**STATION MÉTÉO à base de Raspberry Pi**

**Alexia TURINES / Maxime COLINET**

# **Historique du rapport**

Le tableau suivant fournit un récapitulatif des principales modifications effectuées dans le rapport.

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Auteur | Description de la mise à jour |
| V0 | 12/04/2019 |  | Création du rapport |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table des matières**

# Historique du rapport 2

# [Cahier Des Charges](#_Toc338969004) 4

## [Périmètre du projet](#_Toc338969005) 4

## [L’équipe](#_Toc338969006) projet 4

## Tâches, jalons, livrables

## Planification du projet

## Gantt : Plan de communication, Démarche qualité au cours du projet

# Cahier Des Charges Fonctionnel 4

## Contexte

# Retour d’expérience

## Indicateurs de pilotage

## Avantages/inconvénients de l’environnement technique

# 

# Historique du rapport

Le tableau suivant fournit un récapitulatif des principales modifications effectuées dans le rapport.

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Auteur | Description de la mise à jour |
| V0 | 12/04/2019 |  | Création du rapport |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# **Cahier Des Charges**

* 1. **Périmètre du projet**

Ce projet s’insère dans la formation C++ Linux embarqué qui va permettre de mettre en application les acquis de cette formation. Celui-ci va mettre en œuvre un projet C sur une carte Raspberry Pi.

L’objectif de ce projet est la réalisation d’une station météo avec pour contraintes matérielles l’utilisation d’un Raspberry Pi 3 B+ et d’un capteur BME280 à travers une interface graphique.

* 1. **L’équipe projet**

L’équipe se compose de 2 personnes : Alexia et Maxime.

Le profil de l’équipe est le suivant :

* Alexia : ingénieur à dominante qualité (système et produit) avec une double compétence en système d’information
* Maxime : ingénieur informatique orienté sécurité informatique

Pour l’organisation du projet, Alexia s’est occupée d’une grande partie de la maîtrise d’ouvrage ainsi que de la partie interface graphique et Maxime s’est occupé d’une grande partie de la maîtrise d’œuvre ainsi que des tests fonctionnels.

* 1. **Tâches, jalons et livrables**

La réunion d’ouverture du projet a eu lieu le vendredi 12 avril 2019. Nous avons défini les différentes tâches, les jalons et les livrables du projet.

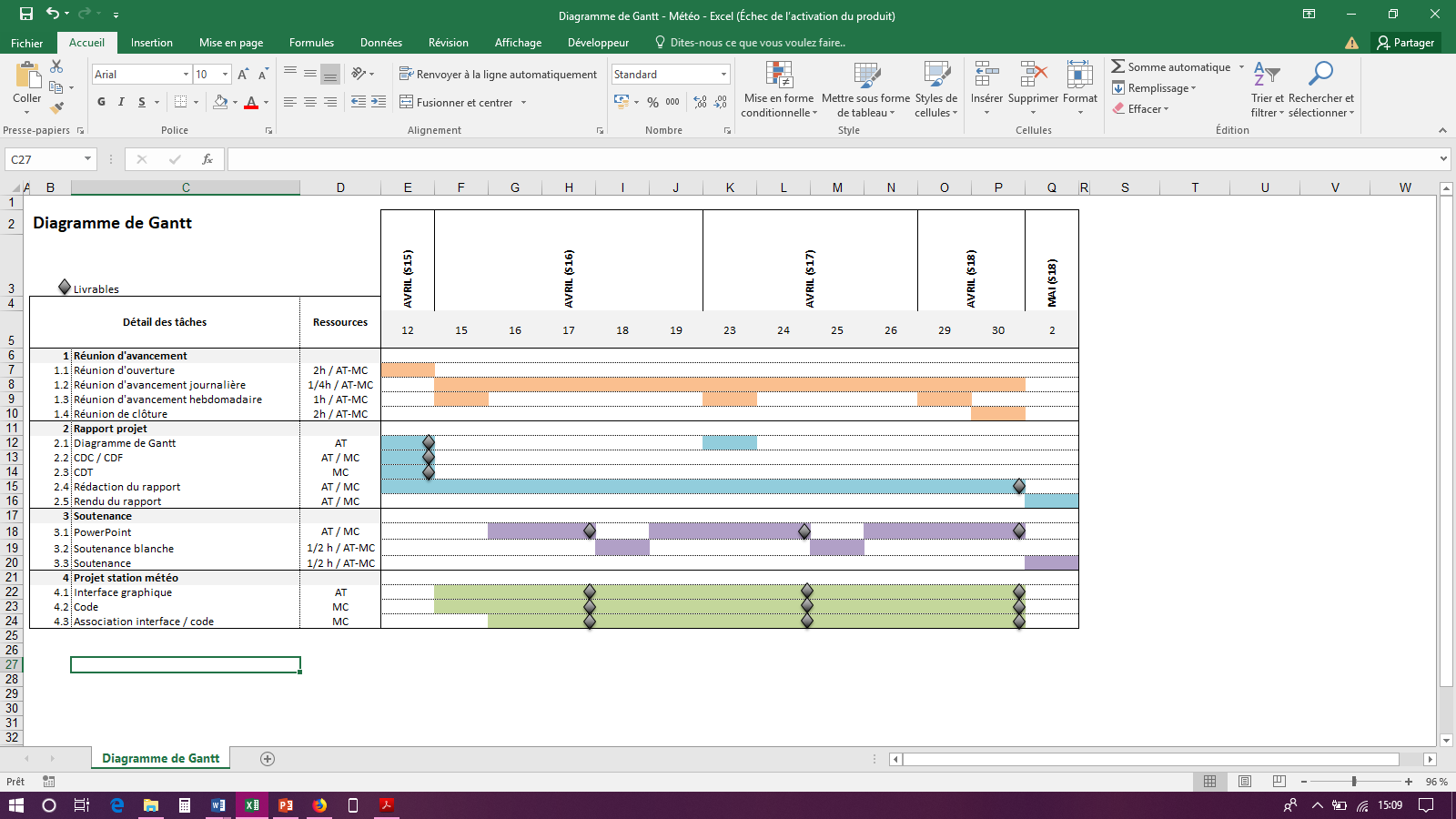
Les tâches qui ont été définies sont les suivantes :

* Les réunions d’avancement du projet avec la réunion de clôture, les réunions d’avancement journalières (1/4h), les réunions d’avancement hebdomadaires (1h) et la réunion de clôture
* Le rapport du projet avec la réalisation d’un cahier des charges, d’un cahier des charges fonctionnel et technique, un diagramme de Gantt, sa rédaction et son rendu
* La présentation du projet devant un jury avec la préparation d’un PowerPoint, 2 soutenances blanches (prévues le jeudi 18 et 25 avril d’une durée de 30 min) et la présentation finale le jeudi 2 mai (durée de 30 min)
* La réalisation du projet en lui-même avec la création de l’interface graphique, le programme et l’association interface graphique/programme.

Les livrables sont donc un rapport écrit, un PowerPoint de présentation et la station météo en fonctionnement. Les jalons ont été définis en fonction des soutenances blanches et de la présentation finale.

La dead-line du projet est le mardi 30 avril.

* 1. **Planification du projet**



# **Cahier Des Charges Fonctionnels**

* 1. . **Contexte**

Les ordinateurs embarqués fonctionnant sous le système d’exploitation Linux sont massivement présents dans les technologies modernes (transports, multimédia, téléphonie mobile, appareils photos …).

Les systèmes embarqués sous Linux disposent généralement de peu de RAM et utilisent fréquemment de la mémoire flash plutôt qu'un disque dur. Comme ils sont souvent dédiés à un nombre de tâches réduites sur une cible matérielle bien définie, ils utilisent plutôt des versions du noyau Linux optimisées pour des contextes précis.

L’objectif de ce projet est de réaliser une station météo avec un Raspberry Pi 3 B+ et un capteur BME280 qui permettra de mesurer la température, la pression atmosphérique et l’humidité.

**1.2. Produit**

Dans ce chapitre, nous allons expliquer les moyens mis en œuvre pour réaliser ce projet en définissant le produit et ses composants.

Le Raspberry PI 3.B + est une carte électronique comprenant un processeur intégré Quad-core ARM Cortex-A53 1.2 GHz (Broadcom BCM2837), RAM : 1024 Mo, GPU Dual Core VideoCore IV MultiMedia Co-Processor, lecteur de cartes Micro SD, ports : HDMI, 4x USB, RJ45, jack 3.5 mm, connecteurs pour APN et écran tactile, Wi-Fi b/g/n et Bluetooth 4.1.

Il est destiné à être piloté par un système Linux. Nous avons utilisé la distribution Raspbian qui est une version Debian avril 2019 avec un kernel version 4.14. Cette distribution Linux est installée sur une carte SD de 16 Go.

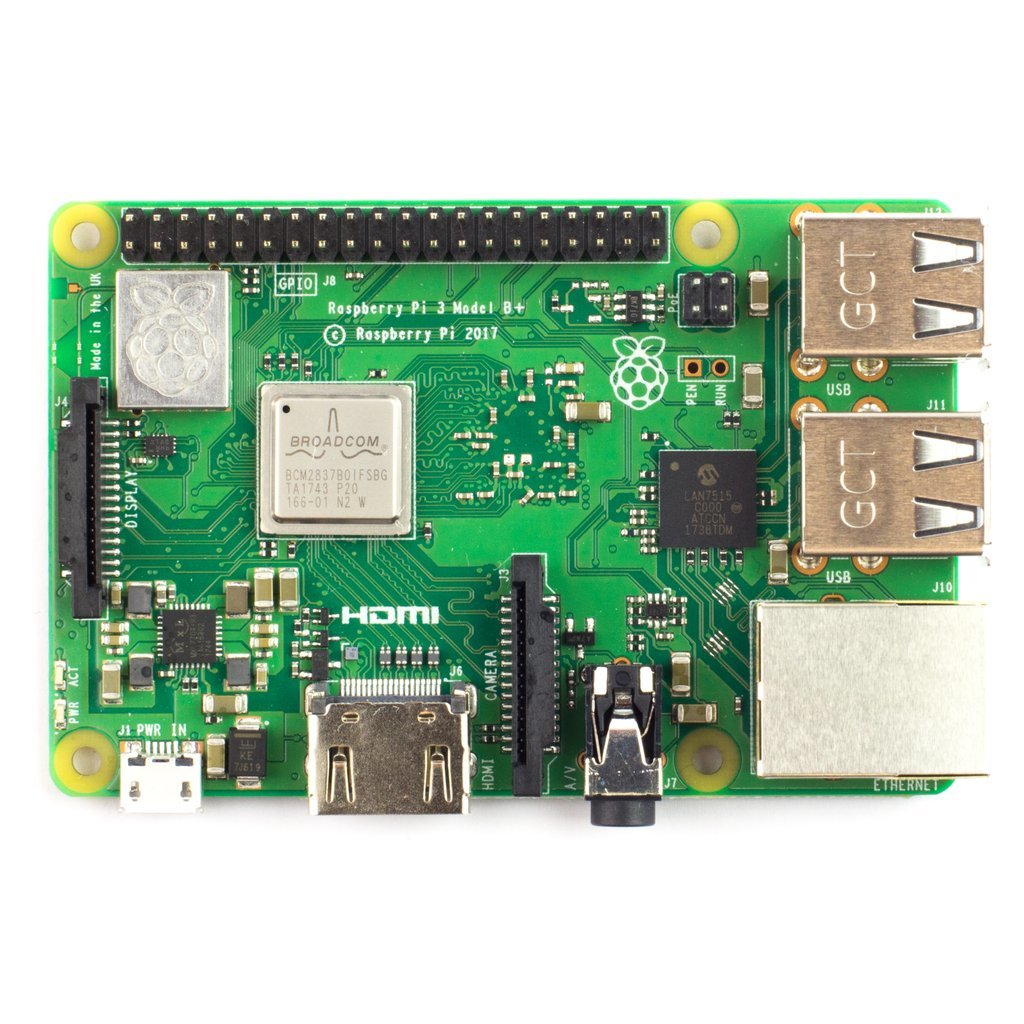


Fig.1 : Vue du dessus d’un Raspberry Pi 3 B+

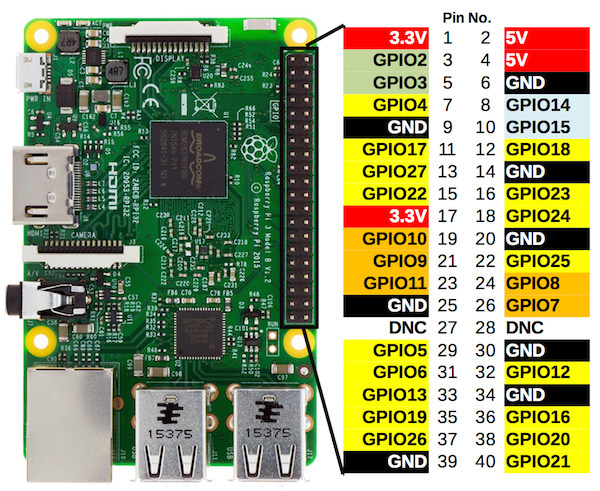
Le capteur BME280 est un circuit qui concentre un thermomètre, un mesureur de pression atmosphérique et un mesureur d’humidité. Il peut mesurer la température entre -40°C et 85°C avec une précision d’un degré, la pression entre 300hPa et 1100hPa à +/- 1hPa et l’humidité entre 0 et 100%.

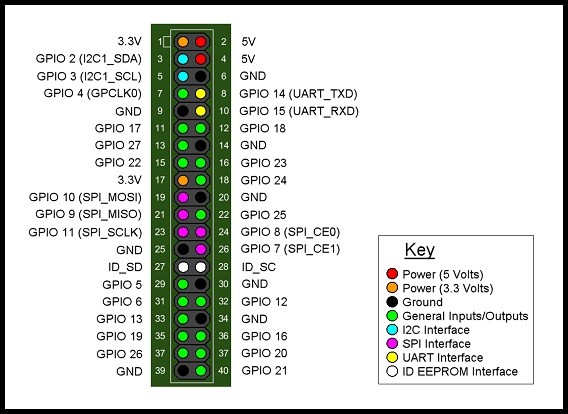


Fig.2 : Vue du dessus d’un capteur BME280

Il peut se connecter en SPI ou en I2C, accepte une tension d’alimentation jusqu’à 3.6V et consomme très peu de courant. Les connections seront les suivantes pour l’I2C :

|  |  |
| --- | --- |
| **BME280** | **Raspberry** |
| VCC | 3.3V : Pin 1 ou 17 (surtout pas le 5V) |
| GND | Masse : Pin 6, 9, 14, 30, 34 ou 39 |
| SDI | SDA : Pin 3 |
| SCK | SCL : Pin 5 |
| CSB | 3.3V Pin 1 ou 17 |
| SDO | Masse : adresse 0x76 ou 3.3V : adresse 0x77 |





Les méthodes permettant d’avoir la température, la pression et l’humidité avec la compensation mémorisée dans le module sont très complexes.

En pratique, les librairies permettant la gestion du module comprendront une partie commune pour les fonctionnalités et une partie spécifique pour la lecture du module en I2C.

La partie commune sera une classe abstraite qui ne sera pas utilisable seule, elle devra obligatoirement être étendue par le module I2C qui contiendra les fonctions de lecture et d’écriture dans les registres du module ainsi que les constructeurs.

Les principales fonctions disponibles seront les suivantes :

- Des fonctions permettant de lire l’identifiant du module, de faire un Reset, de régler la finesse des mesures en fonction de l’utilisation.

- Une fonction permettant de lire toutes les mesures en même temps afin d’avoir des mesures de température, pression et humidité cohérente entre elles. Cette fonction devra être appelée avant tout utilisation ou affichage des mesures (getAllMesures).

- Une fonction donnant la température (getTemperature).

- Deux fonctions donnant directement la pression mesurée (getPressurePa, getpressureHectoPa).

- Une fonction donnant l’humidité (getHumidity).

- Une fonction donnant l’altitude. Toutefois, cette fonction n’est valable que si la pression au niveau de la mer est de 1013hPa (getAltitude).

- Une fonction donnant l’altitude en fonction de la pression au niveau de la mer (getCompensedAltitude). C’est cette fonction qui peut être employée pour utiliser le module en altimètre.

- Deux fonctions donnant la pression au niveau de la mer en fonction de l’altitude de votre point de mesure. C’est cette pression qui est donnée dans les bulletins météo.

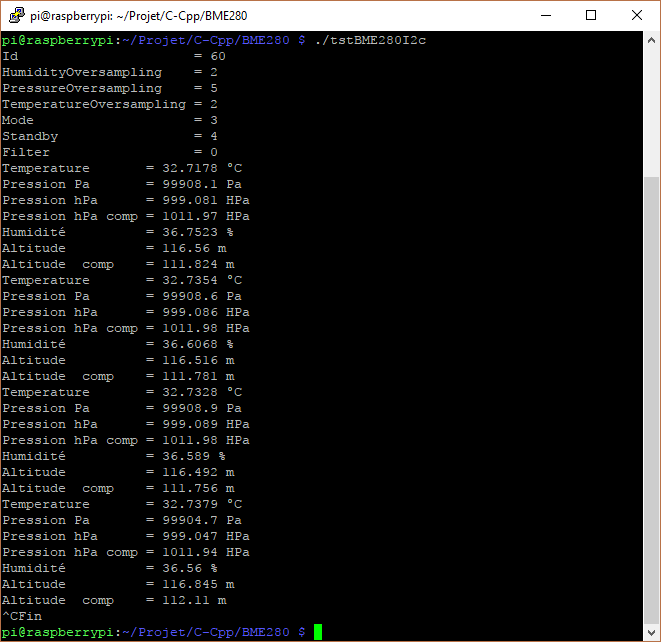
Voici la liste des fichiers sources et leur utilité :

|  |  |
| --- | --- |
| **Fichier** | **Utilité** |
| GestionBME280.h | Entête de la librairie des fonctions principales |
| GestionBME280.cpp | Librairie des fonctions principales |
| GestionSPI.h | Entête de gestion des fonctionnalités SPI |
| GestionBME280Spi.h | Entête de la librairie des fonctions spécifiques pour le SPI |
| GestionBME280Spi.cpp | Librairie des fonctions spécifiques pour le SPI |
| GestionBME280I2c.h | Entête de la librairie des fonctions spécifiques pour l’I2C |
| GestionBME280I2c.cpp | Librairie des fonctions spécifiques pour l’I2C |
| tstBME280Spi.cpp | Programme de test pour le câblage en SPI |
| tstBME280I2c.cpp | Programme de test pour le câblage en I2C |

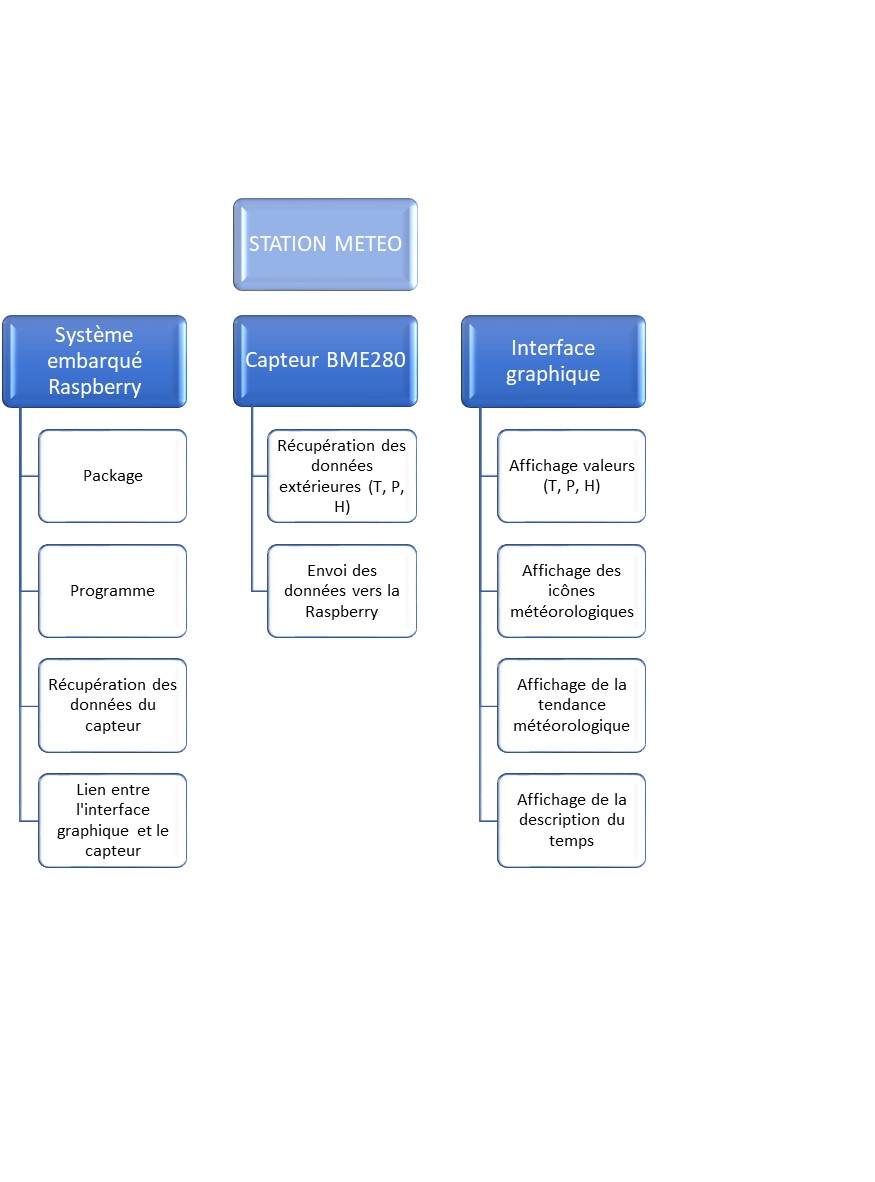
Voici la commande permettant la compilation des programmes de test pour le module câblé en I2C :

http://raspberrypi.pagesperso-orange.fr/dossiers/26-19_fichiers/image008.jpg

Et voici le résultat de l’exécution :



**STORY MAPPING**



**Retour d’expérience**

Voici quelques indicateurs que l'on peut trouver sur un tableau de bord:

* Utilisation des ressources (en %)
* Tâches réalisées/tâches planifiées
* Jalons
* Date de fin initiale
* Date de fin finale
* Avancement en délai (%)
* Nombre de tâches terminées par rapport au nombre de tâches prévues
* Nombre de changements
* Nombre de risques réalisés

#### La démarche qualité au cours du projet

La démarche qualité doit s�intégrer à toutes les phases du projet.

**Phases d�études préalables et de définition fonctionnelle du besoin**

Lors des phases d�étude préalable et de définition fonctionnelle des besoins, la qualité se caractérise par la capacité des études produites à adresser les véritables objectifs du projet et de complètement satisfaire les attentes associées. Lors de ces phases, les contrôles qualité reposent d�une part, sur la vérification du respect des principes de la méthodologie mise en �uvre et d�autre part, sur des revues visant à garantir l�adéquation et la cohérence des solutions proposées avec les attentes entre les différentes phases (schéma directeur, puis analyse préalable puis définition fonctionnelle du besoin). Le processus qualité s�attachera également à s�assurer du bon fonctionnement du cycle de validation pour vérifier que tous les principes et solutions proposés ont fait l�objet d�une validation ad hoc.

**Phases de réalisation**

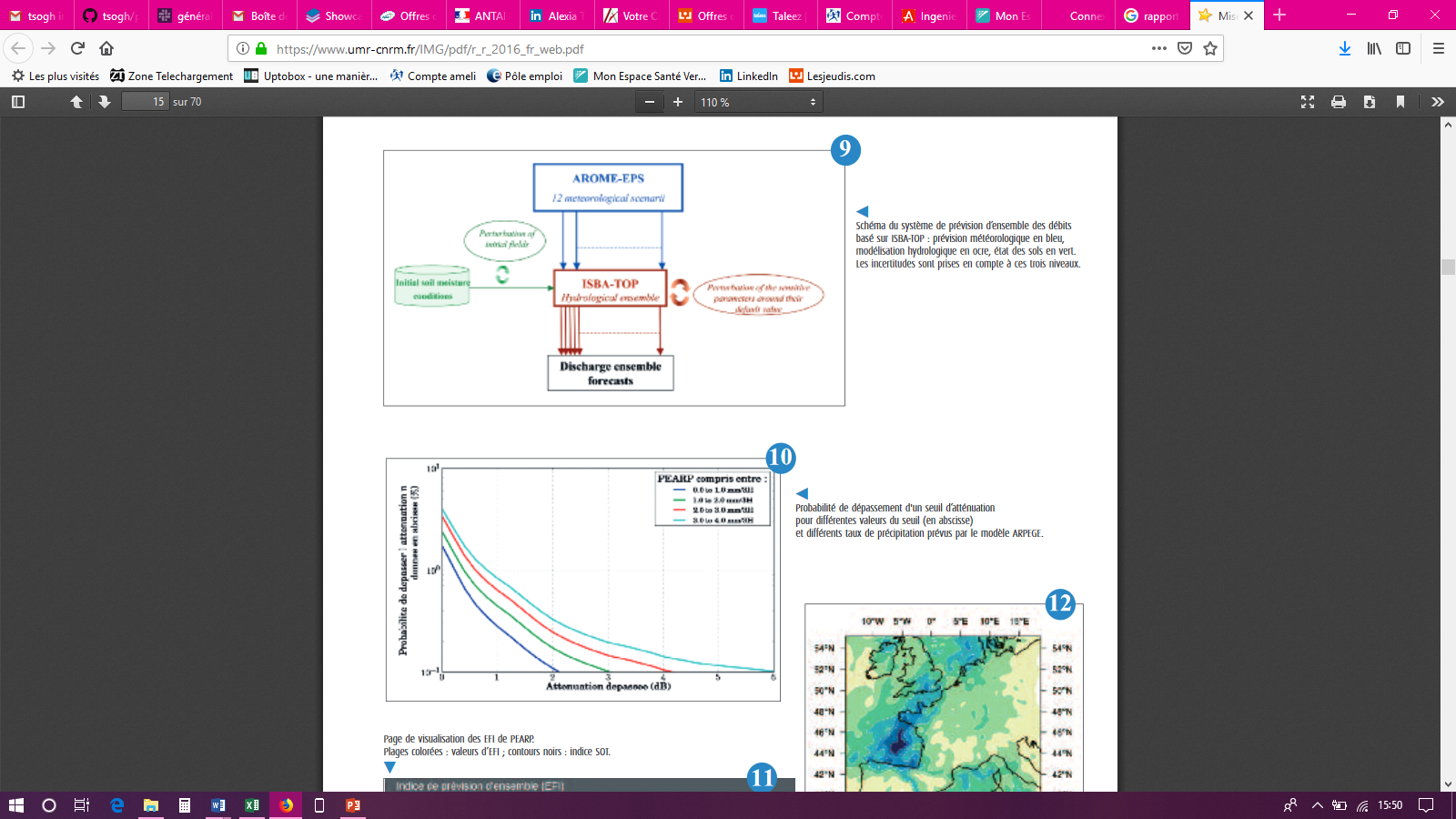
Le contrôle de la qualité porte d�une part, sur la vérification de la méthodologie mise en �uvre et d�autre part, sur des revues de code permettant de garantir les performances et la maintenabilité des applications développées.

**Phases de test**

Les phases de test permettent de vérifier d�une part, le bon fonctionnement intrinsèque des applications livrées et d�autre part, l�adéquation entre les fonctions réalisées par ces applicatifs et les fonctions spécifiées dans les dossiers de définition du besoin. En conséquence, lors de ces phases, le contrôle de qualité va consister à vérifier le bon fonctionnement de ce processus.

**Contrôle**

Contrôle de l�élaboration des fiches de tests (cycle de test, cas de test, description des jeux de test, description des résultats attendus),



**LE CODE**

**LA BASE DE DONNEES**

**Interface de visualisation des données**

**Affichage des courbes**

**Affichage des données via l’interface graphique**