, Figure 2) which is a joint research unit jointly dependent on the University Montpellier (UM) and the National Center for Scientific Research (CNRS).



Figure 2 LIRMM photo

4. Specifications

4.1. Subject / Goal

Présentation of the project:

The project was proposed by Pascal Benoit, because as part of research work carried out at the LIRMM on the theme of the Internet of Things, researchers develop an experimental platform consisting of connected objects, network gateways, as well as servers. for the processing and visualization of collected data.

For this, they work on a prototype Smart Sensor: it is based on an FPGA card, to which any type of sensor and radio can be connected. This architecture allows rapid prototyping of a connected sensor architecture: for example, one may want to test and compare two types of radio transmission, or configure in the FPGA a Cortex M0 processor or a Cortex M3, or play on the size of embedded memory, all this in order to evaluate the various solutions, and to be able to compare them in terms of performance and energy.

This Smart Sensor prototype is based on a DIGILENT CMODA7 card for which a complete microcontroller architecture is described in VHDL is available: it is composed of a Cortex M0 processor, an internal memory, and various peripherals. In order to quickly prototype various types of technologies we want to create a port expansion card that will connect different types of radio modules and sensors.

The objective of the project is to create a test bench to calculate the consumption and performance of a smart object to optimize them. This is to help Guillaume Patrigeon's thesis on "Ultra-low power adaptive integrated systems for the Internet of Things".

Firstly, it will be necessary to make a bibliographic study of the different types of sensors and radio modules possible, and to specify several cases of use.

Then I will create an electronic board extension ports that allows you to connect different sensors and radio modules in order to customize the smart object. This card will be the Smart Sensor, it will also be called the sensor node. It will connect the CMOD A7, radio modules and sensors.

Once the card is created, there will be a software part that will capture the information from the sensors, which will process them and then send them by radio.

In the end, when the test bench will be functional, we will be in a real application of intelligent object. There will be an architectural exploration later to lower the consumption of microcontrollers implemented in connected objects.

Here are the specifications for the work to be done during this Industrial End of Study Project.

Specifications :

- Make a bibliographic study of the different types of sensors and radio modules possible, and specify several use cases
- Operate the GPS module and send the data received by LoRa with the Cmod A7
- Make a Smart Sensor electronic card that will have these features:
 - Modular thanks to Pmod connectors
 - Possibility to connect at least 5 modules (sensor, radio modules)
 - Reconfigurable and flexible thanks to the Cmod A7 FPGA
 - Reprogrammable by micro USB easily accessible
 - Battery powered for outdoor testing and less power supply noise than USB
 - Easy to measure the consumption and power of each module
 - Voltage regulator and current limiter system to secure the components of the board.
- Make the datasheet of the Smart Sensor card
- Make a Pmod LoRa electronic card that will have these features:
 - Modular thanks to a Pmod connection
 - Possibility to connect a LoRa module
 - Possibility to connect an antenna
 - Easy to measure the power consumption and power of the module
- Make the datasheet of the Pmod LoRa card
- Option: Add temperature, humidity and brightness sensors and program them with the electronic board created to send the data by LoRa
- Option: Add a Bluetooth radio module and send the created frame with it
- Option: Make a consumption measurement of the Smart Sensor

Planification des tâches

See the Gantt created at the beginning of the project in Appendix (Gantt).

4.2. System

The system has the same architecture as a connected object, the Smart Sensor architecture is like the diagram below (Figure 3).

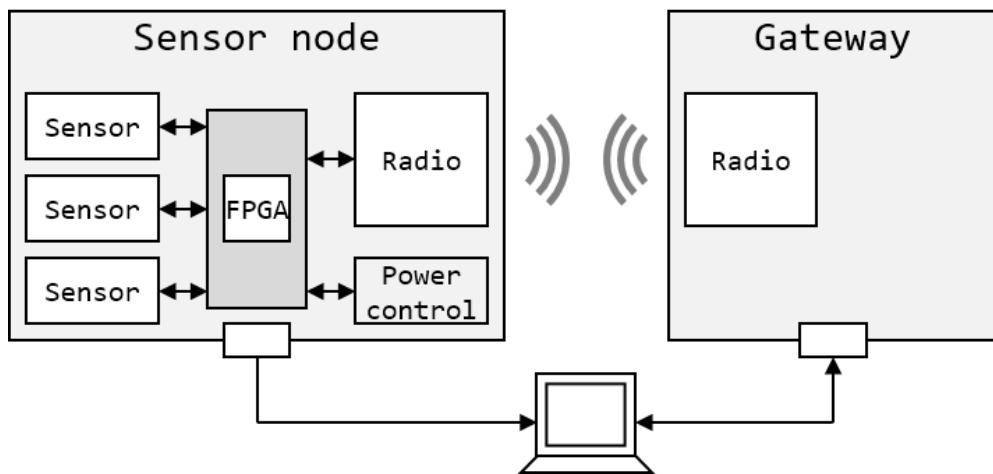


Figure 3 Architecture of Smart Sensor and his Gateway

The card consists of a microcontroller (implemented in an FPGA, the Cmod A7) that will communicate with modules that may be sensors or radio communication systems. Part for battery power is also present.

The particularity of the Smart Sensor that makes it a smart object like no other is that it is modular and therefore reconfigurable. So all possible sensors and radios can be connected to the board if they have Pmod connectivity. Here there are a picture of sensors and radio modules with the Pmod connection (Figure 4)



Figure 4 Pmod modules

We chose to use this connection because there are a very large number of sensors with it. It is also common for prototyping quickly.

These modules also have the advantage of never using analog pins, they communicate with the microcontroller using communication interfaces such as UART, SPI, I2C and GPIO.

Here is the diagram of the Smart Sensor (Figure 5). Each module has Pmod connectivity and they are all interchangeable.

It will just fit the code and indicate the change of the pins.

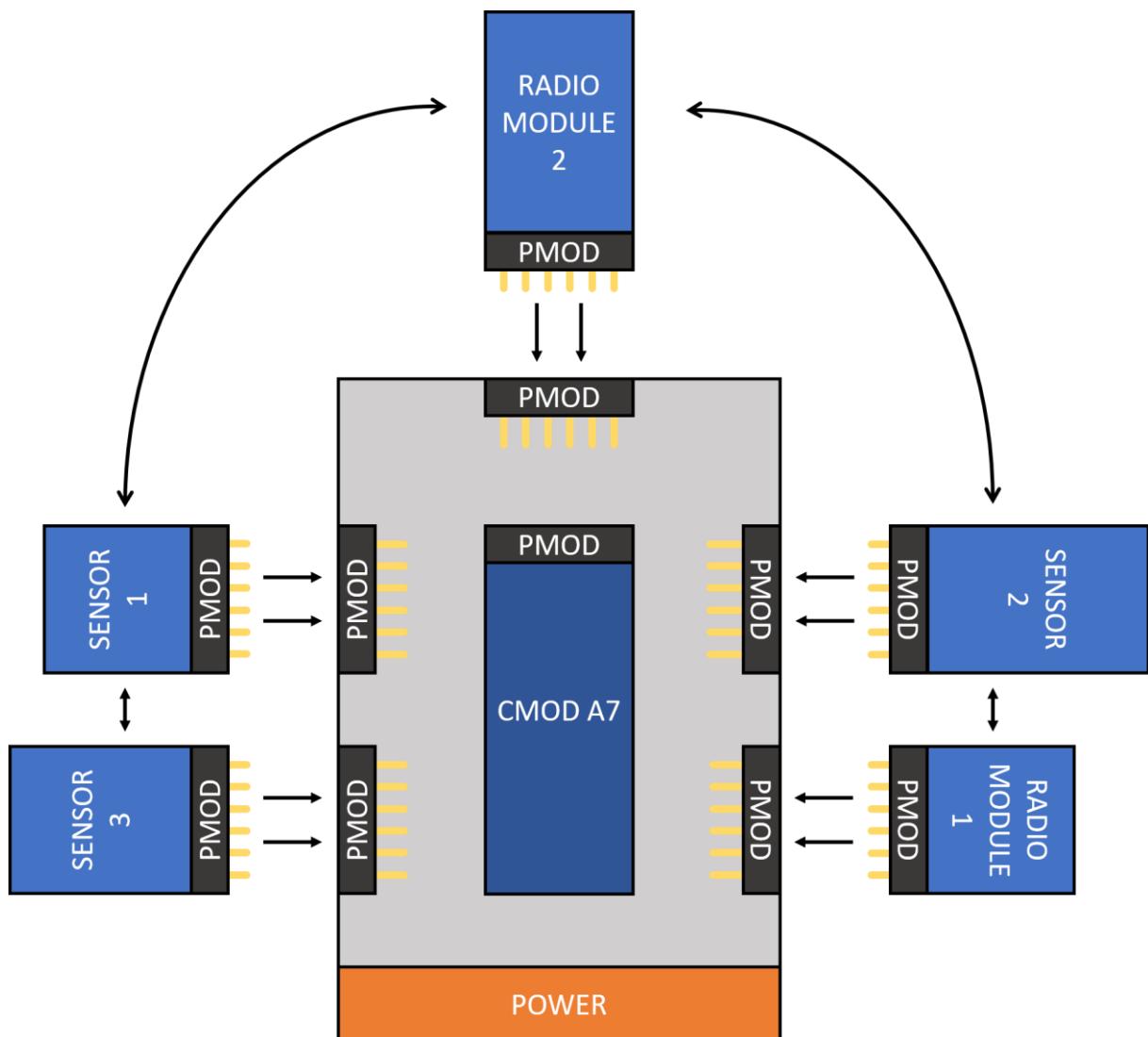


Figure 5 Smart Sensor modulaire

In terms of power, the Smart Sensor is powered by either USB-microUSB cable or battery. The two power systems each have their advantages:

- The USB-microUSB power cable allows you to upload the C code to modify the program. It also allows software development because you can view a UART on the terminal of the computer. This makes it easy to debug and view the data sent and received from the Gateway.
- Battery power has very little noise (high frequencies) and allows for precise and noise-free consumption measurements unlike USB generated by a computer. There is also the advantage of mobility, which allows for scope testing with the Gateway.

4.3. Hardware

The main controller of the node is the Digilent CmodA7. It is a compact 48-pin DIP card that features an Artix 7 FPGA from Xilinx. This is the version of the CmodA7 35T that will be used here. As the CmodA7 already has pushbuttons and light-emitting diodes, it will not be necessary to overload the motherboard of the sensor node with these elements.

In our application, the first need is that the hardware is reconfigurable. Thanks to the flexibility of the FPGA, it is possible to explore different architectures and therefore different processor solutions (for example: ARM Cortex-M0 and Cortex-M3, LIRMM's SecretBlaze, etc ...). It is also possible to replace this card with another, with a microcontroller trade to compare the proposed solutions with existing products. This study of microcontroller architectures is part of other projects within the LIRMM.

Here are the pros and cons of using an FPGA chip:

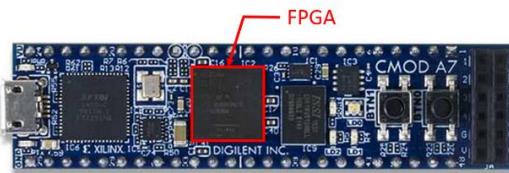


Figure 6 CMOD A7 card of DIGILENT

Advantages	Disadvantages
<ul style="list-style-type: none"> Possibility of prototyping Flexibility Time-to-market weak Adaptability to future evolutions thanks to the reconfiguration 	<ul style="list-style-type: none"> Limited integration by routing resources Performance High unit price for large productions

What interests us the most is the possibility of prototyping and flexibility. Although the Artix 7 FPGA present is limited in terms of performance and resources, the targeted architectures are those of very low-power microcontrollers operating at relatively low frequencies (order of ten MHz) and with a reduced number of peripherals. The limited integration problem due to routing resources has been eliminated because the system is modular with Pmod connectivity and does not require analog pins.

We will integrate in the FPGA chip a microcontroller with ARM cortex-m0 processor. To integrate a microcontroller into an FPGA, it must be described as a VHDL file (.VHD) and follow all the steps summarized in this diagram (Figure 7).

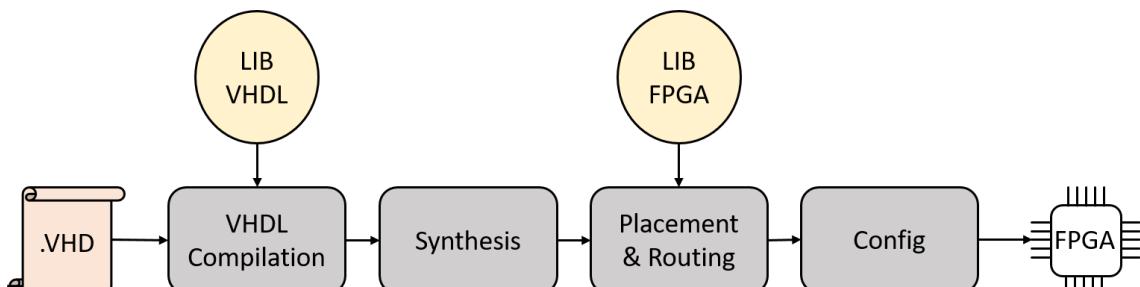


Figure 7 Integrate a microcontroleur in a FPGA

4.4. Software

The software is reprogrammable as in most cases. It allows to configure all the GPIO input and output, configure up to 12 communication interfaces (4 UART, 4 SPI and 4 I2C) each pin of these interfaces can be chosen by software.

At the level of the global structure of the code, here is the cyclic functioning of the code (Figure 8).

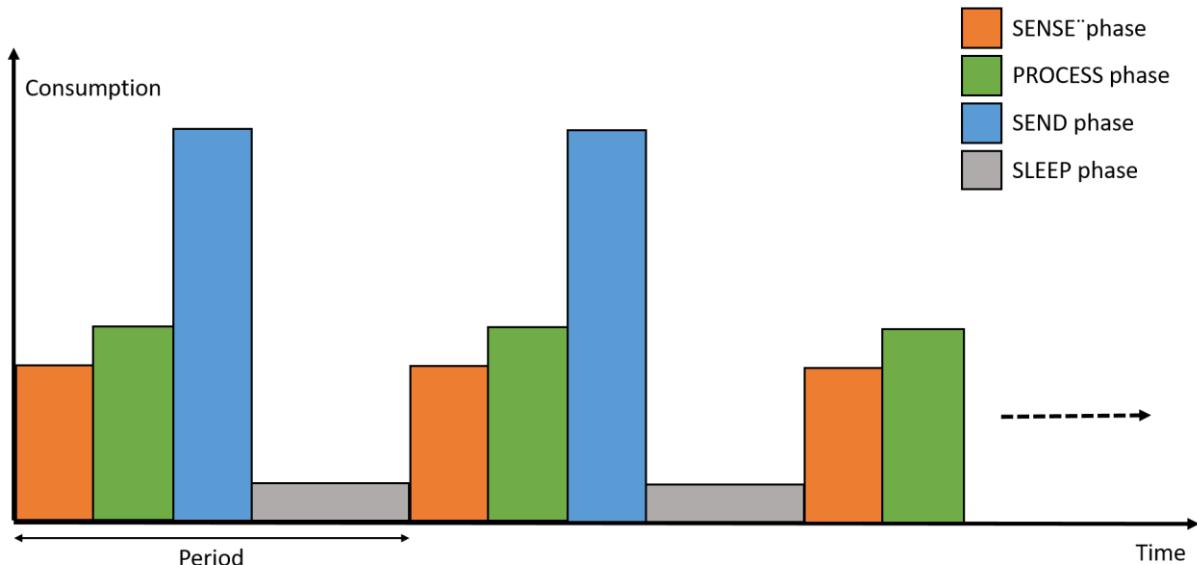


Figure 8 Differents steps of the software

It is composed by 4 parts that are:

- The SENSE phase: It allows to collect the information of the sensors and stores them in a variable.
- The PROCESS phase: It processes the bytes recorded by the previous phase in order to extract the desired information. This information is formatted and grouped in a concise message (a frame to send).
- The SEND phase: This phase sends the previously created frame with one or more radio modules (eg LoRa, BLE).
- SLEEP phase: It puts the microcontroller in sleep mode to save energy because sending the temperature every 5 ms would not be relevant in this application.

5. System

5.1. Design of the card

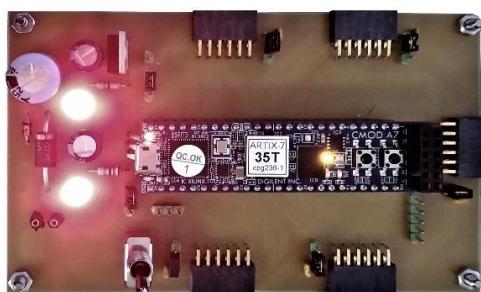


Figure 9 Smart Sensor Card

Electronic card

For more information on the Smart Sensor map, please go to the appendices to view the Smart Card Datasheet. The schematic of the board as well as the board is also in the datasheet.

The electronic board (Figure 9) consists of 5 female Pmod connectors where each pin of each connector is connected to CMOD A7 which is the component in the center of it.

It was chosen a power system to regulate the battery voltage. The voltage of the battery is 7.4V, a voltage higher than 5V to be able to supply all the components of the circuit. There is a first linear voltage regulator to lower the voltage from 7.4 to 5V. 5V is the voltage required by the CMOD A7 to operate. Since there is no 3.3V power pin on the CMOD A7 board, another linear regulator has been placed at 5V to generate a 3.3V supply voltage for all Pmod connectors. Since linear regulators dissipate the extra voltage into heat, it was more appropriate to put them in series to avoid overheating. There are also different components ensuring the robustness of the power supply and the low noise:

- A Schottky diode in series to prevent reverse currents and not to degrade the battery.
- A Zener diode in parallel to have the most constant supply voltage possible.
- Large capacitors in parallel to fully respond to current calls from the Smart Sensor.
- LEDs to visualize the 5V and 3.3V power supply.
- Small capacitors in parallel to limit the high frequencies on the power supply as close as possible to the output pins of the power supply.

I have also positioned pins that allow measurement of current and voltage simultaneously (power). This makes it possible to measure the consumption of each Pmod separately, of all the Pmods together or of the CMOD A7.

Problèmes rencontrés

- Size of the holes for the components to check well (All the holes of the pins of connectors must be in 1mm, the regulators of tension, the connections Pmod also).
- The gap between the two CMOD A7 pin rails was not good.
- The battery connector is in CMS, so on the back, specify it in the software otherwise symmetry problem between VCC and GND.

As solutions to these problems it was made a second version of the card of which no problem has been detected until today.

5.2. LoRa



Figure 10 LoRa Pmod

For more information on the Pmod LoRa card, please go to the appendices to view the map datasheet (Datasheet Pmod LoRa). The schematic of the board as well as the board is also in the datasheet.

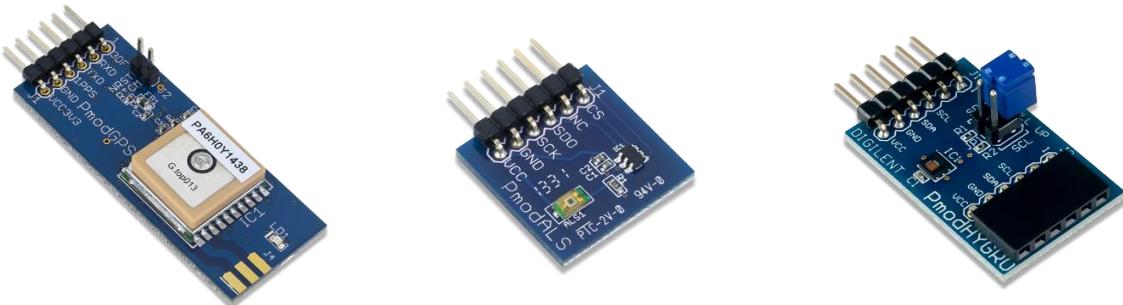
The electronic board (Figure 10) consists of a male Pmod connector where each pin is connected to LoRa RF-LoRa-868-SO module. The 5 extra pins are wired to a single connector.

An 868 MHz SMA antenna can be easily connected using the added connectors.

There is also the same system of measurement of consumption that on the map of the Smart Sensor.

This LoRa module was first used on a breadboard and without an antenna and thanks to this card, the use of it is much easier. It communicates by SPI with the microcontroller.

5.3. Pmod Sensors



(a) GPS Pmod

(b) ALS Pmod

(c) HYGRO Pmod

Figure 11 Differents Pmod used

This GPS Pmod (Figure 11 (a)) was the first sensor used with the CMOD A7, it is easy to use and uses the UART to communicate. It has also been used on breadboard.

The ALS Pmod (Figure 11 (b)) captures the brightness, it uses the SPI interface as well.

The latest Pmod sensor is the HYGRO Pmod (Figure 11 (c)), which captures air humidity and temperature. The one is different and uses the I2C interface.

All these sensors will make interesting measurements of consumption by the actual application of the Smart Sensor.

6. Hardware (FPGA)

6.1. System

The system has a global structure as below (Figure 12), it is the typical structure of a microcontroller.

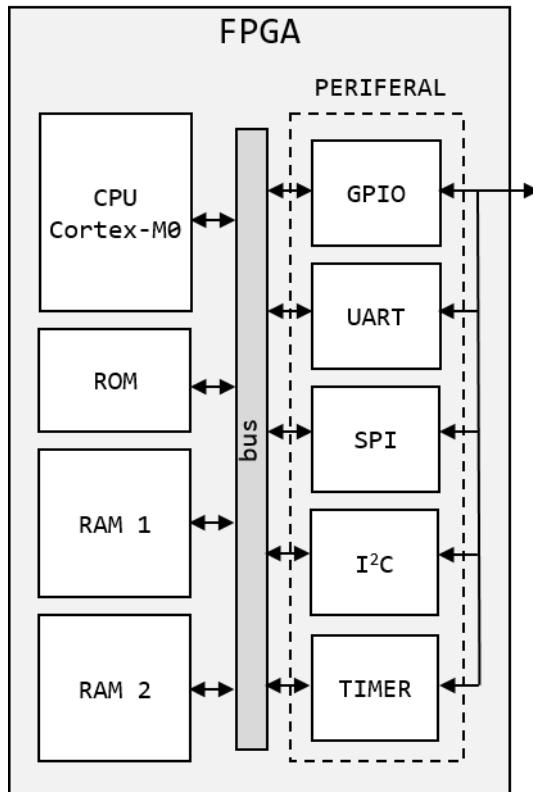


Figure 12 global Structure (FPGA)

So we have inside the FPGA chip a microcontroller composed of a processor, the Cortex-M0 whose specificities will be detailed in the next part, a ROM, two RAM and various peripherals. The TIMER will mainly serve to set a cycle of a constant period with interruption to release the microcontroller from its standby state.

6.2. Processor

The processor used in this study is the Cortex-M0 r1p0, from the well-known ARM Cortex-M series. Products based on this processor can be found to validate and compare the operation of the system implemented on the FPGA. The Cortex-M0 is a 3-stage 32-bit RISC processor that implements the ARMv6-M instruction set. It is given for a maximum frequency of 50 MHz, has 32 interrupt lines, a non-maskable interrupt line and a 32-bit single cycle multiplier in its r1p0 version. It has only one hardware interface, type AHB-Lite.

Here is the diagram of the functional blocks of the processor (Figure 13).

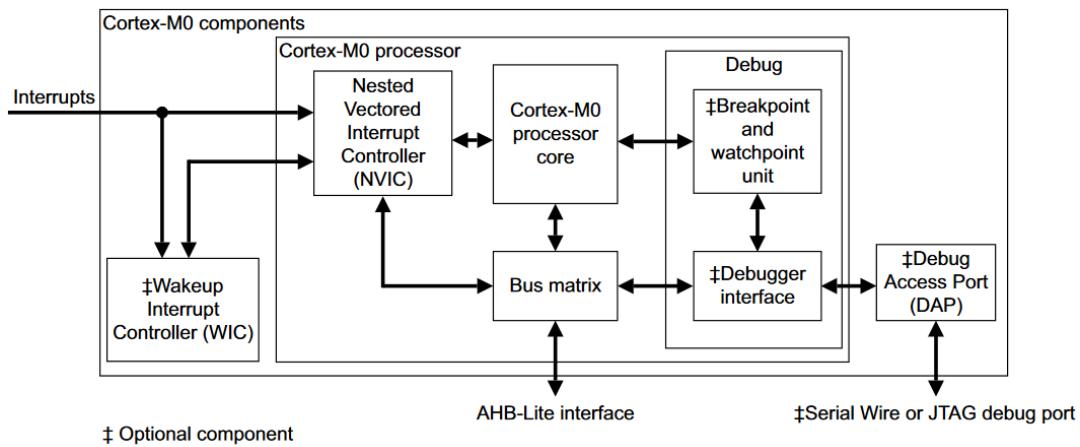


Figure 13 Functional block diagrams of the Cortex-M0

6.1. UART, SPI, I²C Interfaces

To communicate between the modules and the microcontroller, we will use communication interfaces such as UART, I²C and SPI.

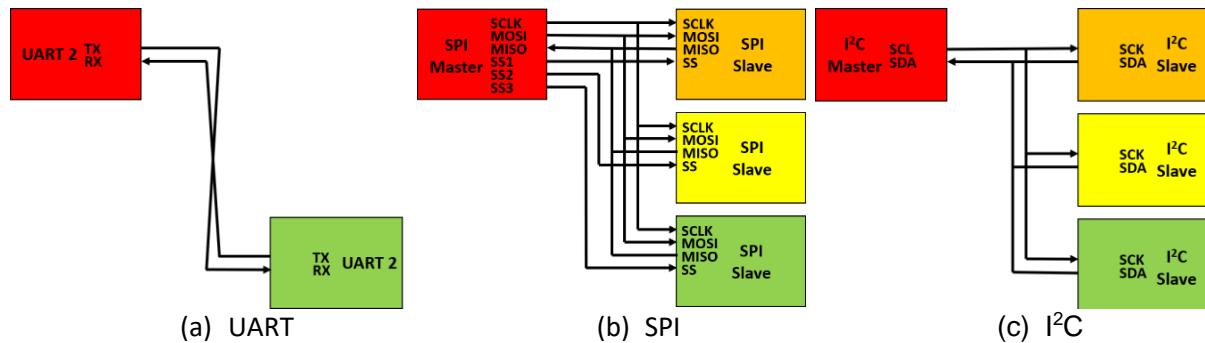


Figure 14 Communications interfaces

A UART (Figure 14 (a)), for Universal Asynchronous Receiver Transmitter, is a universal asynchronous transceiver.

An SPI link (Figure 14 (b)) (for Serial Peripheral Interface) is a synchronous serial data bus. The circuits communicate according to a master-slave scheme, where the master controls the communication. Several slaves can coexist on the same bus, in this case, the selection of the recipient is done by a dedicated line between the master and the slave called Slave Select.

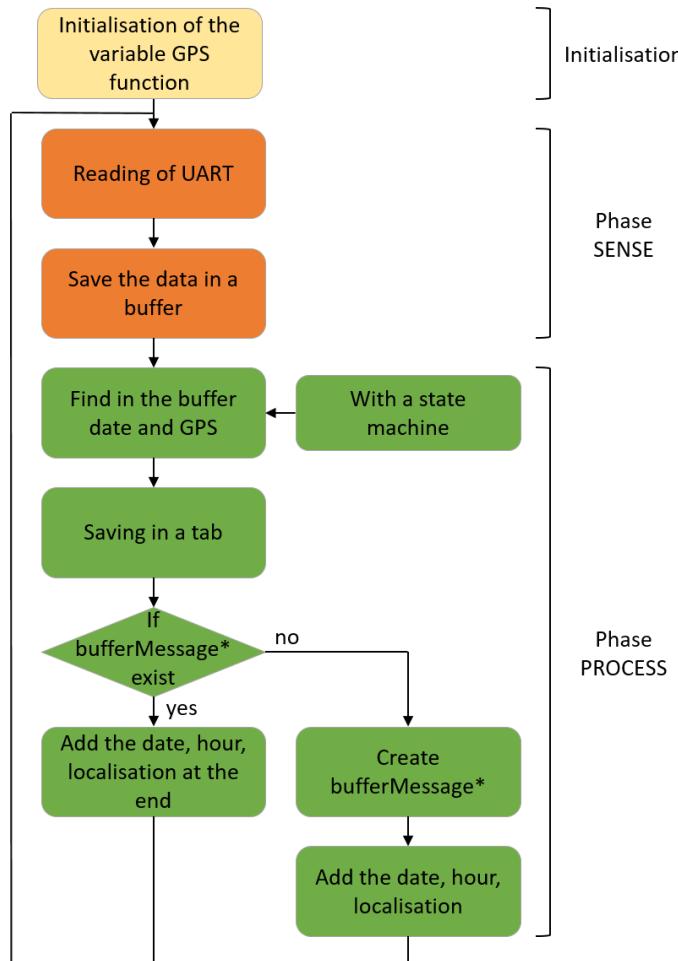
I²C (Figure 14 (c)) is a half-duplex bidirectional synchronous serial bus, where multiple devices, masters or slaves, can be connected to the bus.

UARTs, SPIs, and I²Cs are usually integrated into components like microcontrollers. In this case they are a peripheral function of the microcontroller.

7. Software (Flow chart)

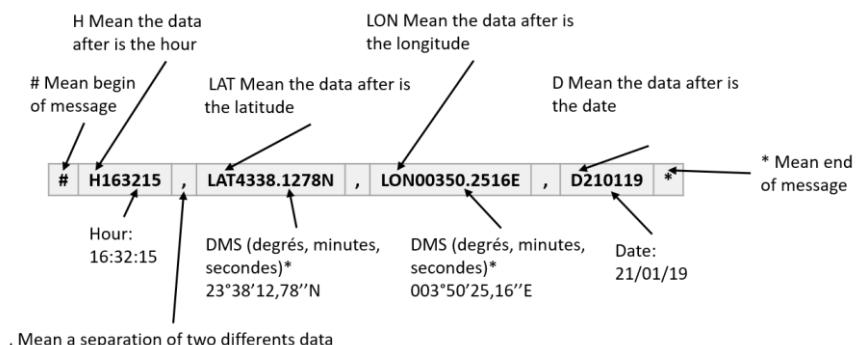
7.1. GPS

The flowchart below (Figure 15) shows 3 functions, the "initGPS" function that initializes the Pmod, the "readGPS" function (Phase SENSE) and the "processMessageGPS" function (Phase PROCESS) that creates the frame (Figure 16).



*bufferMessage : It's the tab where we save data for send by radio.

Figure 15 Flow chart of the GPS function

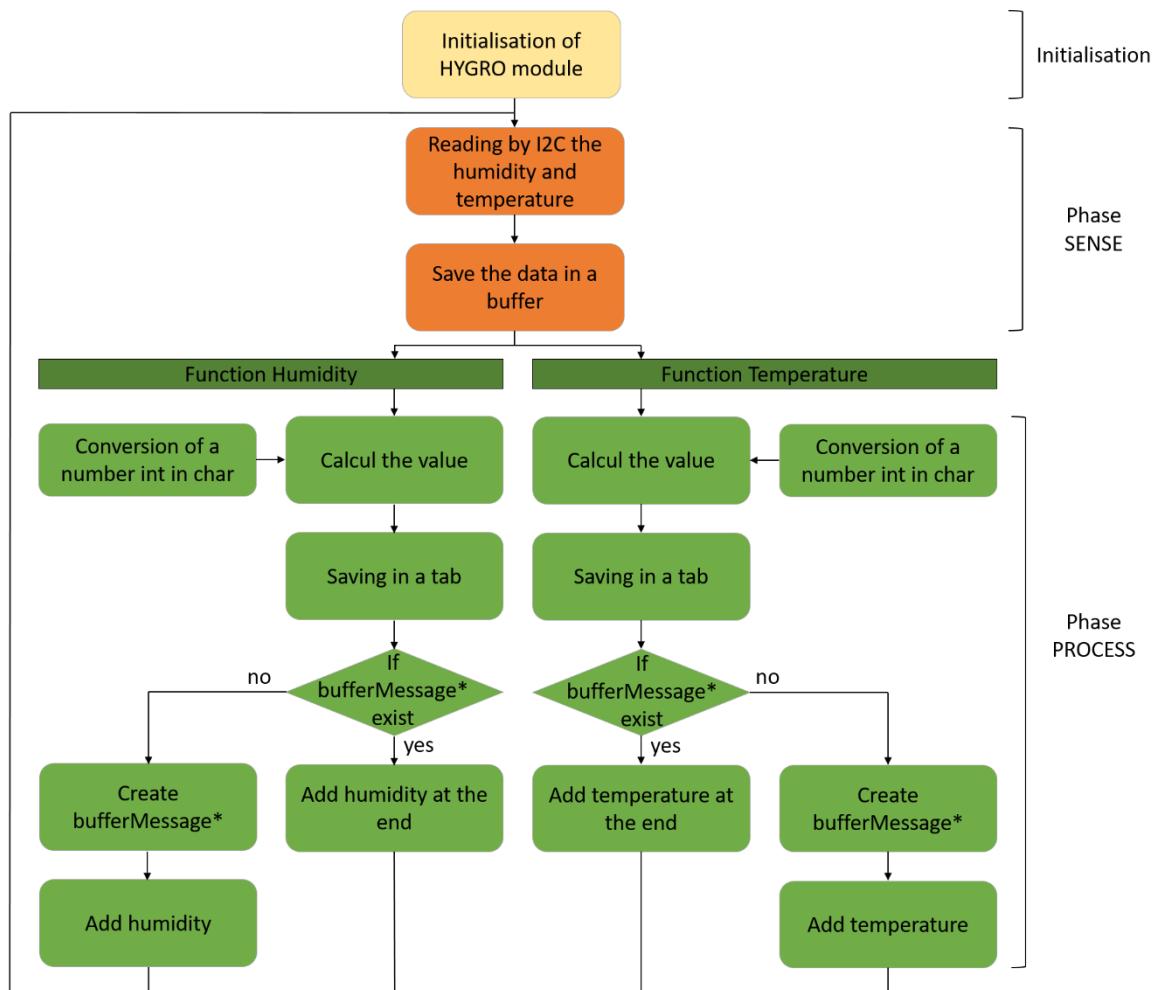


*Géodesic WGS 84 System

Figure 16 Function GPS frame

7.2. HYGRO

As the flow chart shows (Figure 17), there are going to be 4 functions here. The "initHYGRO" function to initialize the Pmod, the "readHYGRO" function that captures the humidity and temperature, then the last two functions "processMessageHumidityHYGRO" and "processMessageTemperatureHYGRO" which records and processes the raw data of the humidity and the temperature to create the frames (Figure 18, Figure 19).



*bufferMessage : It's the tab where we save data for send by radio.

Figure 17 Flow chart of the HYGRO function

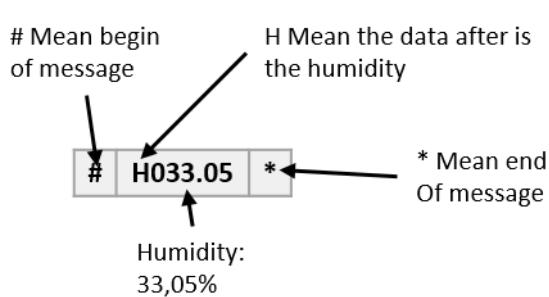


Figure 18 Function Humidity HYGRO frame

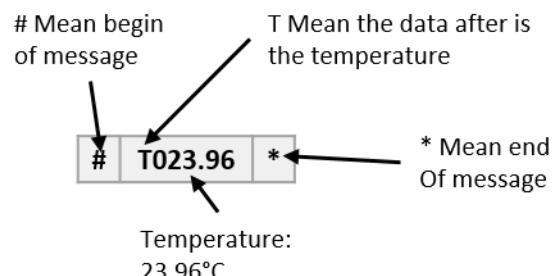
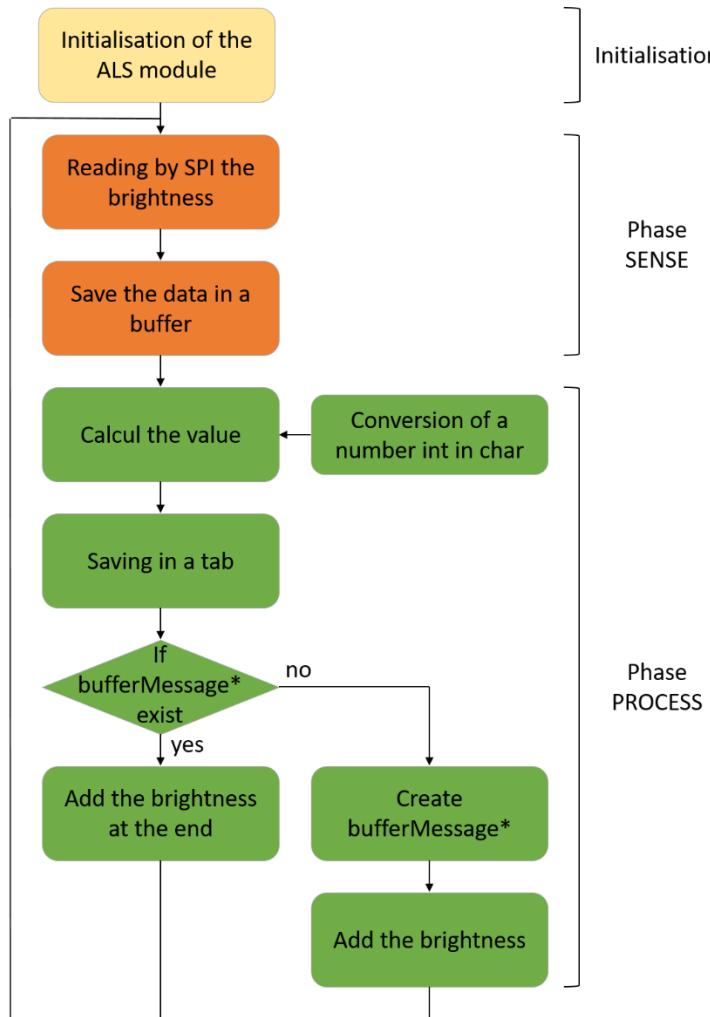


Figure 19 Function Temperature HYGRO frame

7.3. ALS

Here is the flow chart of the Pmod ALS, there is a function for the initialization, the phase SENSE of capturing raw data of the sensor and the function of the phase PROCESS which transforms the data.

In all functions of type "processMessage", there is the particularity that if the bufferMessage array is empty, they fill it with their data. Otherwise they add their data at the end of the table. This means that if we interchange the functions between them we will be able to create the frame in the desired order.



*bufferMessage : It's the tab where we save data for send by radio.

Figure 20 Flow chart of the ALS function

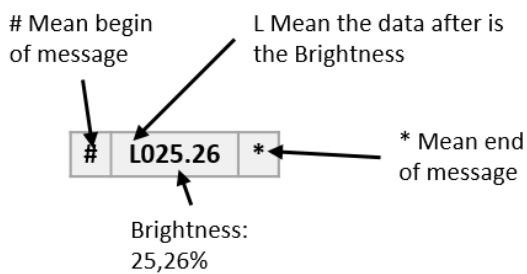


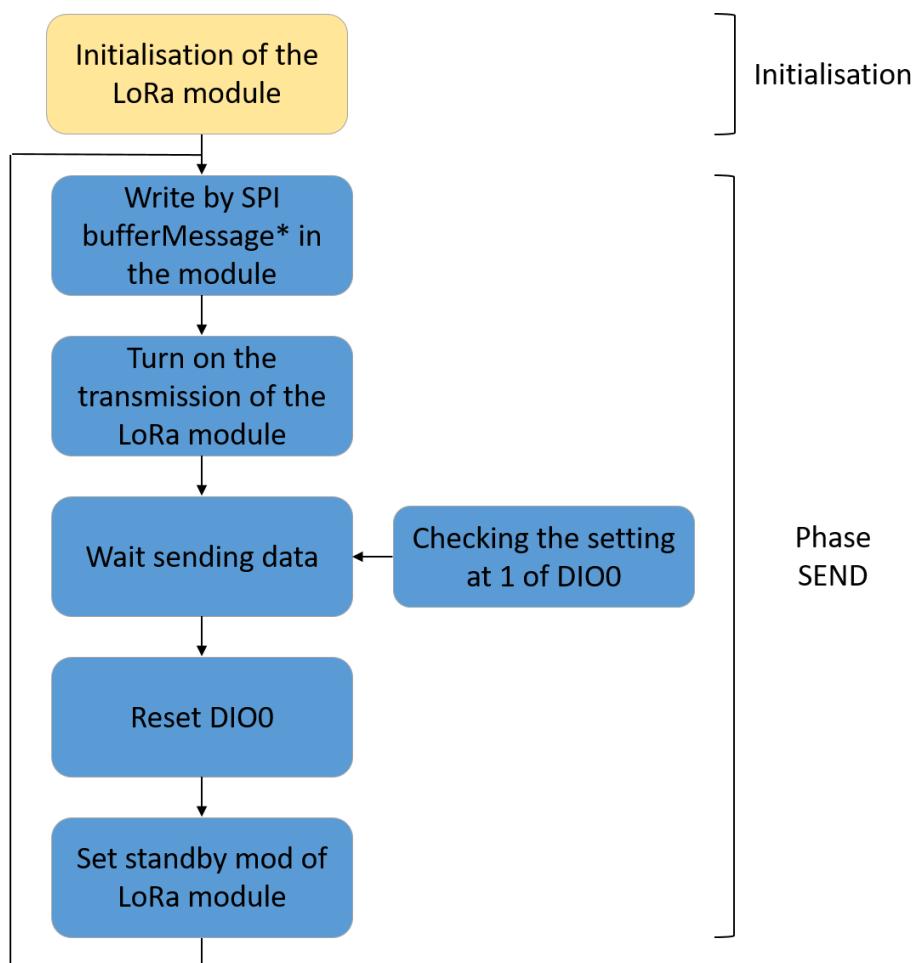
Figure 21 Function ALS frame

7.4. LoRa

There are two LoRa functions to run the Pmod:

- The "initLoRa" function initializes the module
- The "sendTabLora1" function that sends the array passed as a parameter.

These two functions are detailed in the flowchart below (Figure 22).



*bufferMessage : It's the tab where we save data for send by radio.

Figure 22 Flow chart of the LoRa function

The message to send in our application is bufferMessage which contains all processed and ordered sensor information.

7.5. Smart Sensor

The general flowchart of the code (Figure 23) consists of an initialization phase and 4 periodic phases.

The initialization phase in yellow makes it possible to initialize all the Pmods but also to define the GPIOs, the pins of the communications interfaces, the timers etc ...

The 4 periodic phases are described in the flowchart.

Below is an example of a frame (Figure 24) that the Smart Sensor sends.

The explanations of previous frames allow to decode this frame.

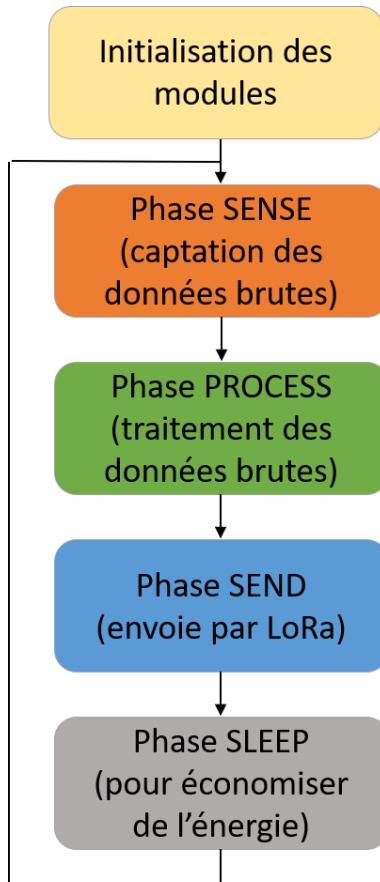


Figure 23 Flow chart of the Smart Sensor

#	H026,10	,	L014,50	,	T023,44	,	H163215	,	LAT4338.1278N	,	LON00350.2516E	,	D210119	*
---	---------	---	---------	---	---------	---	---------	---	---------------	---	----------------	---	---------	---

Figure 24 Frame send by the Smart Sensor

8. Result

8.1. Flex Node

Here is the final result of the project (Figure 25): we see the CMOD A7, the sensors and the radio module.

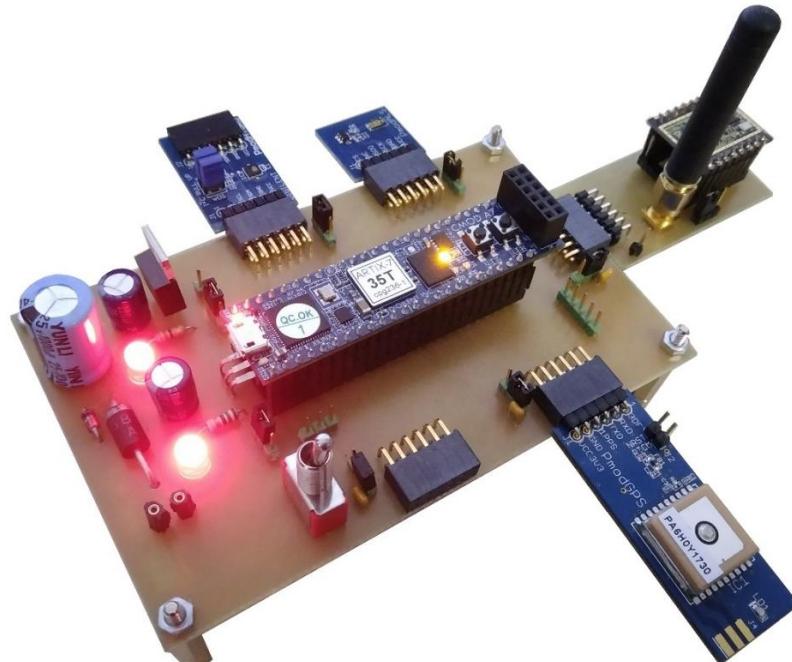


Figure 25 Picture of the Smart Sensor

8.2. Terminal

Here is the data displayed by the terminal, it validates the proper operation of the sensors and each step. Just after "PHASE SEND" we can see the frame sent then the sensor data displayed more legibly (Figure 26).

```

Code loader V0.1
Serial settings
Serial port: COM4: USB Serial Port (COM4) Baudrate: 115200 Serial Close
Bootloader
HEX file: C:/Users/gil/Documents/PIFE/2018-2019/Software/SMART_SENSOR/Debug/SMART_SENSOR.hex ... Start
----- Paul Program Smart Sensor -----
SMART SENSOR N°0
PHASE SENSE
PHASE PROCESS
PHASE SEND
#H026.33,L003.13,T023.07,H082054,,,D290119*
Humidité : 026.33 %
Luminosité : 003.13 %
Température : 023.07 °C
Heure : 08:20:54
Latitude : Pas de réseau GPS
Longitude : Pas de réseau GPS
Heure : 29/01/19

```

Figure 26 Terminal with print data sensors

8.3. Test of consumption

Here is the general consumption of the Smart Sensor (Figure 27). We observe that in these diagrams there are 4 periods of 5 seconds. During the beginning of the period (Phase SENSE, Phase PROCESS and Phase SEND), there is an increase of the power up to 650 mW whereas during the SLEEP Phase the average power is 570mW. The last two curves show that the consumption of the sensors is low ($\approx 130\text{mW}$) compared to the microcontroller implanted in the CMOD A7 board ($\approx 470\text{mW}$).

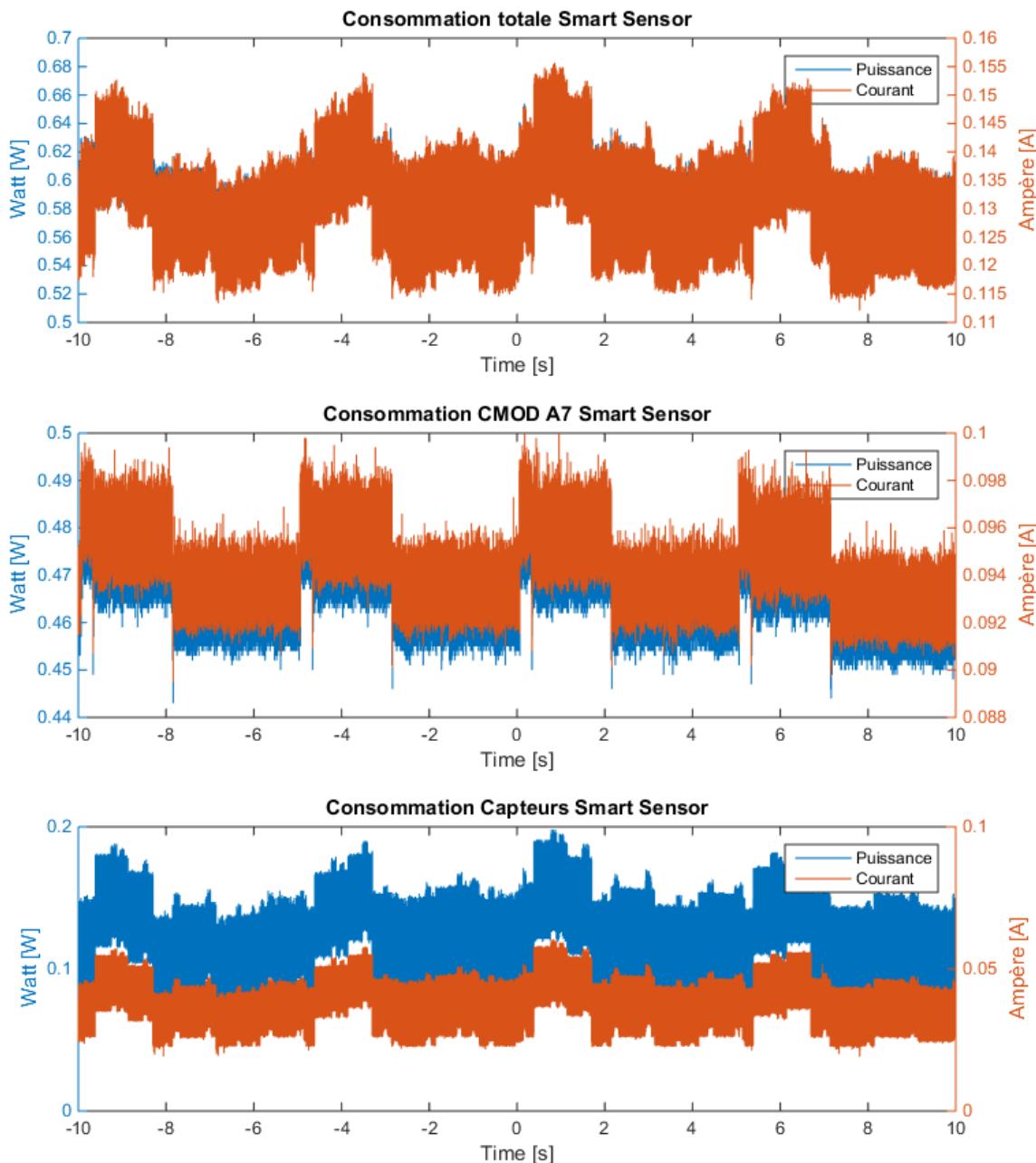


Figure 27 Total consumption, microcontroleur consumption and sensors (« capteur » in french) consumption

To visualize which sensor consumes the most and when, a measurement of each sensor was made. Thanks to the following 4 curves (Figure 28) we clearly see the SENSE Phase which captures the sensor data on the Pmod HYGRO and Pmod ALS. However the Pmod GPS sends data regularly, every 2 seconds, because it emits data without any requests.

The LoRa of the SEND Phase emits about 1.5 seconds at a power of 28mW.

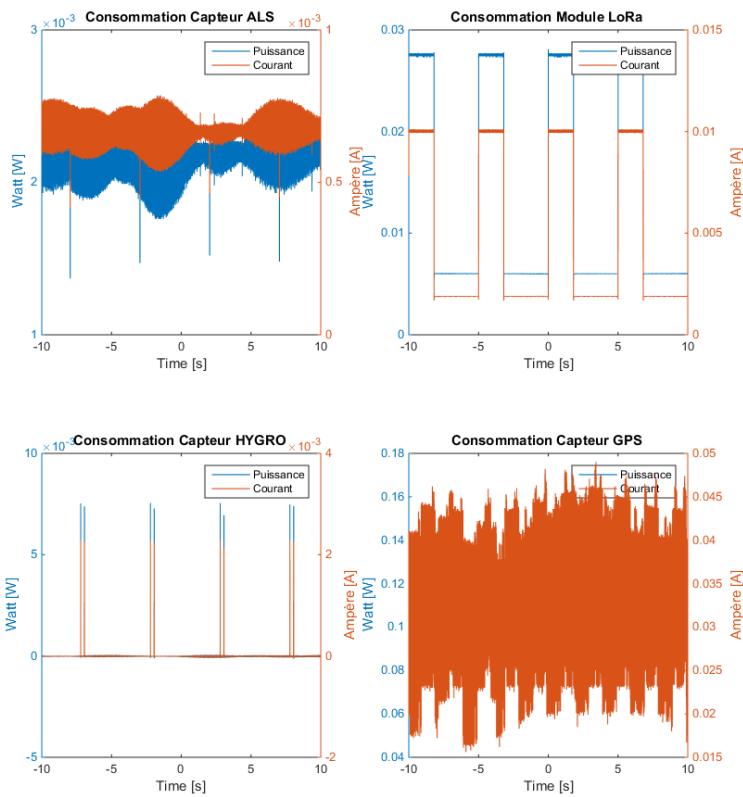


Figure 28 Consumption of each sensors/radio modules

With this chart (Figure 29) with the same scale on each curve, we see that the most important consumption is the CMOD A7. The 4 modules consume very little. In the modules it is the GPS that consumes the most (measurements made without the GPS attached to satellites). Then there is the consumption of LoRa during the sending phase which consume a lot also. The other sensors, HYGRO and ALS, consume very little.

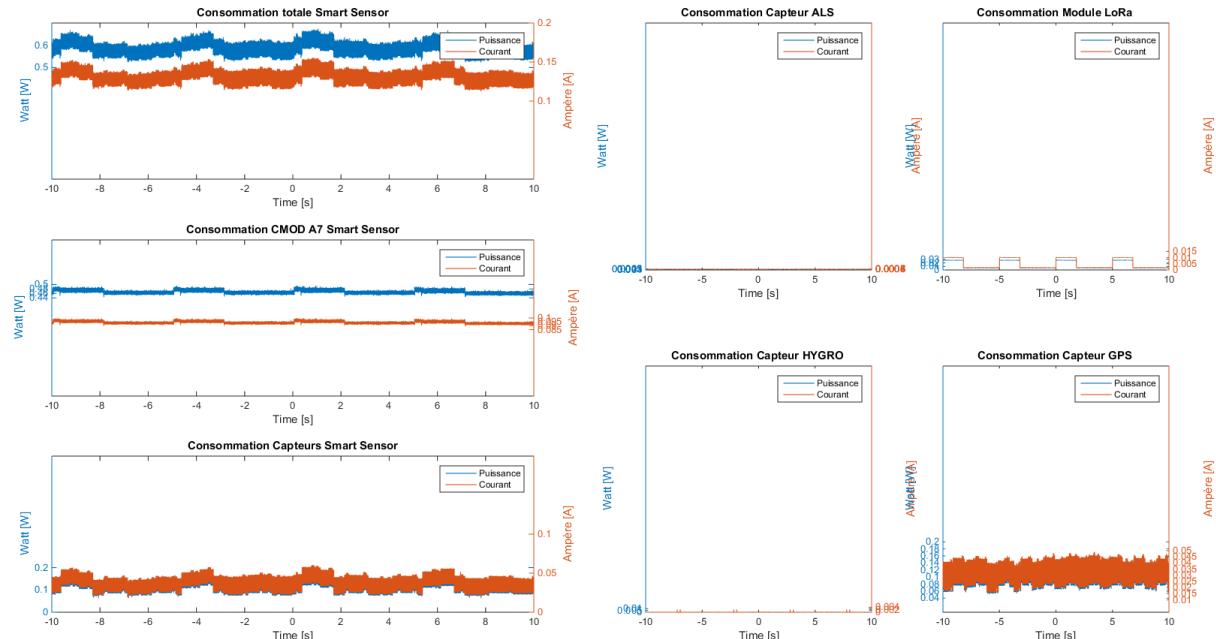


Figure 29 Global view of the consumption

9. Conclusion

The Smart Sensor is now operational, it meets the specifications and was done on time. Consumption tests show a consistent result. The cost of the Smart Sensor is around 150 € by counting the card and the components.

This industrial project of end of study brought me a lot, either personally or professionally. I was able to discover the laboratory research community. I was able to develop myself personally by working on a project composed of hardware and software.

This experience supported and anchored my knowledge learned during my training. In addition to validating my skills, I was able to develop others such as knowing how to set up an IDE or use communication interfaces other than the UART.

10. Reference

- [1] <https://fr.statista.com/statistiques/584481/internet-des-objets-nombre-d-appareils-connectes-dans-le-monde--2020/>
- [2] https://www.iea.org/publications/freepublications/publication/MoreData_LessEnergy.pdf
- [3] <https://www.objetconnecte.net/objets-connectes-energie-101116/>
- [4] <https://store.digilentinc.com/pmod-modules-connectors/>
- [5] <https://fr.farnell.com/>
- [6] <http://www.lirmm.fr/>

11. Appendix

State of the art

Etude & Conception d'un Smart Sensor sur FPGA

Projet Industriel de fin d'étude

AUTEUR : PAUL LELOUUP

TUTEUR : PASCAL BENOIT

ENCADRANTS : PASCAL BENOIT & GUILLAUME PATRIGEON

2018



MEA

Microélectronique
et automatique



POLYTECH[®]
MONTPELLIER



LIRMM

Table des matières

Table des matières	1
1. Introduction	1
2. Gantt	2
3. Capteurs	3
3.1. Définition, types de capteurs.....	3
3.2. Analogique / Numérique	3
4. Communication	5
4.1. Réseaux sans fil pour l'IOT.....	5
4.2. Energie.....	5
5. Conclusion.....	6
Références	7

1. Introduction

Le lieu d'étude est le Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM) qui est une unité mixte de recherche dépendant conjointement de l'Université Montpellier (UM) et du Centre National de la Recherche Scientifique (CNRS) [1].

Le projet a lieu dans le cadre d'un projet industriel de fin d'étude qui a pour objectif de mettre l'élève dans la situation d'un chef de projet chargé de trouver des solutions à un problème concret qui lui est posé [2].

L'internet des objets (IOT) est l'extension d'Internet à des choses et à des lieux du monde physique. L'internet des objets connectés représente les échanges d'informations et de données provenant de dispositifs du monde réel avec le réseau Internet.

L'architecture de l'appareil est présentée sur le schéma ci-dessous (Figure 1). Le projet a pour but de créer la carte électronique qui reliera les capteurs, la carte FPGA et les modules de communication ainsi que de les programmer pour les faire fonctionner ensemble. Cela a pour objectif d'aider une thèse sur l'exploration d'architecture de microcontrôleur afin de baisser la consommation d'énergie.

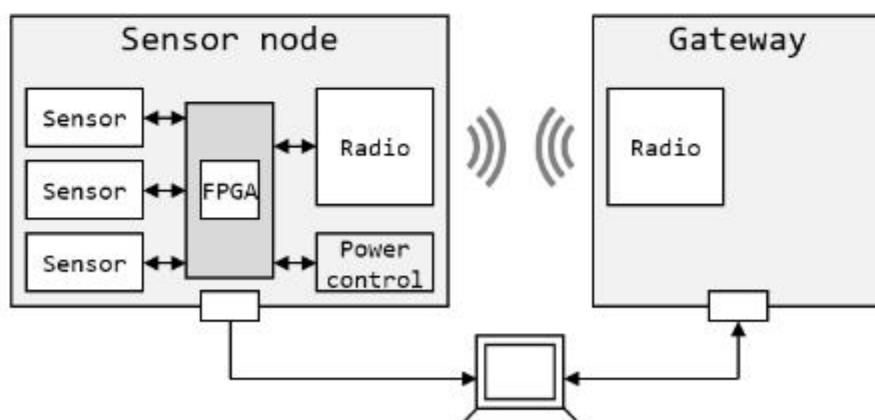
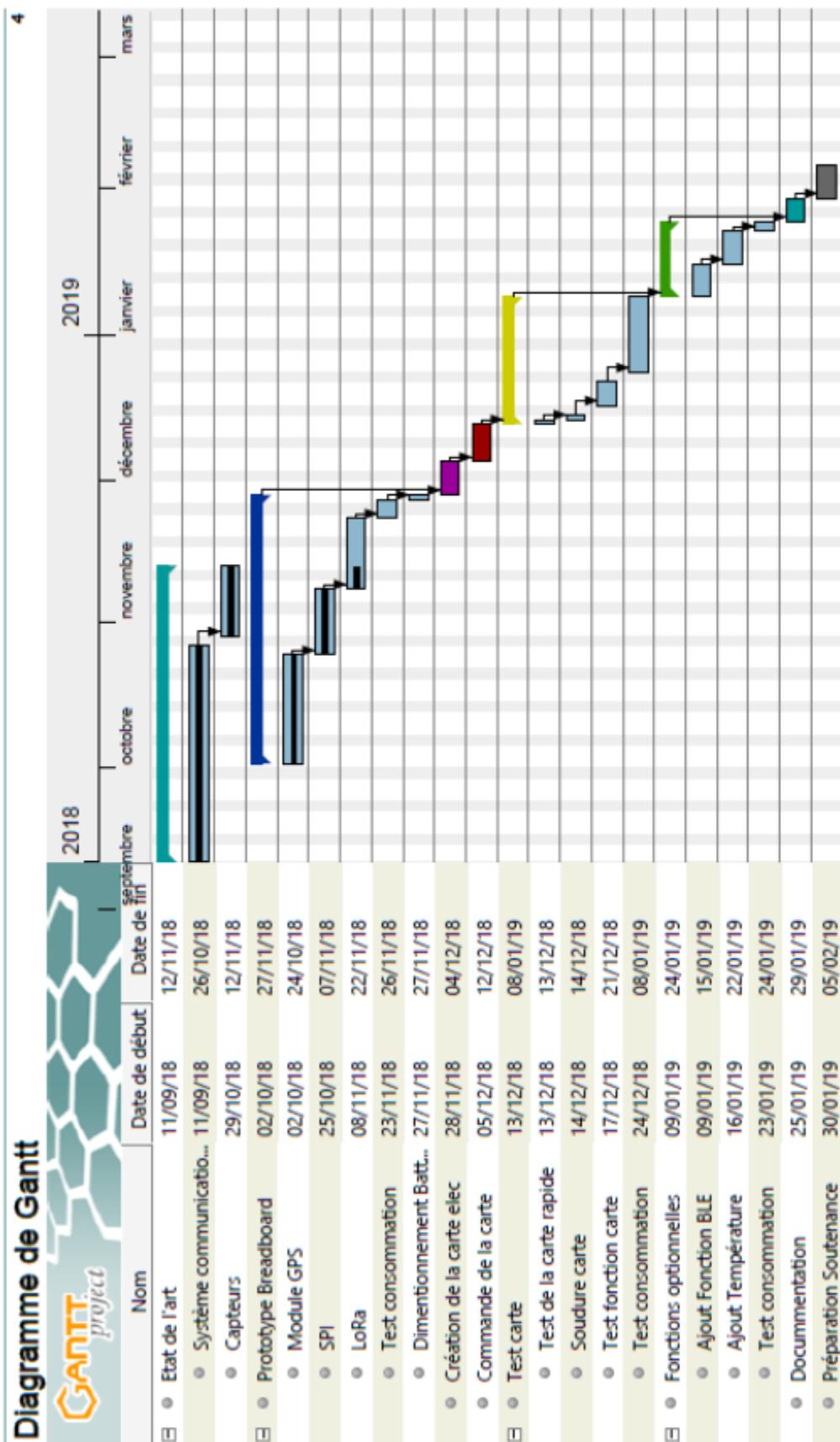


Figure 1 Architecture du Smart Sensor

Gantt

2. Gantt



3. Capteurs

3.1. Définition, types de capteurs

Un capteur est un dispositif qui fournit à partir d'une grandeur physique prélevée, une autre grandeur physique de nature différente (souvent électrique) [3].

Il y a deux types de capteurs différents. Les capteurs actifs sous l'action du mesurande, génère un signal électrique (courant ou tension), on peut l'appeler générateur. Les capteurs passifs sous l'action du mesurande font apparaître une variation d'impédance. Le capteur passif ne produit pas directement un signal électrique, il sera indispensable d'utiliser un circuit électrique extérieur pour déterminer la valeur du mesurande.

Quelques exemples de capteurs passifs : température, flux optique, déformation, humidité...

Quelques exemples de capteurs actifs : flux optique, force, pression, vitesse...

Le capteur de température utilisé est le PmodTMP2 de Digilent [4] (Figure 3) qui fonctionnera dans les conditions suivantes : $V_{dd} = 3.3V$, lecture périodique. Les avantages sont les suivants :

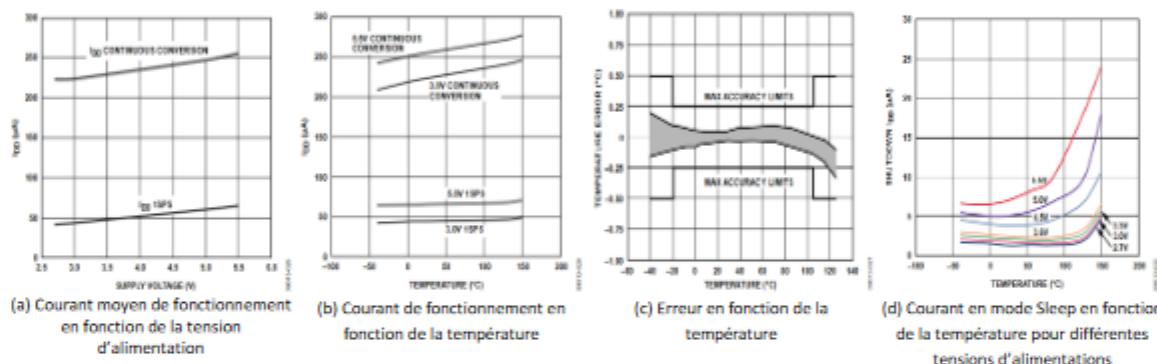


Figure 2 Caractéristique du capteur de température PmodTMP2 de Digilent

Le courant moyen consommé est de $225\mu A$ en fonctionnement continu (Figure 2 (a)), il est cependant légèrement variable en fonction de la température (Figure 2 (b)). Le capteur est très précis car il a une erreur maximale de $0.5^{\circ}C$ entre $-20^{\circ}C$ et $105^{\circ}C$ (Figure 2 (c)). Le mode Sleep consomme seulement $5\mu A$ maximum (Figure 2 (d)) et permet d'économiser de l'énergie ce qui est très important dans notre application [5].

3.2. Analogique / Numérique

Il faut transformer ces données analogiques en données numériques. On va créer des circuits électriques pour recevoir des valeurs numériques ou utiliser, comme dans la plupart des cas, un microcontrôleur. Il existe beaucoup de capteurs sous forme de module qui délivre la donnée numérique. Capteurs utilisés :

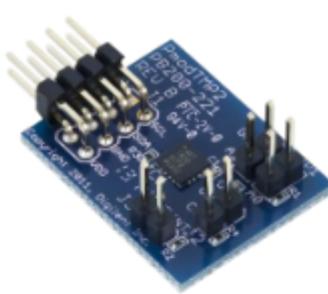


Figure 3 Capteur de température PmodTMP2 de Digilent



Figure 4 Capteur GPS PmodGPS de Digilent

Pour faire l'interface entre le module et le microcontrôleur central, il faut communiquer par des câbles. On va utiliser des interfaces de communications tels que l'UART, l'I²C, le SPI et d'autres encore.

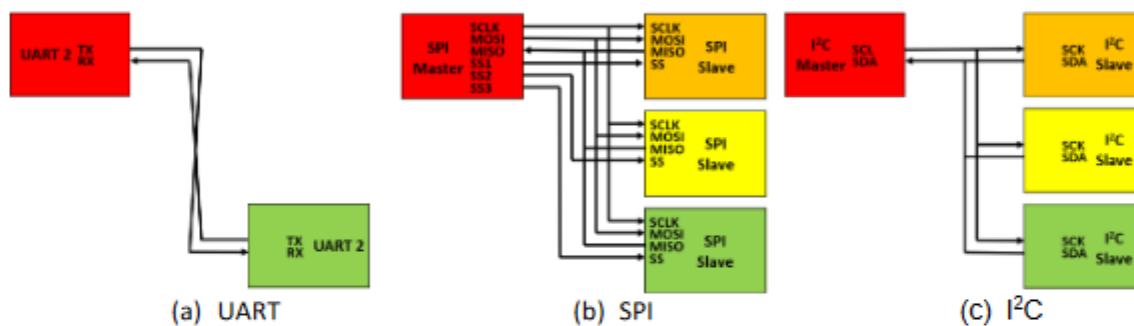


Figure 5 Interfaces de communications

Un UART (Figure 5 (a)), pour Universal Asynchronous Receiver Transmitter, est un émetteur-récepteur asynchrone universel.

Une liaison SPI (Figure 5 (b)) (pour Serial Peripheral Interface) est un bus de données série synchrone. Les circuits communiquent selon un schéma maître-esclaves, où le maître contrôle la communication. Plusieurs esclaves peuvent coexister sur un même bus, dans ce cas, la sélection du destinataire se fait par une ligne dédiée entre le maître et l'esclave appelée Slave Select.

I²C (Figure 5 (c)) est un bus série synchrone bidirectionnel half-duplex, où plusieurs équipements, maîtres ou esclaves, peuvent être connectés au bus.

Les UART, SPI et I²C sont généralement intégrés dans des composants comme des microcontrôleurs. Ils ne sont dans ce cas plus un composant à proprement parler, mais une fonction périphérique du composant central.

4. Communication

4.1. Réseaux sans fil pour l'IOT

Pour l'IOT il est important que le plus d'appareils soient reliés pour avoir plus d'informations et agir de la meilleure manière possible. L'intérêt du réseau sans fil est qu'il est pratique (pas de fil) et que l'installation d'un nouvel appareil sera rapide et simple. Voici quelques réseaux sans fil adapté à l'IOT qui vont être comparés au niveau de la consommation énergétique :

SIGFOX est une technologie UNB (Ultra Narrow Band) qui opère à 868 MHz avec un débit de 1000 b/s pour une bande passante de 1 KHz et une sensibilité de -140 dBm avec une portée annoncée de 40 km. Les appareils utilisant cette technologie peuvent envoyer 140 messages par jour à une station. C'est une technologie asynchrone, donc les nœuds ne se réveillent pas pour se synchroniser. Il est donc plus simple de calculer la consommation [6].

LoRa est faite pour les applications IOT à longue portée. LoRa Alliance propose une station qui reçoit des paquets depuis un appareil et retransmet les données à un serveur par connexion TCP. Le LoRa utilise principalement de la modulation basée sur le CSS (Chirp Spread Spectrum) qui permet un débit de 0,25 à 11 kb/s avec une bande passante de 7 à 250 KHz. Dans cette comparaison seule LoRa classe A sera pris en compte [6].

Bluetooth Low Energy est une version du Bluetooth standard. Il est le plus souvent utilisé pour des applications tel que des moniteurs de fréquence cardiaque ou des contrôles à distance de température d'une pièce. L'avantage de cette technologie est qu'elle est très populaire (dans la majorité des smartphones) et que les dernières versions (4.2, 5.0) permettent d'avoir plusieurs nœuds. BLE a une fréquence d'utilisation de 2,4 GHz, le débit maximal est de 2 Mb/s. Sur la dernière version 5.0, on peut utiliser 4 schémas différents qui permettent d'avoir plusieurs débit (125 kb/s, 500 kb/s, 1 Mb/s et 2 Mb/s) pour moins consommer en fonction des applications [6].

La technologie **NB-IoT** (Narrowband Internet of Things) est une technologie de communication sans fil qui fonctionne à 180KHz avec un débit de 250 kb/s. Appliquée à ce jour principalement au secteur de l'industrie, le NB-IoT, porté par la Chine, souhaite conquérir maintenant celui de la Smart Home, du Smart Building et de la Smart City. Les atouts de cette technologie sont nombreux : faible consommation d'énergie, couverture étendue et faible coût de production et d'utilisation [7].

4.2. Energie

L'IOT est intéressant par le fait de son autonomie et de sa source d'énergie sans fil. Les résultats présentés se basent sur un article scientifique : *Comparison of the Device Lifetime in Wireless Networks for the Internet of Things* [6].

Les calculs de la durée de vie des objets connectés sont faits à partir des valeurs des variables de ce schéma (Figure 6).

t_a signifie la période d'application

t_{on} signifie la période d'émission

t_{cl} signifie la période de la synchronisation

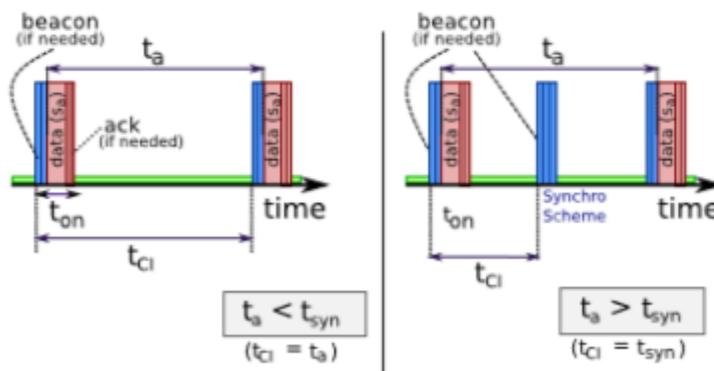


Figure 6 Calcul de consommation

Les auteurs de l'article ont créé l'algorithme qui va traiter ces valeurs et nous donner la consommation.

Voici la consommation de chaque mode (Figure 7) (il ne faut pas prendre en compte uniquement ce tableau car chaque protocole utilisera pendant des temps plus ou moins long ces modes) :

Power	P_Tx	P_Rx	P_Idle	P_Sleep
SIGFOX	147mW	39mW	Ø	4.32uW
LoRa	419.6mW	44mW	Ø	4.32uW
BLE	24.11mW	9.26mW	4.67mW	3.24uW
NB-IOT	Ø	Ø	Ø	Ø

Figure 7 Récapitulatif des consommations

NB-IOT n'a pas de valeur dans le tableau car ce système de communication n'était pas dans l'article étudié et prendre des valeurs d'autres études ne serait pas objectif étant donné que les conditions de test ne seraient pas les mêmes.

Il y a un autre algorithme qui permet de calculer la consommation en fonction de la probabilité de paquet perdu que l'on doit renvoyer.

5. Conclusion

L'enjeu majeur de l'IOT est l'autonomie, d'où la préférence d'utiliser des capteurs actifs (qui génèrent une tension ou un courant). L'intérêt est de faire de l'exploration d'architecture sur FPGA de microcontrôleur pour baisser la consommation.

Voici les résultats de mes recherches sur la consommation des protocoles radio (rappel : sauf NB-IOT) :

- Lorsque les paquets ne sont pas perdus :

Pour une émission tous les jours, toute les 100s, toute les secondes et millisecondes le BLE est le plus efficace en termes d'économie d'énergie.

- Lorsque 20 % des paquets sont perdus :

Le BLE reste plus économique en énergie mais la durée de vie diminue d'environ 10 %.

- La dérive d'horloge :

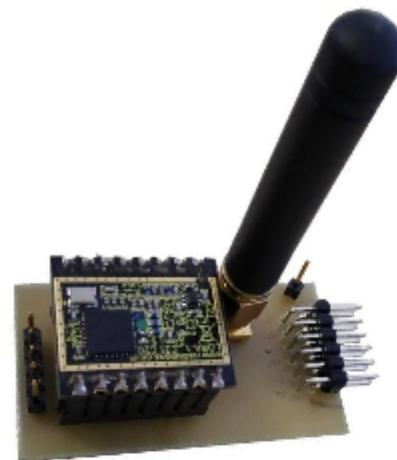
Elle ne fait que très peu baisser la durée de vie du LoRa et du SIGFOX à une faible intensité de trafic, contrairement au BLE.

Pour notre application on va donc préférer le LoRa car il consomme finalement assez peu mais a une portée bien plus grande que le BLE. On va préférer utiliser des capteurs Pmod pour faciliter les tests de différents capteurs lorsque la carte sera terminée. Des capteurs tels que la température, la localisation vont être utilisés afin d'avoir des données pertinentes à transmettre.

L'utilisation du LoRa est intéressante pour le LIRMM car elle permettra d'obtenir des données sur cette technologie récente.

LoRa Pmod V2.0

Pmod designed by Paul LELOUP



Additional features

- Integrated RF-LoRa-868 of RF Solutions
- A Pmod Connector for plug and play (using FPGA card for prototype)
- An SMA connector for the antenna
- Small card dimensions: 27.9mm x 50.9 mm
- 4 connectors for measure current and tension of the modules
- 5 connectors for use optional PINS of the RF-LoRa-868

RF-LoRa-868 Features

- Up to 16KM Range
- Integrated LoRa Modem Semtech SX1272
- Highly Efficient Integral Impedance Matching Network
- Provides Full Functionality of the RFIC:
- 157 dB maximum link budget
- +20 dBm at 100mW constant RF output vs. V supply
- -14 dBm high efficiency PA
- Built in RF switch
- High sensitivity: down to -130 dBm
- Bullet-proof front end: IIP3 = -12.5 dBm
- 89 dB blocking immunity
- Programmable bit rate up to 300kbps
- Low RX current of 10 mA, 100nA register retention
- FSK, GFSK, MSK, GMSK, LoRaTM and OOK modulation
- Built-in bit synchronizer for clock recovery
- Preamble detection
- 127 dB Dynamic Range RSSI
- Automatic RF Sense and CAD with ultra-fast AFC
- Packet engine up to 256 bytes with CRC
- Built-in temperature sensor and low battery indicator

Application

- Prototype with FPGA
- Smart Sensor
- Home Automation
- RF Alarms
- Sensor networks
- Long Range Telemetry
- Meter Reading
- Irrigation Systems
- Wireless Applications

Part Numbers

Pmod connector

PIN	Definition	Direction	Function
1	RESET	In	Reset Trigger Input
2	DIO0	In/Out	Digital I/O software configured
3	TX_SWITCH	In	Enable TX RF Path Active High
4	RX_SWITCH	In	Enable RX RF Path Active High
5	GND	-	Ground connection
6	VCC	In	Power connection
7	SCLK	In	SPI Serial Clock Input
8	NSEL	In	Device Select Active Low
9	SDI	In	SPI Serial Data Input
10	SDO	Out	SPI Serial Data Output
11	GND	-	Ground connection
12	VCC	In	Power connection

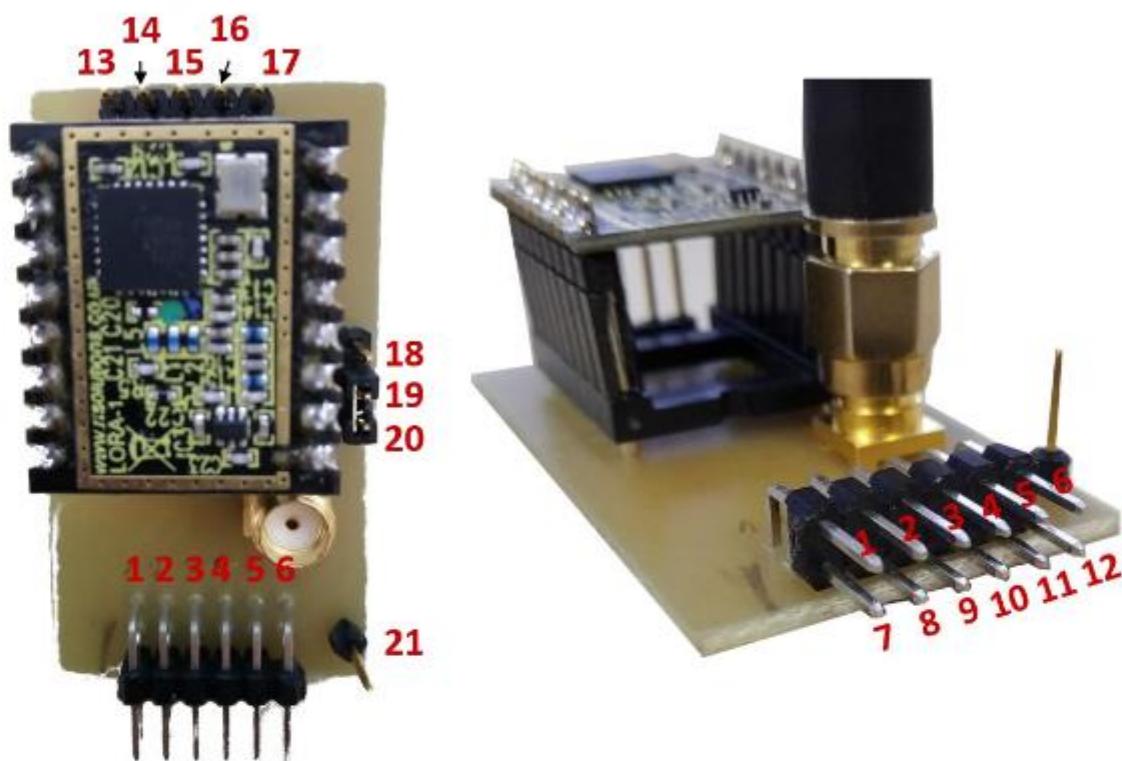
5 Pins connector

PIN	Definition	Direction	Function
13	DIO3	In/Out	Digital I/O software configured
14	DIO4	In/Out	Digital I/O software configured
15	DIO5	In/Out	Digital I/O software configured
16	DIO1	In/Out	Digital I/O software configured
17	DIO2	In/Out	Digital I/O software configured

4 Pins connector

PIN	Definition	Direction	Function
18	VCC Module	-	Pin for tension measure
19	VCC Module	-	Pin for current measure
20	VCC Pmod	-	Pin for current measure
21	GND	-	Pin for tension measure

Pin Description



Using the Pmod

First you can plug the RF-LoRa-868 like this in the Pmod card:



Then for turn ON the alimentation of the Pmod, you must put a bridge between the Pin 19 and 20. Now there is 3.3V to the module.

It's highly recommended to put an SMA antenna (868 MHz).

Go on your keyboards, you can use the Pmod!

Schematic

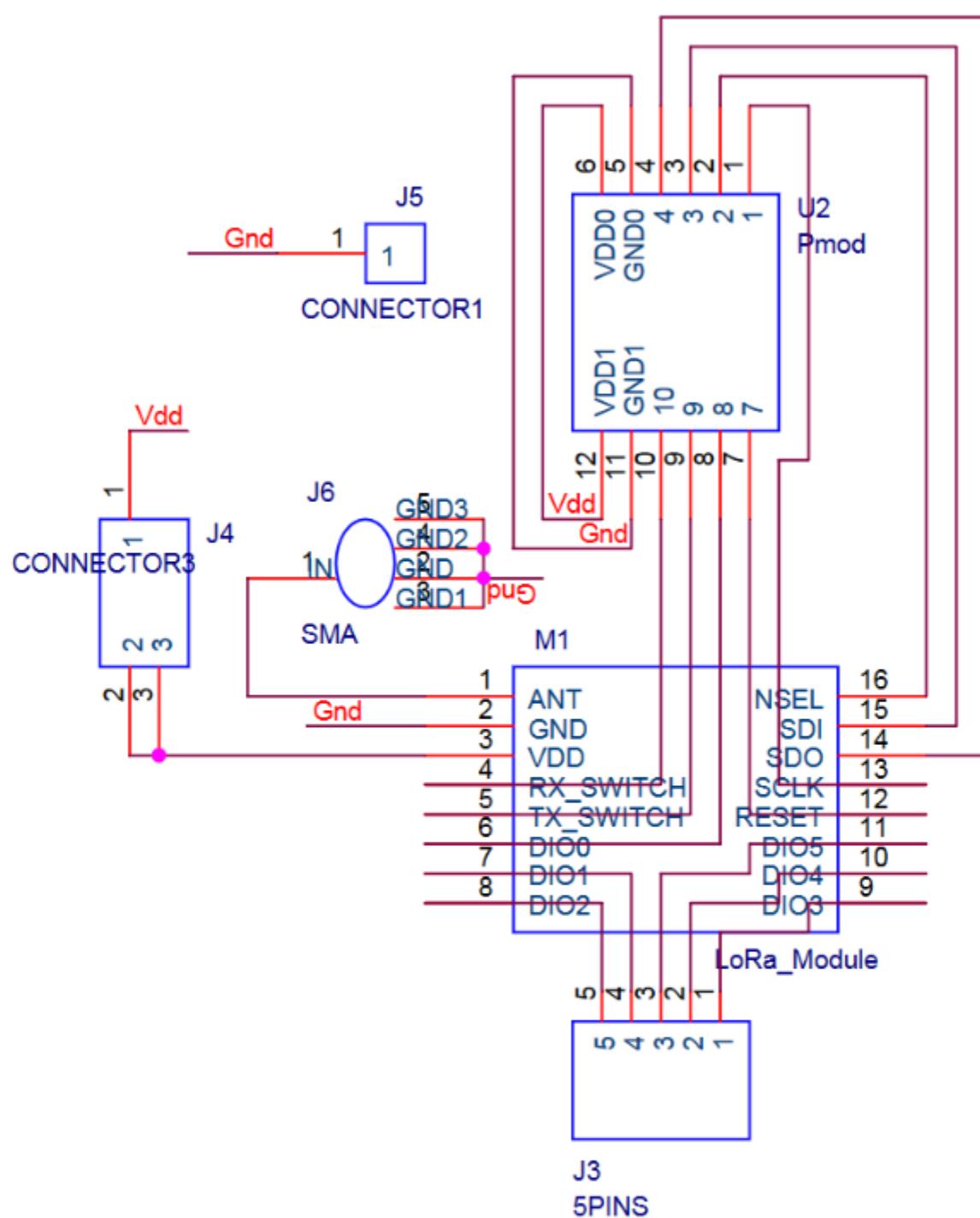


Figure 1 Schematic of Pmod v2.0

Board

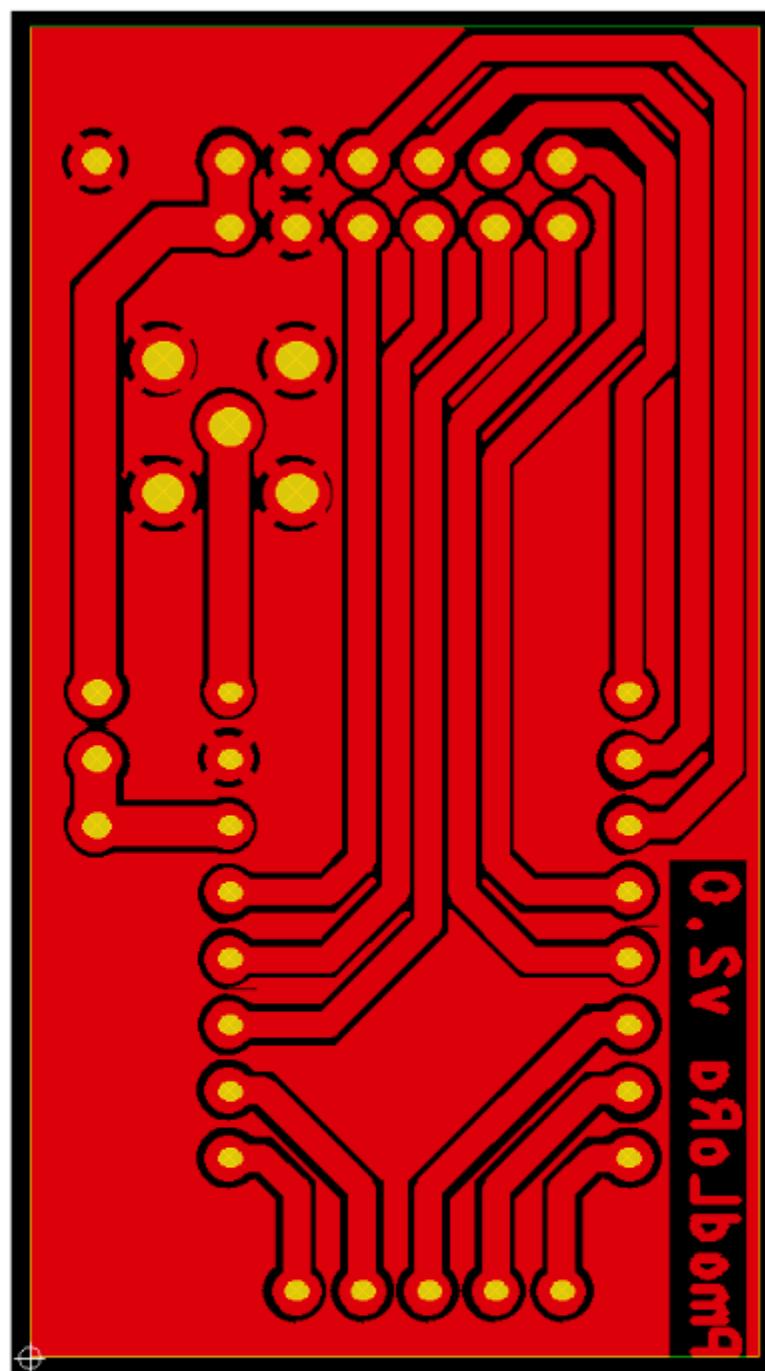


Figure 2 Bottom view Board of Pmod v2.0

References

- Datasheet RF-LORA-868
http://www.farnell.com/datasheets/2162976.pdf?_ga=2.254649036.578254260.1548165701-413366784.1537950013
- Datasheet SX1272/73
<https://www.semtech.com/uploads/documents/sx1272.pdf>

Tutoriel decouverte

Tutoriel de prise en main du CmodA7 FPGA

Etude & Conception d'un Smart Sensor sur FPGA

AUTEUR : PAUL LELOUP

TUTEUR : PASCAL BENOIT

THESE DE : GUILLAUME PATRIGEON

2019



Table des matières

Table des matières	1
1. Prérequis	2
1.1. Matériel.....	2
1.2. Logiciel.....	2
2. Tutoriel.....	3
2.1. Mettre en place l'IDE, le Bootload.....	3
Installer JAVA.....	3
Installation IDE Eclipse.....	3
Paramètres du projet (IDE Eclipse).....	7
Installation du Codeloader	13
Utilisation du Codeloader.....	14
Faire clignoter la LED 2	16
2.2. Utiliser l'interface UART avec le module GPS	16
Initialiser l'UART2	16
Les afficher	16
Les stocker.....	16
2.3. Mettre sous un bon format	16
Machine d'état	16
Créer une trame	16
3. Pour approfondir	16
3.1. Utiliser les autres interfaces du CmodA7, SPI et I2C.....	16
3.2. Envoyer la trame par LoRa.....	16

1. Prérequis

1.1. Matériel

Pour le matériel vous aurez besoin de :

- Un ordinateur
- Un FPGA CmodA7 de DIGILENT
- Un module Pmod GPS de DIGILENT
- Un câble USB-microUSB

1.2. Logiciel

Au niveau du logiciel vous aurez besoin de :

- Un IDE, Eclipse CDT :
 - Lien vers le site : <https://www.eclipse.org/cdt/downloads.php>
 - Téléchargement d'Eclipse version 2018.2 :
<https://www.eclipse.org/downloads/download.php?file=oomph/epp/2018-12/R/eclipse-inst-win64.exe>
- Le compilateur gcc-arm-none-eabi :
 - Lien vers le site :
<https://developer.arm.com/open-source/gnu-toolchain/gnu-rm/downloads>
Télécharger la version « 7-2017-q4-major » et cliquer sur « window zip »
L'extraire et le déplacer dans un dossier à ne pas supprimer.
- Python 3 :
 - Lien vers le site : <https://www.python.org/downloads/>
 - Téléchargement de Python 3.7.2 :
<https://www.python.org/ftp/python/3.7.2/python-3.7.2.exe>
- Java JRE :
 - Lien vers le téléchargement
<https://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html>
Accepter les droits
Aller dans « Java SE Runtime Environment 8u201 » et cliquer sur la version « x64 » en « .exe »
- La librairie du cortex M0 est présente dans l'archive du tutoriel
- Les fichiers sources sont présents dans l'archive du tutoriel
- Les fichiers *linker.Id* et *startup.asm* sont présent dans l'archive du tutoriel
- L'outil Codeloader V0.1 en python est présent dans l'archive du tutoriel
- Le datasheet du CmodA7 pour avoir les PINS est présent dans l'archive du tutoriel

2. Tutoriel

2.1. Mettre en place l'IDE, le Bootload

Installer JAVA

Double cliquer sur le .exe

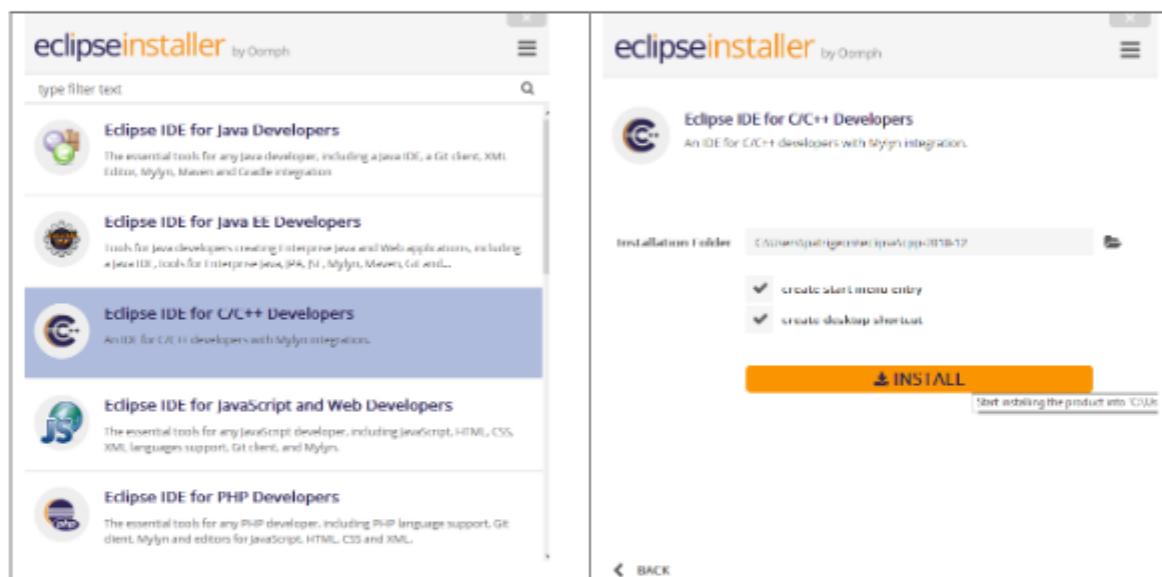
Cliquer sur installer

Cliquer sur fermer

Java est maintenant installé

Installation IDE Eclipse

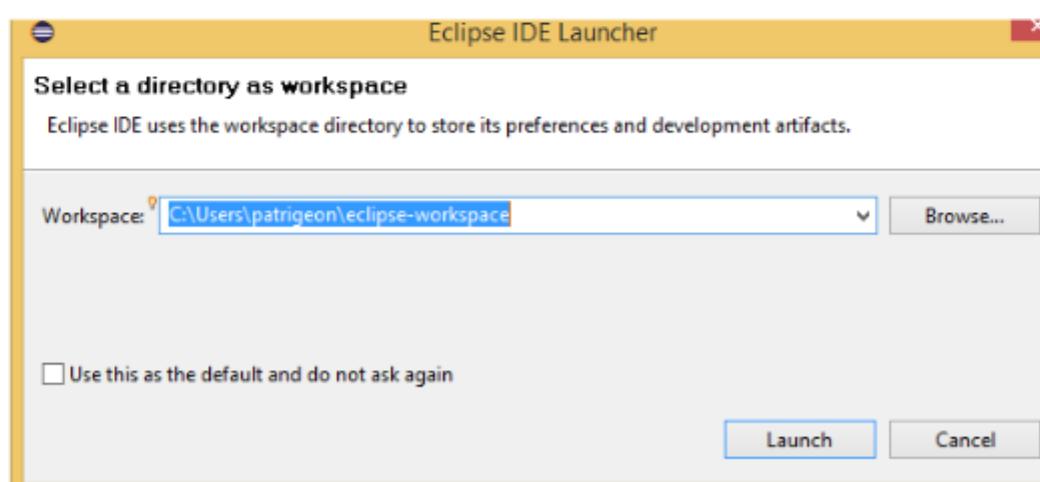
Double cliquer sur le .exe



Accepter toutes les conditions

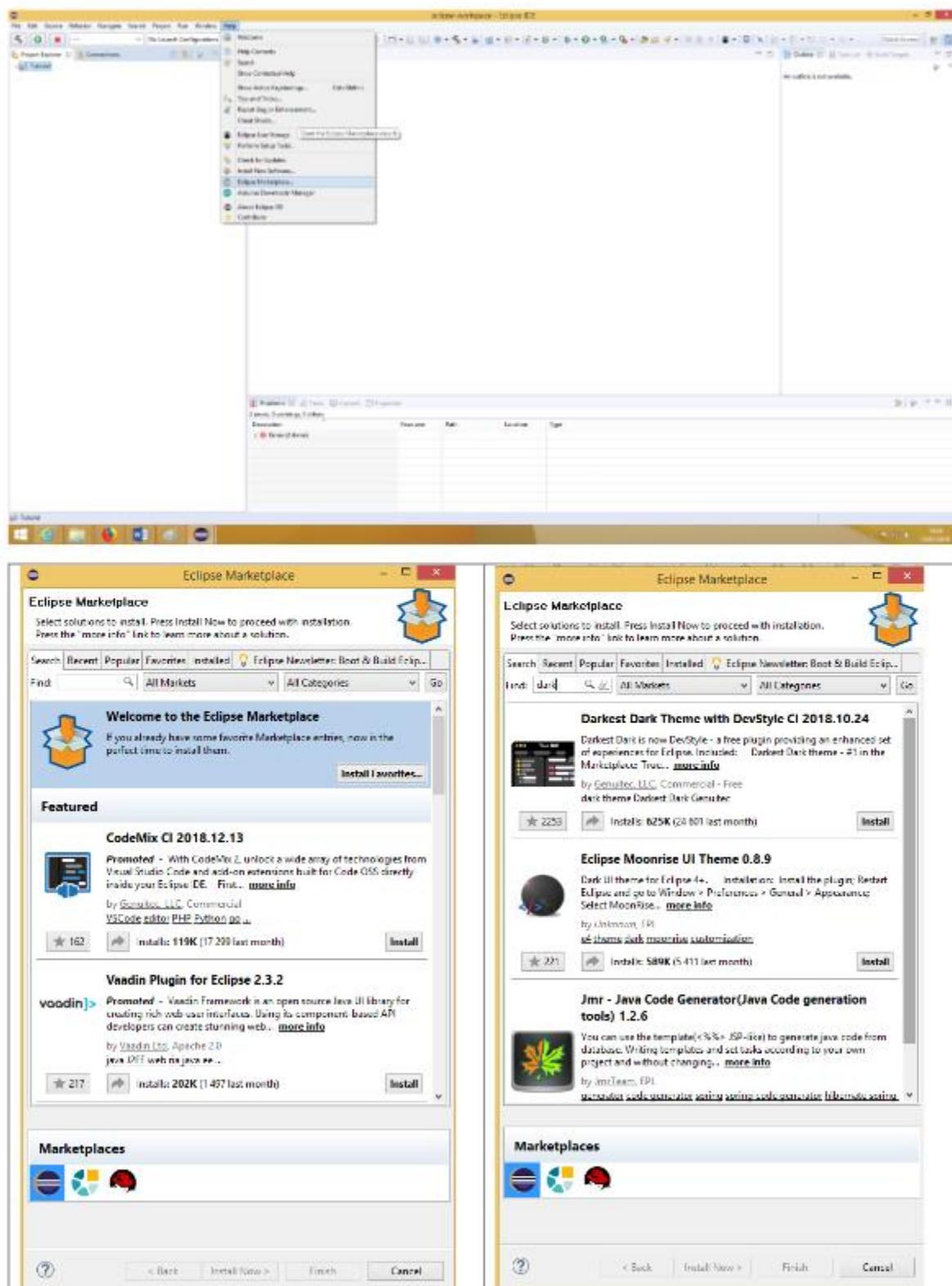
Définir le dossier où tous les projets seront enregistrés :

(Bien conserver le chemin qui mène aux projets)



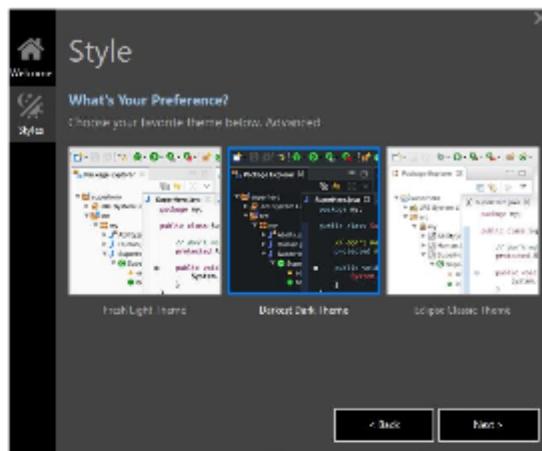
Pour le Dark theme ;

Help/marketplace



Confirmer les conditions

Restart

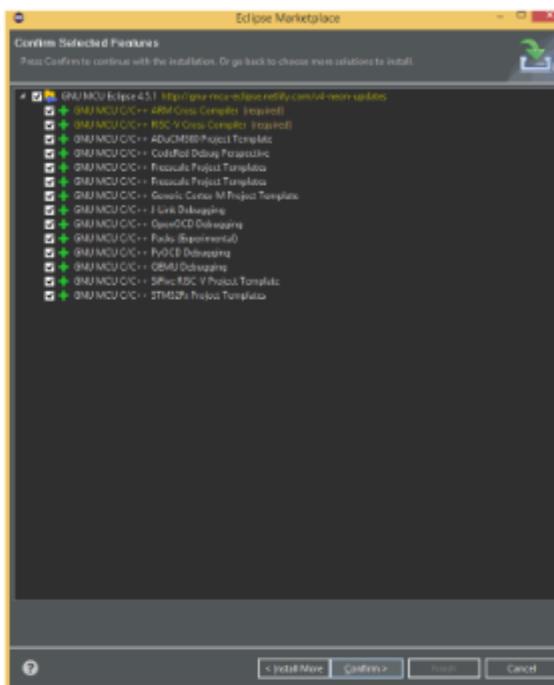


Installer GNU ARM ;

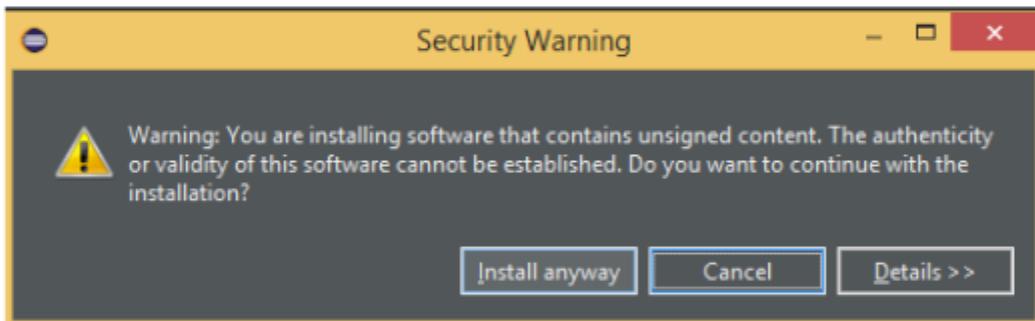
Help/marketplace



Confirmer les conditions



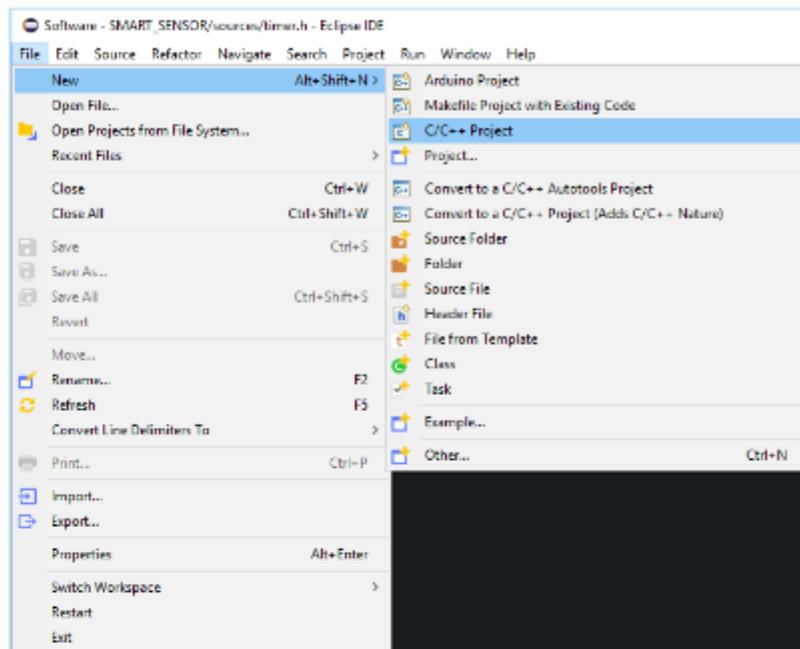
Install anyway



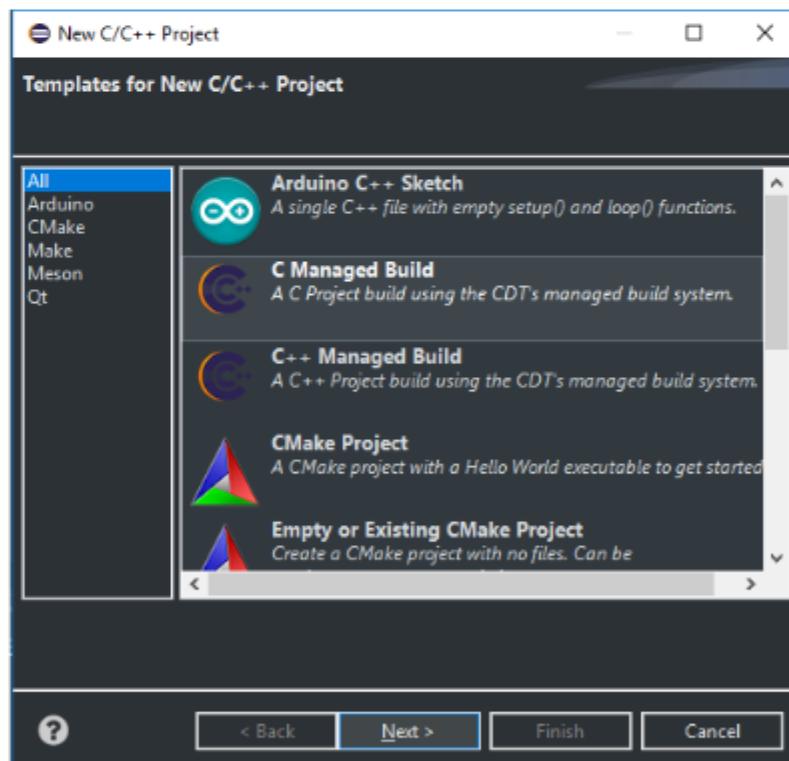
Restart

Paramètres du projet (IDE Eclipse)

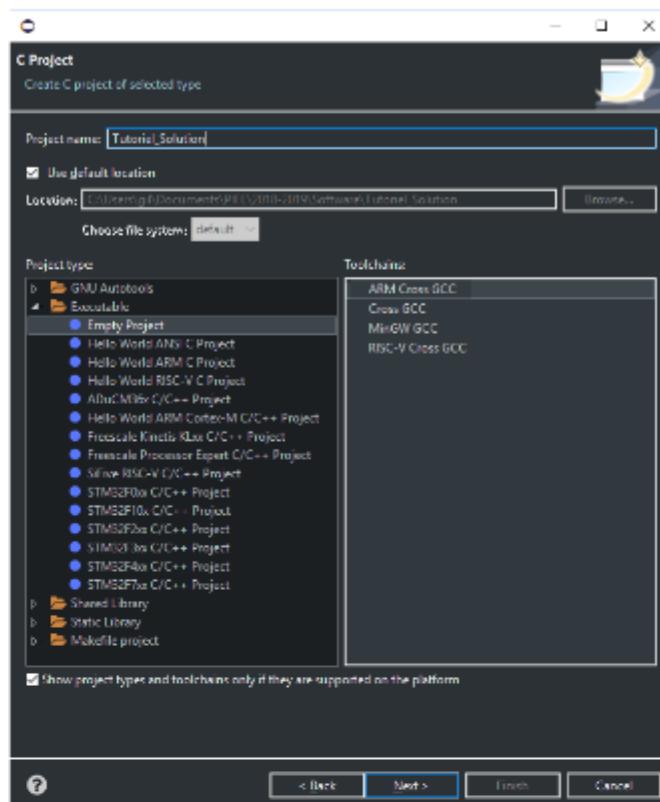
Faire un nouveau projet C/C++



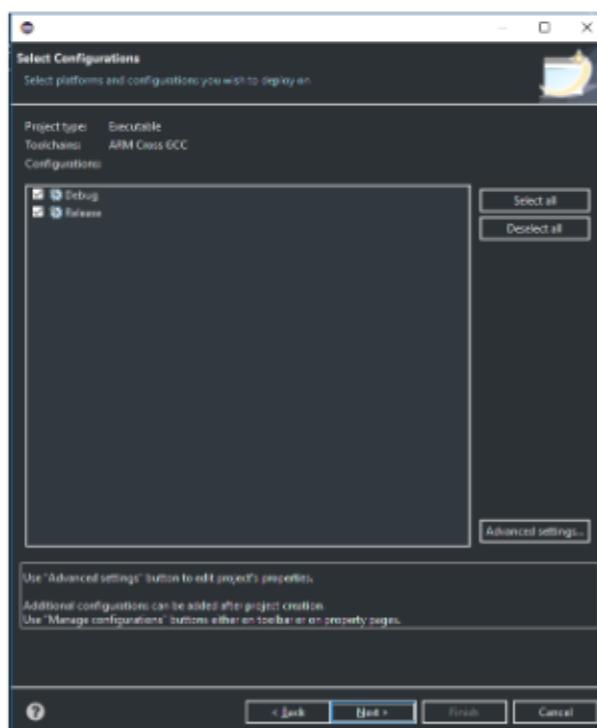
Sélectionner « C Managed Build »



Choisir le nom du projet, sélectionner ARM Cross GCC et cliquer sur « Next »



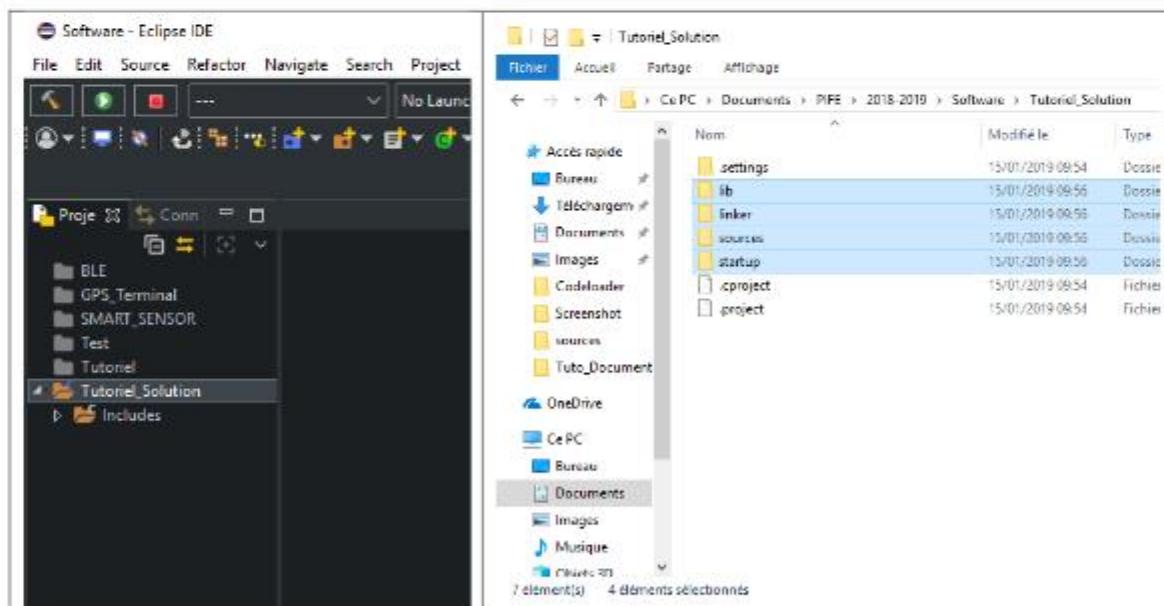
Puis cliquer encore sur « Next »



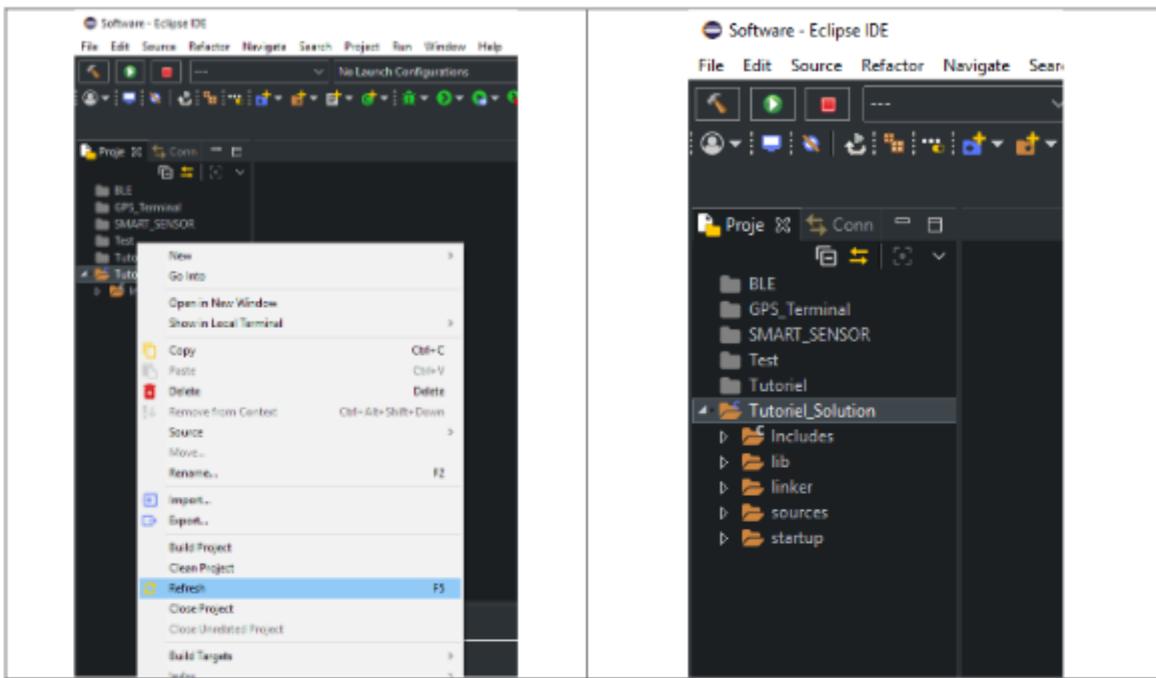
Pour le GNU ARM Cross Toolchain, sélectionner le path de gcc-arm-none-eabi-7-2017-q4-major-win32 préalablement téléchargé.



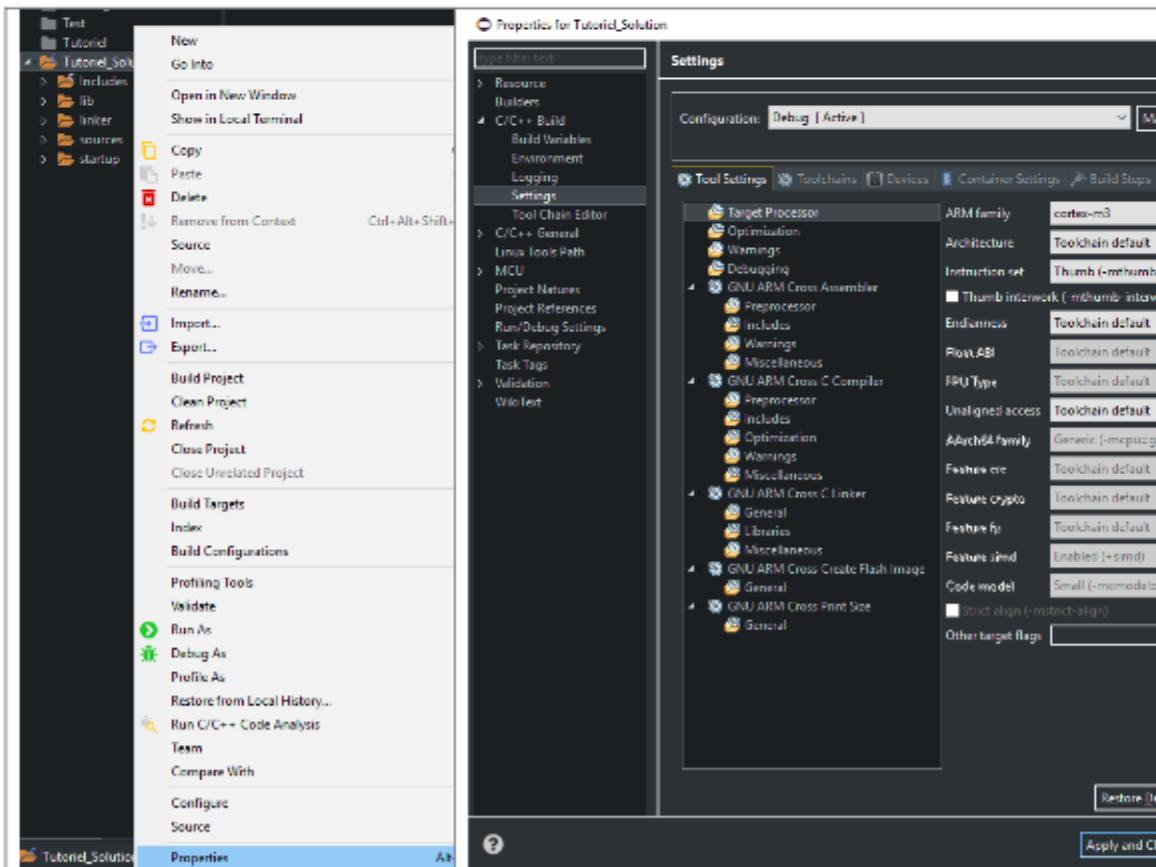
Le dossier compressé « Tutoriel_vX.X » qui vous a été envoyé par mail contient le dossier « Software ». A l'intérieur de celui-ci, vous trouverez 4 dossiers : les fichiers sources, la librairie, le linker et le startup. Ensuite copier/coller ces dossiers dans le dossier de votre projet



Clic droit sur le projet dans Eclipse et cliquer sur Refresh.

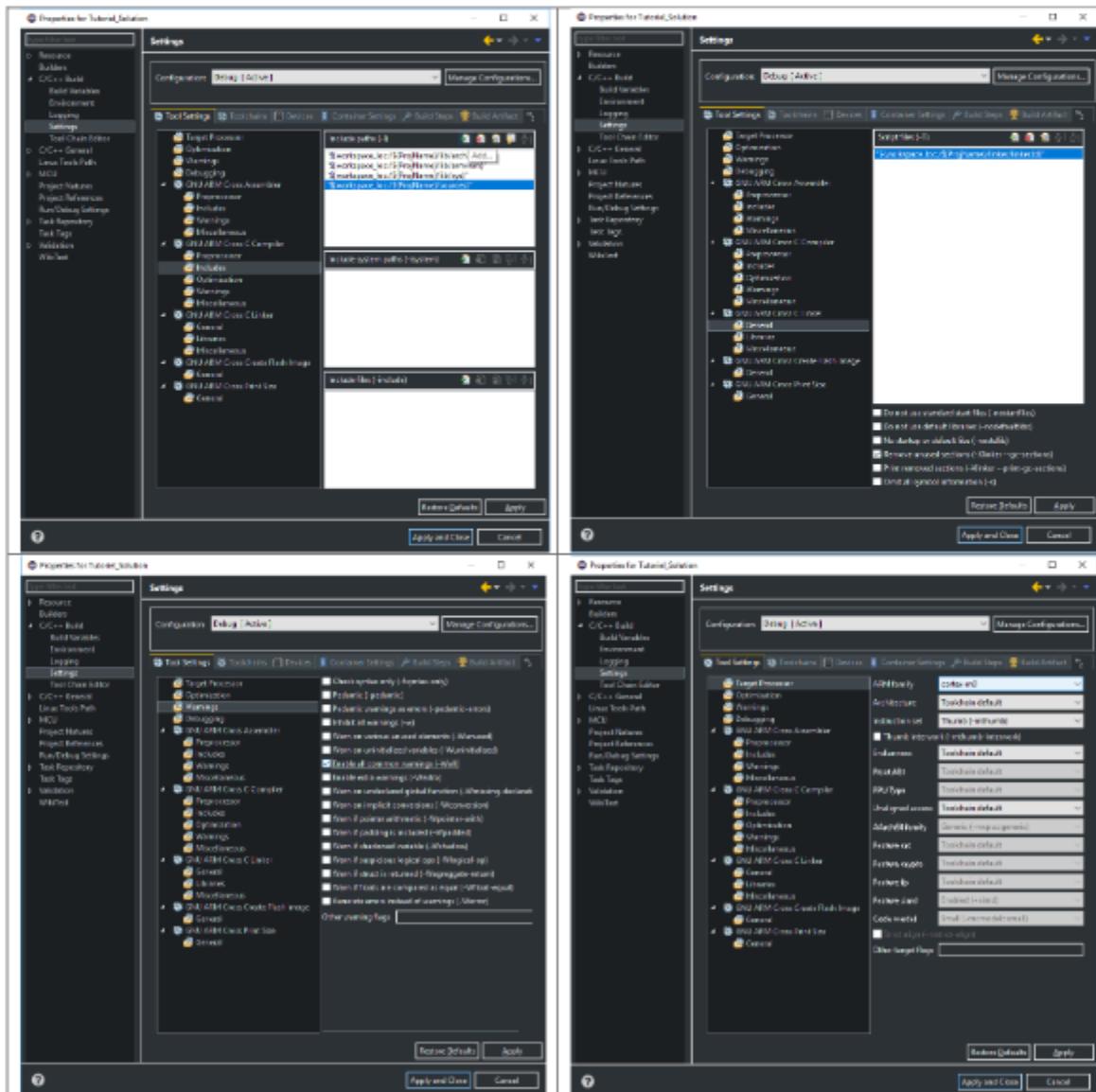


Dans l'IDE, sélectionner le projet et aller dans les Properties :



- Properties/C/C++ Build/Settings/Tool Settings/GNU ARM Cross C Compiler/Includes/Include paths
- Ajouter le dossier source et les 3 dossier librairies.
- Properties/C/C++ Build/Settings/Tool Settings/GNU ARM Cross C Linker/General
- Ajouter le fichier linker
- Properties/C/C++ Build/Settings/Tool Settings/Warnings
- Cocher la case « Enable all common warnings (-Wall) »
- Properties/C/C++ Build/Settings/Tool Settings/Target Processor
- Selectionner le cortex-m0 dans ARM family

Voici le résultat attendu :



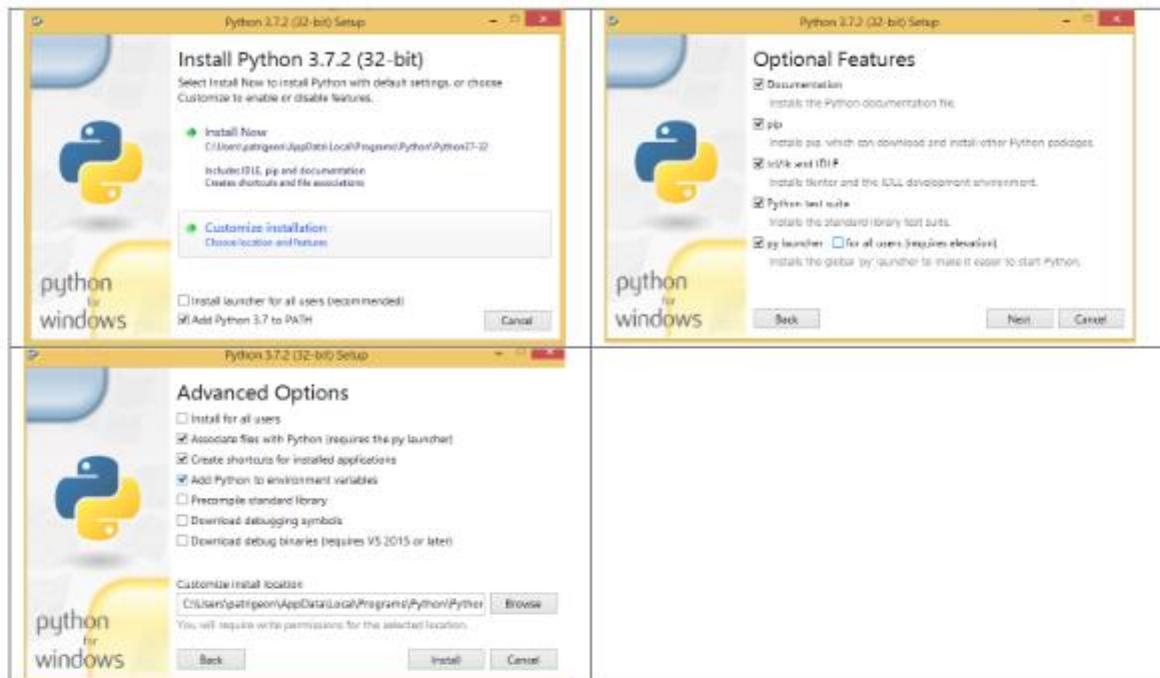
Compiler le projet en cliquant sur le marteau en haut à gauche :

```

1 // Main.c
2
3 #include "main.h"
4 #include "TIMERS.h"
5
6 void main(void)
7 {
8     int counter = 0;
9
10    systemInit();
11
12    printf("Hello ---- Tutorial Program Start Server ----\n");
13
14    while(1)
15    {
16        if (TIMERS_flag == 1)
17        {
18            systemPrint();
19        }
20
21        // Task 1
22        // Task 2
23
24        // Task 3
25
26        // Task 4
27
28        // Task 5
29
30        // Task 6
31
32        // Task 7
33
34        // Task 8
35
36        // Task 9
37
38        // Task 10
39
40        // Task 11
41
42        // Task 12
43
44        // Task 13
45
46        // Task 14
47
48        // Task 15
49
50        // Task 16
51
52        // Task 17
53
54        // Task 18
55
56        // Task 19
57
58        // Task 20
59
60        // Task 21
61
62        // Task 22
63
64        // Task 23
65
66        // Task 24
67
68        // Task 25
69
70        // Task 26
71
72        // Task 27
73
74        // Task 28
75
76        // Task 29
77
78        // Task 30
79
80        // Task 31
81
82        // Task 32
83
84        // Task 33
85
86        // Task 34
87
88        // Task 35
89
90        // Task 36
91
92        // Task 37
93
94        // Task 38
95
96        // Task 39
97
98        // Task 40
99
100       // Task 41
101
102       // Task 42
103
104       // Task 43
105
106       // Task 44
107
108       // Task 45
109
110       // Task 46
111
112       // Task 47
113
114       // Task 48
115
116       // Task 49
117
118       // Task 50
119
120       // Task 51
121
122       // Task 52
123
124       // Task 53
125
126       // Task 54
127
128       // Task 55
129
130       // Task 56
131
132       // Task 57
133
134       // Task 58
135
136       // Task 59
137
138       // Task 60
139
140       // Task 61
141
142       // Task 62
143
144       // Task 63
145
146       // Task 64
147
148       // Task 65
149
150       // Task 66
151
152       // Task 67
153
154       // Task 68
155
156       // Task 69
157
158       // Task 70
159
160       // Task 71
161
162       // Task 72
163
164       // Task 73
165
166       // Task 74
167
168       // Task 75
169
170       // Task 76
171
172       // Task 77
173
174       // Task 78
175
176       // Task 79
177
178       // Task 80
179
180       // Task 81
181
182       // Task 82
183
184       // Task 83
185
186       // Task 84
187
188       // Task 85
189
190       // Task 86
191
192       // Task 87
193
194       // Task 88
195
196       // Task 89
197
198       // Task 90
199
200       // Task 91
201
202       // Task 92
203
204       // Task 93
205
206       // Task 94
207
208       // Task 95
209
210       // Task 96
211
212       // Task 97
213
214       // Task 98
215
216       // Task 99
217
218       // Task 100
219
220       // Task 101
221
222       // Task 102
223
224       // Task 103
225
226       // Task 104
227
228       // Task 105
229
230       // Task 106
231
232       // Task 107
233
234       // Task 108
235
236       // Task 109
237
238       // Task 110
239
240       // Task 111
241
242       // Task 112
243
244       // Task 113
245
246       // Task 114
247
248       // Task 115
249
250       // Task 116
251
252       // Task 117
253
254       // Task 118
255
256       // Task 119
257
258       // Task 120
259
260       // Task 121
261
262       // Task 122
263
264       // Task 123
265
266       // Task 124
267
268       // Task 125
269
270       // Task 126
271
272       // Task 127
273
274       // Task 128
275
276       // Task 129
277
278       // Task 130
279
280       // Task 131
281
282       // Task 132
283
284       // Task 133
285
286       // Task 134
287
288       // Task 135
289
290       // Task 136
291
292       // Task 137
293
294       // Task 138
295
296       // Task 139
297
298       // Task 140
299
300       // Task 141
301
302       // Task 142
303
304       // Task 143
305
306       // Task 144
307
308       // Task 145
309
310       // Task 146
311
312       // Task 147
313
314       // Task 148
315
316       // Task 149
317
318       // Task 150
319
320       // Task 151
321
322       // Task 152
323
324       // Task 153
325
326       // Task 154
327
328       // Task 155
329
330       // Task 156
331
332       // Task 157
333
334       // Task 158
335
336       // Task 159
337
338       // Task 160
339
340       // Task 161
341
342       // Task 162
343
344       // Task 163
345
346       // Task 164
347
348       // Task 165
349
350       // Task 166
351
352       // Task 167
353
354       // Task 168
355
356       // Task 169
357
358       // Task 170
359
360       // Task 171
361
362       // Task 172
363
364       // Task 173
365
366       // Task 174
367
368       // Task 175
369
370       // Task 176
371
372       // Task 177
373
374       // Task 178
375
376       // Task 179
377
378       // Task 180
379
380       // Task 181
381
382       // Task 182
383
384       // Task 183
385
386       // Task 184
387
388       // Task 185
389
390       // Task 186
391
392       // Task 187
393
394       // Task 188
395
396       // Task 189
397
398       // Task 190
399
400       // Task 191
401
402       // Task 192
403
404       // Task 193
405
406       // Task 194
407
408       // Task 195
409
410       // Task 196
411
412       // Task 197
413
414       // Task 198
415
416       // Task 199
417
418       // Task 200
419
420       // Task 201
421
422       // Task 203
423
424       // Task 204
425
426       // Task 205
427
428       // Task 206
429
430       // Task 207
431
432       // Task 208
433
434       // Task 209
435
436       // Task 210
437
438       // Task 211
439
440       // Task 212
441
442       // Task 213
443
444       // Task 214
445
446       // Task 215
447
448       // Task 216
449
450       // Task 217
451
452       // Task 218
453
454       // Task 219
455
456       // Task 220
457
458       // Task 221
459
460       // Task 222
461
462       // Task 223
463
464       // Task 224
465
466       // Task 225
467
468       // Task 226
469
470       // Task 227
471
472       // Task 228
473
474       // Task 229
475
476       // Task 230
477
478       // Task 231
479
480       // Task 232
481
482       // Task 233
483
484       // Task 234
485
486       // Task 235
487
488       // Task 236
489
490       // Task 237
491
492       // Task 238
493
494       // Task 239
495
496       // Task 240
497
498       // Task 241
499
500       // Task 242
501
502       // Task 243
503
504       // Task 244
505
506       // Task 245
507
508       // Task 246
509
509       // Task 247
510
511       // Task 248
512
513       // Task 249
514
515       // Task 250
516
516       // Task 251
517
518       // Task 252
519
519       // Task 253
520
520       // Task 254
521
521       // Task 255
522
522       // Task 256
523
523       // Task 257
524
524       // Task 258
525
525       // Task 259
526
526       // Task 260
527
527       // Task 261
528
528       // Task 262
529
529       // Task 263
530
530       // Task 264
531
531       // Task 265
532
532       // Task 266
533
533       // Task 267
534
534       // Task 268
535
535       // Task 269
536
536       // Task 270
537
537       // Task 271
538
538       // Task 272
539
539       // Task 273
540
540       // Task 274
541
541       // Task 275
542
542       // Task 276
543
543       // Task 277
544
544       // Task 278
545
545       // Task 279
546
546       // Task 280
547
547       // Task 281
548
548       // Task 282
549
549       // Task 283
550
550       // Task 284
551
551       // Task 285
552
552       // Task 286
553
553       // Task 287
554
554       // Task 288
555
555       // Task 289
556
556       // Task 290
557
557       // Task 291
558
558       // Task 292
559
559       // Task 293
560
560       // Task 294
561
561       // Task 295
562
562       // Task 296
563
563       // Task 297
564
564       // Task 298
565
565       // Task 299
566
566       // Task 300
567
567       // Task 301
568
568       // Task 302
569
569       // Task 303
570
570       // Task 304
571
571       // Task 305
572
572       // Task 306
573
573       // Task 307
574
574       // Task 308
575
575       // Task 309
576
576       // Task 310
577
577       // Task 311
578
578       // Task 312
579
579       // Task 313
580
580       // Task 314
581
581       // Task 315
582
582       // Task 316
583
583       // Task 317
584
584       // Task 318
585
585       // Task 319
586
586       // Task 320
587
587       // Task 321
588
588       // Task 322
589
589       // Task 323
590
590       // Task 324
591
591       // Task 325
592
592       // Task 326
593
593       // Task 327
594
594       // Task 328
595
595       // Task 329
596
596       // Task 330
597
597       // Task 331
598
598       // Task 332
599
599       // Task 333
600
600       // Task 334
601
601       // Task 335
602
602       // Task 336
603
603       // Task 337
604
604       // Task 338
605
605       // Task 339
606
606       // Task 340
607
607       // Task 341
608
608       // Task 342
609
609       // Task 343
610
610       // Task 344
611
611       // Task 345
612
612       // Task 346
613
613       // Task 347
614
614       // Task 348
615
615       // Task 349
616
616       // Task 350
617
617       // Task 351
618
618       // Task 352
619
619       // Task 353
620
620       // Task 354
621
621       // Task 355
622
622       // Task 356
623
623       // Task 357
624
624       // Task 358
625
625       // Task 359
626
626       // Task 360
627
627       // Task 361
628
628       // Task 362
629
629       // Task 363
630
630       // Task 364
631
631       // Task 365
632
632       // Task 366
633
633       // Task 367
634
634       // Task 368
635
635       // Task 369
636
636       // Task 370
637
637       // Task 371
638
638       // Task 372
639
639       // Task 373
640
640       // Task 374
641
641       // Task 375
642
642       // Task 376
643
643       // Task 377
644
644       // Task 378
645
645       // Task 379
646
646       // Task 380
647
647       // Task 381
648
648       // Task 382
649
649       // Task 383
650
650       // Task 384
651
651       // Task 385
652
652       // Task 386
653
653       // Task 387
654
654       // Task 388
655
655       // Task 389
656
656       // Task 390
657
657       // Task 391
658
658       // Task 392
659
659       // Task 393
660
660       // Task 394
661
661       // Task 395
662
662       // Task 396
663
663       // Task 397
664
664       // Task 398
665
665       // Task 399
666
666       // Task 400
667
667       // Task 401
668
668       // Task 402
669
669       // Task 403
670
670       // Task 404
671
671       // Task 405
672
672       // Task 406
673
673       // Task 407
674
674       // Task 408
675
675       // Task 409
676
676       // Task 410
677
677       // Task 411
678
678       // Task 412
679
679       // Task 413
680
680       // Task 414
681
681       // Task 415
682
682       // Task 416
683
683       // Task 417
684
684       // Task 418
685
685       // Task 419
686
686       // Task 420
687
687       // Task 421
688
688       // Task 422
689
689       // Task 423
690
690       // Task 424
691
691       // Task 425
692
692       // Task 426
693
693       // Task 427
694
694       // Task 428
695
695       // Task 429
696
696       // Task 430
697
697       // Task 431
698
698       // Task 432
699
699       // Task 433
699       // Task 434
700
700       // Task 435
701
701       // Task 436
702
702       // Task 437
703
703       // Task 438
704
704       // Task 439
705
705       // Task 440
706
706       // Task 441
707
707       // Task 442
708
708       // Task 443
709
709       // Task 444
710
710       // Task 445
711
711       // Task 446
712
712       // Task 447
713
713       // Task 448
714
714       // Task 449
715
715       // Task 450
716
716       // Task 451
717
717       // Task 452
718
718       // Task 453
719
719       // Task 454
720
720       // Task 455
721
721       // Task 456
722
722       // Task 457
723
723       // Task 458
724
724       // Task 459
725
725       // Task 460
726
726       // Task 461
727
727       // Task 462
728
728       // Task 463
729
729       // Task 464
730
730       // Task 465
731
731       // Task 466
732
732       // Task 467
733
733       // Task 468
734
734       // Task 469
735
735       // Task 470
736
736       // Task 471
737
737       // Task 472
738
738       // Task 473
739
739       // Task 474
740
740       // Task 475
741
741       // Task 476
742
742       // Task 477
743
743       // Task 478
744
744       // Task 479
745
745       // Task 480
746
746       // Task 481
747
747       // Task 482
748
748       // Task 483
749
749       // Task 484
750
750       // Task 485
751
751       // Task 486
752
752       // Task 487
753
753       // Task 488
754
754       // Task 489
755
755       // Task 490
756
756       // Task 491
757
757       // Task 492
758
758       // Task 493
759
759       // Task 494
760
760       // Task 495
761
761       // Task 496
762
762       // Task 497
763
763       // Task 498
764
764       // Task 499
765
765       // Task 500
766
766       // Task 501
767
767       // Task 502
768
768       // Task 503
769
769       // Task 504
770
770       // Task 505
771
771       // Task 506
772
772       // Task 507
773
773       // Task 508
774
774       // Task 509
775
775       // Task 510
776
776       // Task 511
777
777       // Task 512
778
778       // Task 513
779
779       // Task 514
780
780       // Task 515
781
781       // Task 516
782
782       // Task 517
783
783       // Task 518
784
784       // Task 519
785
785       // Task 520
786
786       // Task 521
787
787       // Task 522
788
788       // Task 523
789
789       // Task 524
790
790       // Task 525
791
791       // Task 526
792
792       // Task 527
793
793       // Task 528
794
794       // Task 529
795
795       // Task 530
796
796       // Task 531
797
797       // Task 532
798
798       // Task 533
799
799       // Task 534
800
800       // Task 535
801
801       // Task 536
802
802       // Task 537
803
803       // Task 538
804
804       // Task 539
805
805       // Task 540
806
806       // Task 541
807
807       // Task 542
808
808       // Task 543
809
809       // Task 544
810
810       // Task 545
811
811       // Task 546
812
812       // Task 547
813
813       // Task 548
814
814       // Task 549
815
815       // Task 550
816
816       // Task 551
817
817       // Task 552
818
818       // Task 553
819
819       // Task 554
820
820       // Task 555
821
821       // Task 556
822
822       // Task 557
823
823       // Task 558
824
824       // Task 559
825
825       // Task 560
826
826       // Task 561
827
827       // Task 562
828
828       // Task 563
829
829       // Task 564
830
830       // Task 565
831
831       // Task 566
832
832       // Task 567
833
833       // Task 568
834
834       // Task 569
835
835       // Task 570
836
836       // Task 571
837
837       // Task 572
838
838       // Task 573
839
839       // Task 574
840
840       // Task 575
841
841       // Task 576
842
842       // Task 577
843
843       // Task 578
844
844       // Task 579
845
845       // Task 580
846
846       // Task 581
847
847       // Task 582
848
848       // Task 583
849
849       // Task 584
850
850       // Task 585
851
851       // Task 586
852
852       // Task 587
853
853       // Task 588
854
854       // Task 589
855
855       // Task 590
856
856       // Task 591
857
857       // Task 592
858
858       // Task 593
859
859       // Task 594
860
860       // Task 595
861
861       // Task 596
862
862       // Task 597
863
863       // Task 598
864
864       // Task 599
865
865       // Task 600
866
866       // Task 601
867
867       // Task 602
868
868       // Task 603
869
869       // Task 604
870
870       // Task 605
871
871       // Task 606
872
872       // Task 607
873
873       // Task 608
874
874       // Task 609
875
875       // Task 610
876
876       // Task 611
877
877       // Task 612
878
878       // Task 613
879
879       // Task 614
880
880       // Task 615
881
881       // Task 616
882
882       // Task 617
883
883       // Task 618
884
884       // Task 619
885
885       // Task 620
886
886       // Task 621
887
887       // Task 622
888
888       // Task 623
889
889       // Task 624
890
890       // Task 625
891
891       // Task 626
892
892       // Task 627
893
893       // Task 628
894
894       // Task 629
895
895       // Task 630
896
896       // Task 631
897
897       // Task 632
898
898       // Task 633
899
899       // Task 634
900
900       // Task 635
901
901       // Task 636
902
902       // Task 637
903
903       // Task 638
904
904       // Task 639
905
905       // Task 640
906
906       // Task 641
907
907       // Task 642
908
908       // Task 643
909
909       // Task 644
910
910       // Task 645
911
911       // Task 646
912
912       // Task 647
913
913       // Task 648
914
914       // Task 649
915
915       // Task 650
916
916       // Task 651
917
917       // Task 652
918
918       // Task 653
919
919       // Task 654
920
920       // Task 655
921
921       // Task 656
922
922       // Task 657
923
923       // Task 658
924
924       // Task 659
925
925       // Task 660
926
926       // Task 661
927
927       // Task 662
928
928       // Task 663
929
929       // Task 664
930
930       // Task 665
931
931       // Task 666
932
932       // Task 667
933
933       // Task 668
934
934       // Task 669
935
935       // Task 670
936
936       // Task 671
937
937       // Task 672
938
938       // Task 673
939
939       // Task 674
940
940       // Task 675
941
941       // Task 676
942
942       // Task 677
943
943       // Task 678
944
944       // Task 679
945
945       // Task 680
946
946       // Task 681
947
947       // Task 682
948
948       // Task 683
949
949       // Task 684
950
950       // Task 685
951
951       // Task 686
952
952       // Task 687
953
953       // Task 688
954
954       // Task 689
955
955       // Task 690
956
956       // Task 691
957
957       // Task 692
958
958       // Task 693
959
959       // Task 694
960
960       // Task 695
961
961       // Task 696
962
962       // Task 697
963
963       // Task 698
964
964       // Task 699
965
965       // Task 700
966
966       // Task 701
967
967       // Task 702
968
968       // Task 703
969
969       // Task 704
970
970       // Task 705
971
971       // Task 706
972
972       // Task 707
973
973       // Task 708
974
974       // Task 709
975
975       // Task 710
976
976       // Task 711
977
977       // Task 712
978
978       // Task 713
979
979       // Task 714
980
980       // Task 715
981
981       // Task 716
982
982       // Task 717
983
983       // Task 718
984
984       // Task 719
985
985       // Task 720
986
986       // Task 721
987
987       // Task 722
988
988       // Task 723
989
989       // Task 724
990
990       // Task 725
991
991       // Task 726
992
992       // Task 727
993
993       // Task 728
994
994       // Task 729
995
995       // Task 730
996
996       // Task 731
997
997       // Task 732
998
998       // Task 733
999
999       // Task 734
1000
1000      // Task 735
1001
1001      // Task
```

Installation du Codeloader

Lancer Python .exe



Taper cmd dans la barre de recherche windows

Taper : pip install pyserial et taper enter

```

Invite de commandes

Microsoft Windows [version 6.3.9600]
(c) 2013 Microsoft Corporation. Tous droits réservés.

P:>pip install pyserial
Collecting pyserial
  Downloading https://files.pythonhosted.org/packages/0d/e4/2a744dd9e3be04a0c0907414e2a01a7c88bb3915che3c8cc06e209f59c30/pyserial-3.4-py2.py3-none-any.whl (193kB)
    100% |████████████████████████████████| 194kB 2.2MB/s
Installing collected packages: pyserial
Successfully installed pyserial-3.4

P:>_

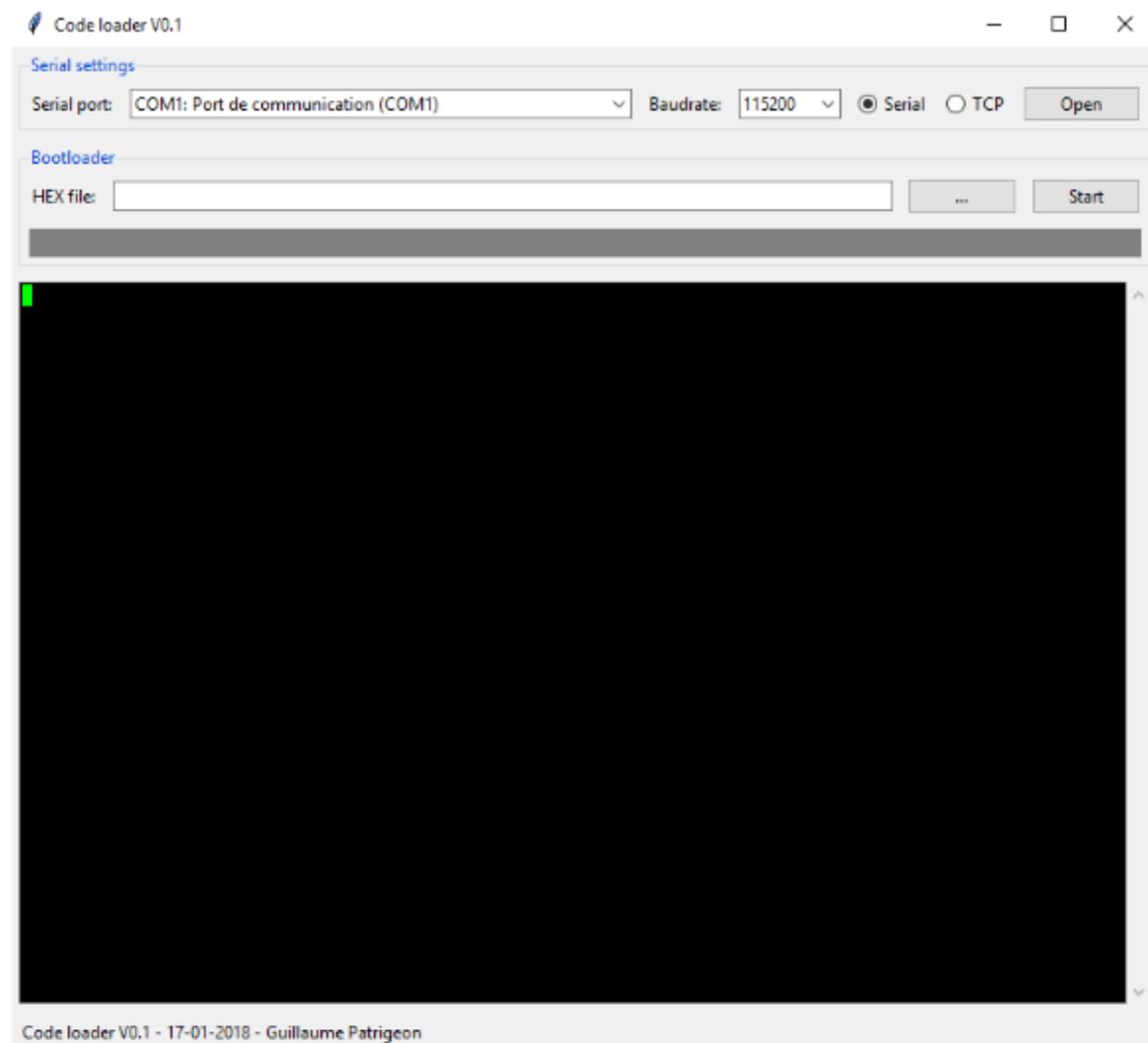
```

Lancer main.pyw pour vérifier la bonne installation !!!!

Il se trouve dans le dossier compressé du tutoriel dans le dossier Codeloader. Décompresser le dossier préalablement.

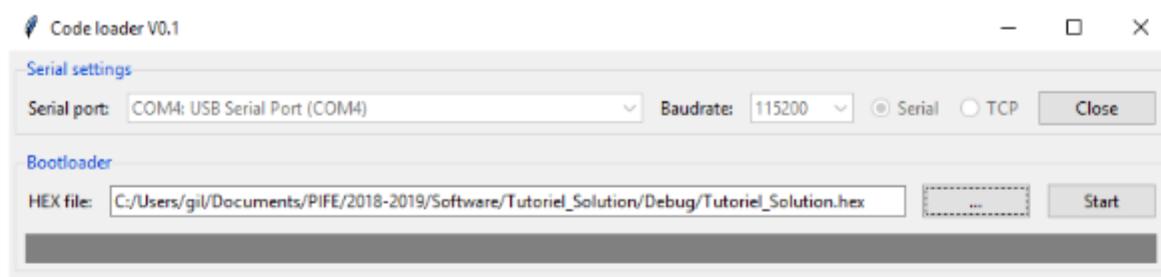
Utilisation du Codeloader

Double cliquez sur le main.pyw pour ouvrir le codeloader, cette fenêtre devra apparaître :

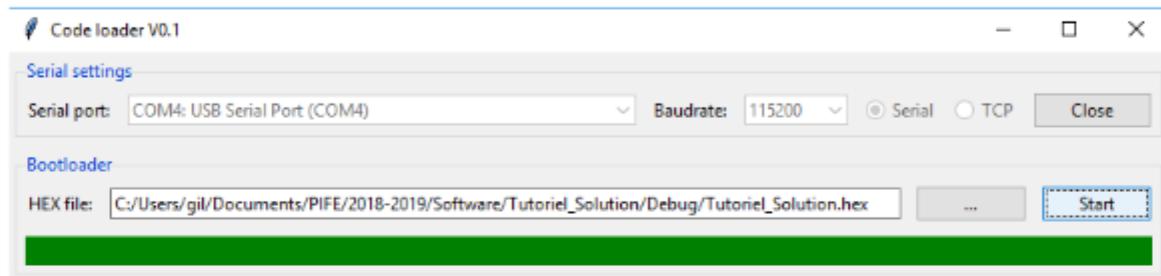


Une fois le Codeloader lancé, sélectionner le port COM où est branché le CmodA7, mettre un Baudrate de 115200 en liaison Serial puis ouvrir la liaison série.

Sélectionner ensuite le fichier hex file et télécharger le code.

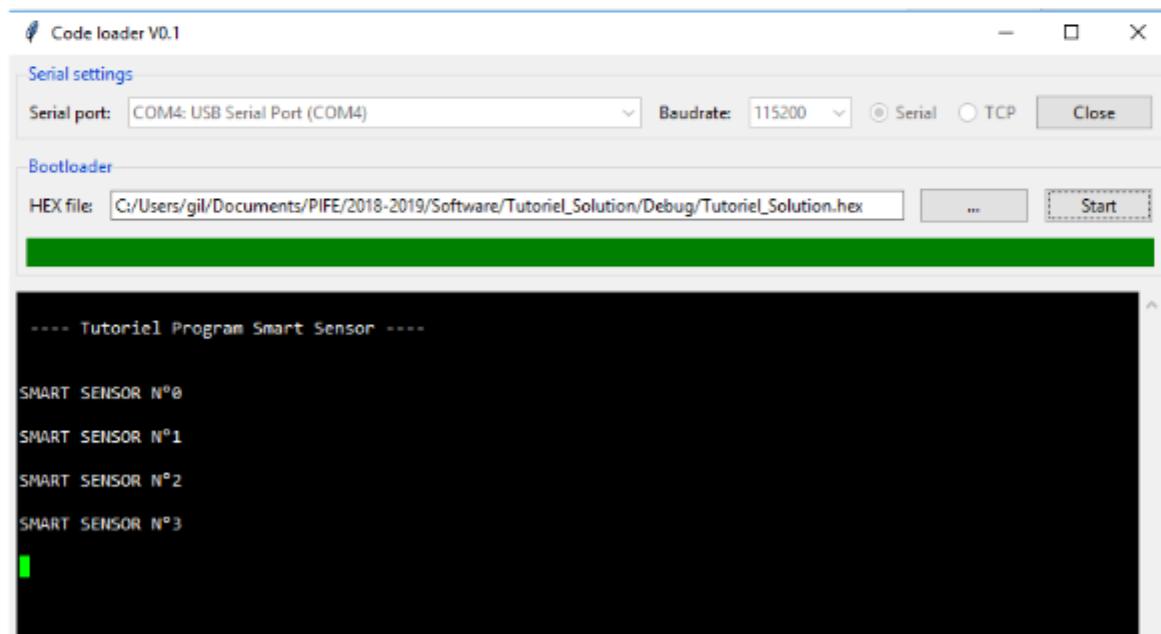


Pour télécharger le code, il faut appuyer sur start et rapidement appuyer sur le reset du CmodA7 qui est le bouton le plus proche de la connectique Pmod.



Si le téléchargement est réussi, la barre de chargement est verte, si le reset n'a pas été appuyé à temps, la barre deviendra rouge.

Si tout a été fait correctement, des informations s'affichent à l'écran et lorsque l'on appuie le bouton BTN1, la LED1 s'allume.



Faire clignoter la LED 2

Adapter le code pour faire clignoter la LED2 avec une période de 2 secondes

2.2. Utiliser l'interface UART avec le module GPS

Initialiser l'UART2

Il faut aller dans le system.c pour initialiser les PIN en entrée/sortie de l'UART2 puis définir l'utilisation de l'UART2 dans uart.setting.h

Les afficher

Renvoyer les données de l'UART2 sur l'UART1 pour l'afficher à l'écran.

Les stocker

Renvoyer les données de l'UART2 dans un tableau, puis afficher le tableau à l'écran.

2.3. Mettre sous un bon format

Machine d'état

Faire une fonction dans GPS.c pour n'afficher qu'une seule trame. Celle qui est intéressante pour le projet (date, heure, latitude, longitude). Voir la signification des trames sur le datasheet du module.

Mettre cette trame dans un tableau et afficher ce tableau.

Ensuite afficher l'heure, la date de manière lisible pour quelqu'un qui ne connaît pas la structure de la trame.

Créer une trame

Créer une trame avec les informations intéressantes dans le but de l'envoyer plus tard par un module LoRa.

3. Pour approfondir

3.1. Utiliser les autres interfaces du CmodA7, SPI et I2C

3.2. Envoyer la trame par LoRa

