[2/05/2019]

Rapport Station Météorologique

Projet Weather Pi

[Conception, Réalisation, et Analyse]

Alix DUMAY & Mehdi TAGUEMA

RAPPORT STATION METEOROLOGIQUE

Version	Date	Modifications	Auteur
v0	09/04/2019	Création du document / plan	Alix Dumay & Mehdi Taguema
v0-2	11/04/2019	Mise en forme + Ecriture Introduction et I/	Alix Dumay & Mehdi Taguema
V0-3	17/04/2019	Ecriture III/	Alix Dumay & Mehdi Taguema
V0-4	22/04/2019	Ecriture IV/ et V/	Alix Dumay & Mehdi Taguema
V0-5	24/04/2019	Ajout partie Méthodologie	Alix Dumay & Mehdi Taguema
V0-6	26/04/2019	Ecriture Conclusion	Alix Dumay & Mehdi Taguema
V0-7	28/04/19	Correction orthographique / syntaxe	Alix Dumay & Mehdi Taguema
V0-8	29/04/2019	Ecriture VI/	Alix Dumay & Mehdi Taguema

TABLE DES MATIERES

TABLE DES ILLUSTRATIONS

TABLE DES TABLEAUX

INTRODUCTION

Pour notre client, le service Marketing de la société Ausy, nous développons une station météorologique contenant diverses fonctionnalités (date, icône météo, Pression mesurée, Température mesurée, Humidité mesurée). Ce projet s'inscrit dans le cadre de la validation d'une formation en informatique financée par Pôle emploi et la société Ausy. Il est réalisé en binôme sur 13 jours ouvrés.

A. PRESENTATION DE L'EQUIPE

L'équipe comprend deux personnes : Alix DUMAY et Mehdi TAGUEMA, toutes deux issues d'une formation en C/C++ linux embarqué en POE (Préparation Opérationnelle à l'Emploi) qui s'est déroulée sur 57 jours et réalisée par l'organisme AJC formation.
La formation comprend divers items autour de la programmation avec entre autres une présentation d'Unix/Linux afin de maitriser l'environnement, comprenant des explications sur l'administration, l'architecture de l'OS et des systèmes de fichiers sous Unix/Linux ; mais également de la programmation en langage C (sur Eclipse, et Visual Studio Code), en Python, en C++ (sous Visual Studio Code et QT Creator), en SQL (avec SQLite). L'apprentissage de ces différents langages a été validé par un projet personnel de développement d'un jeu (dans notre binôme nous avons codé un Puissance 4® et un 1000 bornes®). Afin de pouvoir être opérationnels, nous avons également eu une session sur les méthodes Agile et plus particulièrement Scrum ainsi que sur Git (un logiciel de gestion de version), nous permettant de développer un travail plus collaboratif.
L'objectif de ce projet est donc d'utiliser ce savoir acquis pour développer une application en environnement embarqué.

B. ANALYSE DES ATTENTES DU CLIENT

Afin de produire une application de qualité, il est nécessaire de faire un travail préliminaire de définition des besoins et attentes de notre client concernant sa nouvelle application météo.

Pour son application météo, le client souhaite que ce soit une application embarquée, liée à un capteur BME280 branché sur un Raspberry Pi 3B+.

Comme besoins initiaux notre client souhaite une station météo, avec comme source d'information la pression, la température et l'humidité, qui affiche les 3 valeurs mesurées, une icône représentant la prévision météo, la tendance de la pression et par conséquent celle de la météo du climat, l'heure et la date.

L'objectif de notre projet sera donc de fournir une interface météo fonctionnelle et agréable, qui utilise comme donnée d'entrée (pour l'analyse prévisionnelle de la météo) la Pression mesurée par une sonde BME280.

Besoins fonctionnels : accéder aux fonctionnalités de l'application météo (météo, Température, Humidité, Pression, Tendance) facilement depuis un ordinateur.

Pour réaliser ce projet, un environnement fixé a été décidé avec le client :

- L'utilisation d'un Raspberry Pi 3b+ avec un capteur Sensor BME280
- Date de rendu fixe au 02/05/2019
- Equipe de 2 personnes

De cette spécification, il en est déduit trois axes de travail principal pour le projet :

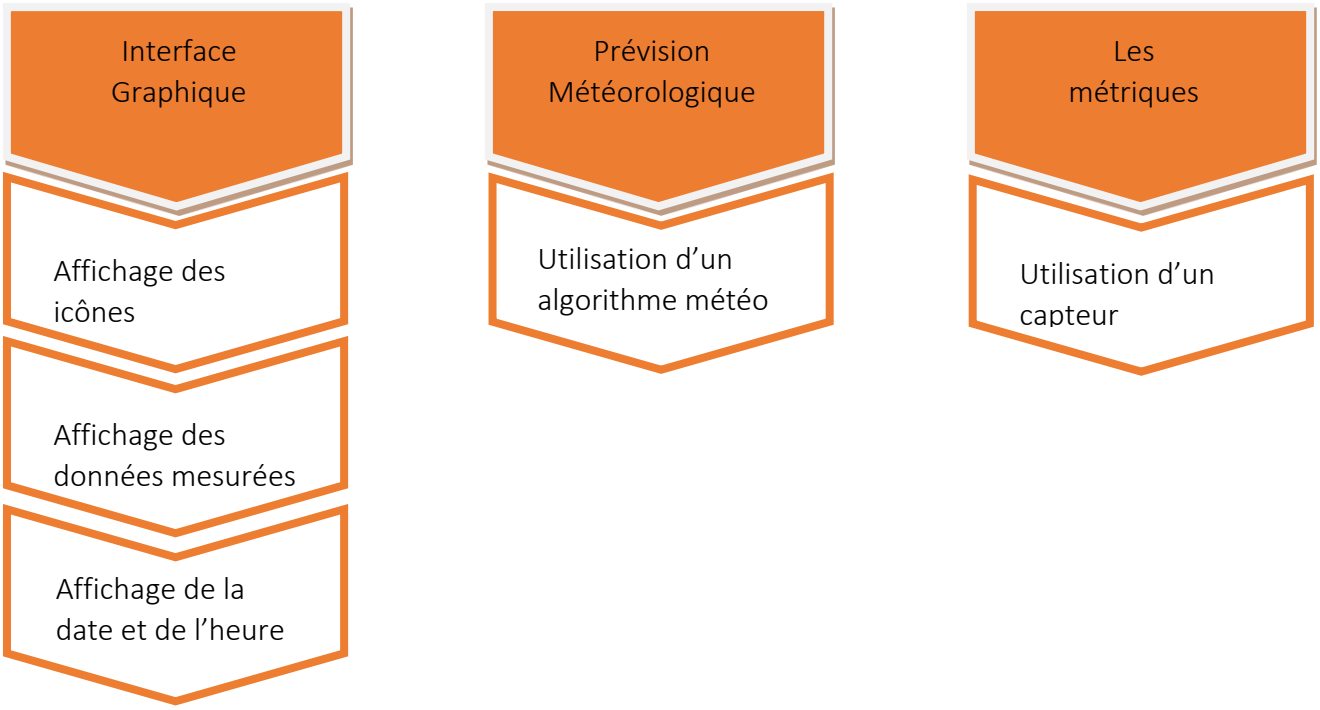

FIGURE 1 : SCHEMA DES BESOINS INITIAUX

C. MOYENS MIS EN ŒUVRE

1) La méthode Agile/Scrum

Il a été décidé d'utiliser une méthode Agile/Scrum pour réaliser ce projet. La méthodologie de gestion de projet Agile se base sur un découpage du projet en sprints successifs ou simultanés. Cette méthode permet de réaliser des livrables plus réguliers à présenter au client, elle permet ainsi de pouvoir recibler ses besoins au fur et à mesure de l'avancée du projet.
Dans la méthode Agile/Scrum classique, les sprints (phase du projet) durent au minimum deux semaines et l'équipe doit être composée d'au moins trois personnes. Au vu des contraintes initiales du projet, le Scrum a été adapté en réalisant des sprints compris entre 3 et 5 jours. Des daily meeting (quotidien comme son nom l'indique) ont été organisés afin de se concerter, d'échanger sur les progrès réalisés et les difficultés rencontrées.

2) Le versionning sur Github

Un échange d'information régulier a été mis en place, avec la création d'un répertoire sur Github. La mise en place de cette plateforme permet de travailler sur différentes tâches simultanément et regrouper chaque partie d'assembler le tout pour créer un projet complet.

Le répertoire Git du projet Weather Pi est disponible à l'adresse : https://github.com/mehditag/Meteo.git

3) La gestion des risques

Afin d'anticiper au maximum toute difficulté extérieure au projet les risques pouvant impacter le bon déroulement du projet ont été définis et des solutions ont été mises en place pour les éviter ou les compenser. Ils sont réunis dans le tableau ci-dessous :

TABLEAU 1: GESTION DES RISQUES

Risques	Solutions
Pannes Matériel	Git, sauvegarde régulières
Absence Collègue	Jira, Réunions régulières
Pannes Sensor	Capteur de secours

D. CONCEPT DU PROJET

Dans un souci de cohérence, un backlog de produit a été réalisé permettant de définir précisément les sprints et les actions à réaliser dans chacun d'eux en partant de la définition initiale des besoins. Ce backlog de produit a servi de trame pour réaliser ce projet en respectant la planification et les attentes du client.

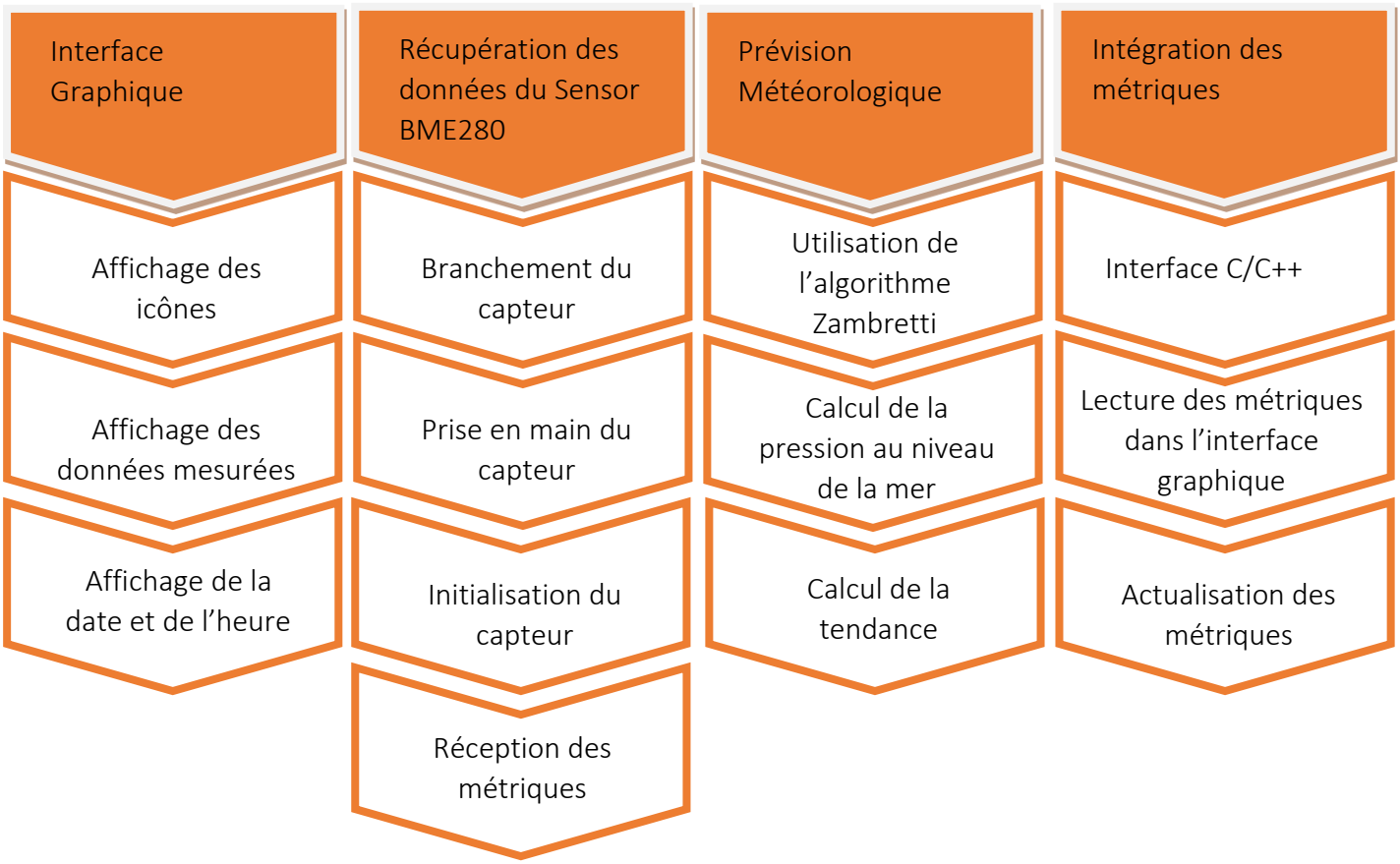

FIGURE 2: SCHEMA DU BACKLOG DE PRODUIT DEFINI D'APRES LES BESOINS INITIAUX

I. MISE EN PLACE DU MATERIEL

A. RASPBERRY PI

Un Raspberry Pi est un nano-ordinateur. Le Raspberry Pi3 Model B+ possède un processeur ARM v7 quad core 64 bits cadencé à 1,4 GHz. Afin de pouvoir l'utiliser, dans un premier temps installé un système d'exploitation Raspbian lite a été installé dessus (téléchargé directement sur le site de Rapsberry Pi), ce système a ensuite été remplacé ce système par un système Raspbian Full dans un souci de performance.
La configuration (Fuseau horaire, langue, activation de L'I2C, activation du Wifi…) a été effectué directement sur le Raspberry Pi grâce à l'outil Raspi-config.

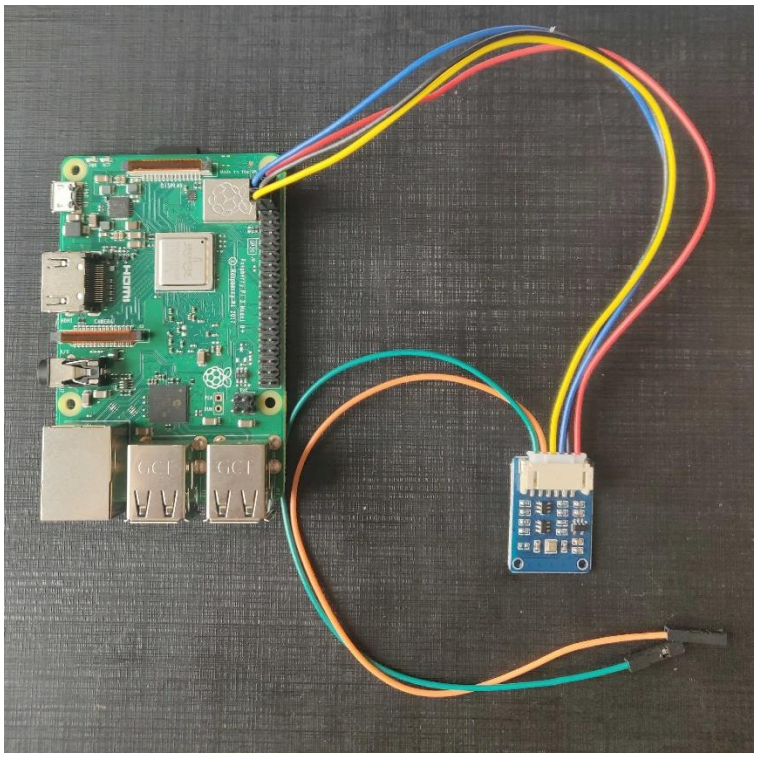

FIGURE 3: PHOTO DE RASPBERRY PI 3B+

B. LE SENSOR BME280

Le capteur fourni pour le projet est un BME280 environnemental sensor de la marque Bosch®. Il permet de récupérer en direct les données de Pression, Humidité relative, et Température. La précision de ce capteur Bosch est de ±3% pour la mesure de l'humidité (dans un intervalle de 0 à 100%) , ±1hPa en ce qui concerne la pression barométrique (entre 300hPa et 1100 hPa) et ±1°C pour la température (entre -40°C et +85°C).

FIGURE 4: PHOTO DU BME280 ENVIRONMENTAL SENSOR

C. LIAISON CAPTEUR-RASPBERRYPI

Le sensor BME280 a été branché sur le port VCC pour l'alimentation, le GND pour la Terre, L'I2C1_SDA pour le SDA (Sensor Data), l'I2C1_SCL pour l'horloge.

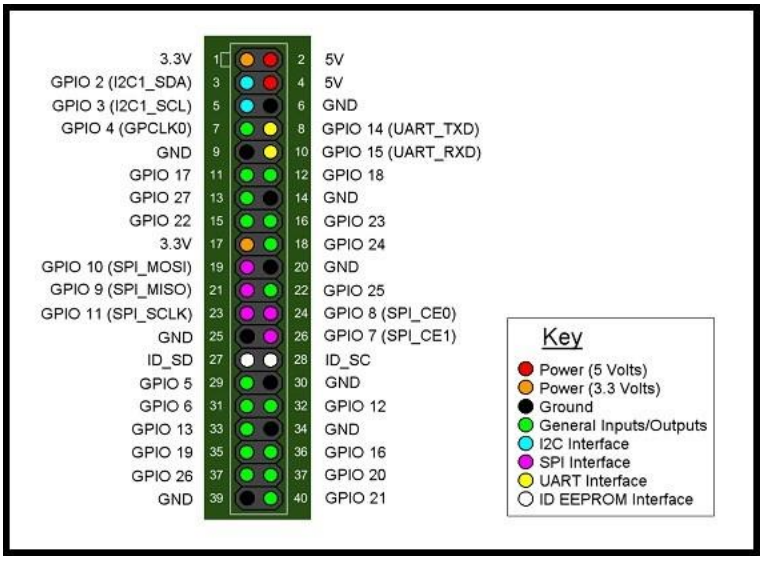

FIGURE 5: SCHEMA DES BRANCHEMENT PI

II/ RECUPERATION DES METRIQUES

La station météo nécessite des données météorologiques pour fonctionner. Pour ce faire, un capteur bme280 est utilisé. Celui-ci permet de relever des données de Température, Pression atmosphérique (essentielle pour le calcul de prévisions météo) et d'Humidité relative de l'air (T, P, H).

Le sensor bme280 est fourni avec des librairies, ou drivers (fichiers bme280.c, bme280.h et bme280_defs.h) et un programme de démo permettant d'utiliser le capteur. Ce programme permet simplement d'afficher les données recueillis par le capteur.

Afin d'intégrer ce dernier au projet, il est nécessaire d'identifier à travers le programme de démo les différentes fonctions permettant de :

- Initialiser le capteur :

Suivant le branchement du capteur au Raspberry Pi (I2C ou SPI) des instructions sont envoyées au capteur afin de le « démarrer » et l'initialiser permettant dans un second temps le recueil des données.

- Récupérer des métriques :

Recevoir du capteur les données de température, pression et humidité en identifiant l'unité des données, le format etc.

- Afficher des valeurs :

Mise en forme et affichage des données.

La station météo étant doté d'une interface graphique, les données récupérées seront affichées via celle-ci. La dernière fonction citée ci-dessus ne sera donc pas utilisée.
Le capteur étant branché en I2C, toutes les fonctions liées au branchement SPI ne sont pas nécessaire.

A. INITIALISATION DU CAPTEUR :

Une fonction est alors créée reprenant les différentes étapes de l'initialisation du capteur dans le programme démo :

- Tests d'ouverture et d'accès au bus I2C
- Détermination des paramètres de lecteur et d'écriture du bus I2C
- Enregistrement des données de paramétrages
- Enregistrement du mode d'utilisation du capteur (Normal ici)
-

Lors de l'initialisation du capteur, toutes les valeurs résultant de l'initialisation sont stockées, utilisées plus tard dans le projet, pour la lecture des métriques.

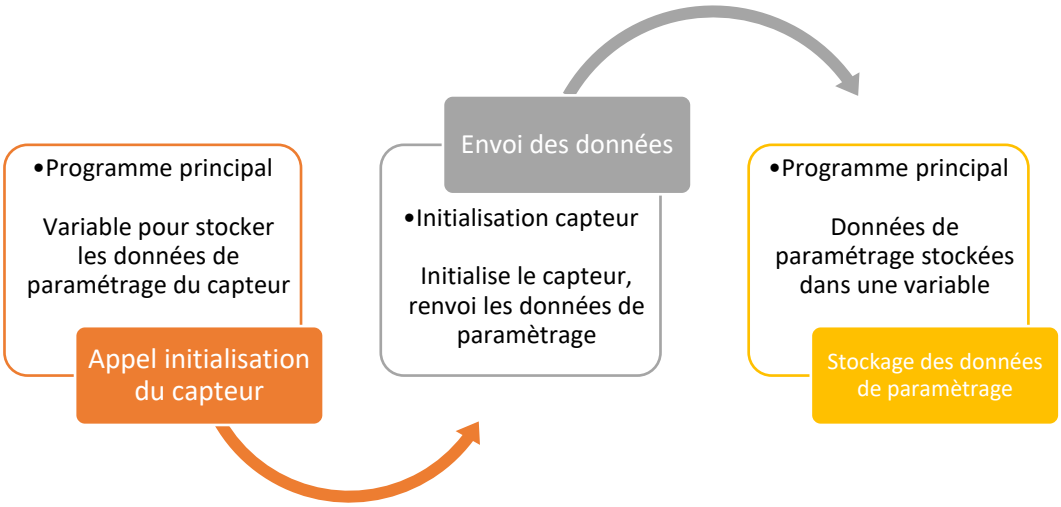

FIGURE 6: SCHEMA INITIALISATION DU CAPTEUR

B. RECEPTION DES METRIQUES

Une fois l'initialisation du capteur réalisé, les données d'initialisation et paramétrage sont connues, notamment le type de mode utilisé (Normal ici).

Une fonction pour récupérer les métriques est créée. Elle permet, une fois appelée, d'utiliser une des propriétés du capteur : la lecture des métriques. Cette fonction a besoin des données d'initialisation et de paramétrage obtenus plus tôt. Elle renvoi une structure de type bme280_data contenant les métriques mesurées lors de l'appel de la fonction.

Le programme principal est donc doté de variables permettant de stocker les métriques lues par le capteur.

Il sera donc aisé, plus tard, d'utiliser dans l'interface graphique ces métriques séparément.

III/ INTEGRATION DU CAPTEUR AU PROJET

A. INTERFACE C/C++ :

Les fonctions d'initialisation et de réception des métriques sont écrites dans un fichier capteur.c ajouté au projet. Cependant, ces fonctions, et toutes les propriétés du capteur, sont codés en C. Or, afin d'ajouter des fonctions à l'interface graphique, une classe, codé en C++ y est liée. Il est donc nécessaire de réaliser une interface C/C++ :

```
#pragma once
#include "bme280.h"
#include "bme280_defs.h"


#if defined (__cplusplus)
extern "C"{
#endif

//Fonctions du fichier capteur.c

#if defined (__cplusplus)
}
#endif
```

FIGURE 7:INTERFACE C/C++

Dans le fichier capteur.h, les lignes #if defined (__cplusplus) seront lues lors de la compilation par le préprocesseur. Ainsi, si le programme appelant les fonctions déclarées dans capteur.h est codé en C++, le préprocesseur « verra » celles-ci comme étant codées en C grâce à la commande extern « C » {…}.

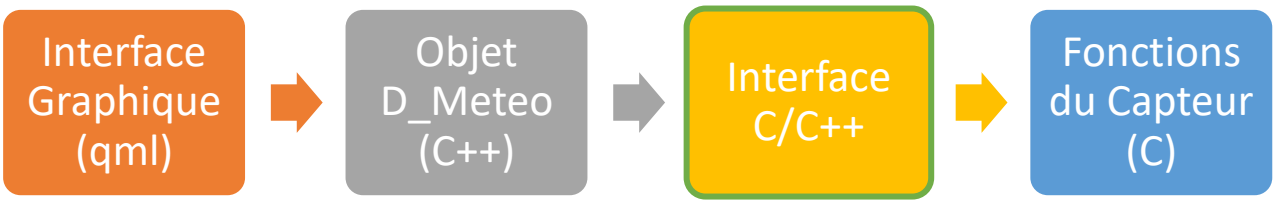

FIGURE 8: SCHEMA STRUCTURE DU PROJET

Cette interface mise en place, le programme principal est donc capable d'utiliser les fonctions d'initialisation et de récupération des métriques :

- Capt_init() :

  Cette méthode permet d'affecter le résultat de l'initialisation du capteur à une variable du programme principal. Elle est appelée lors de la construction de la classe, donc à l'ouverture du programme. Ainsi, il ne sera plus nécessaire de l'initialiser pour lire les métriques.

- Refresh() :

  Cette méthode permet d'affecter le résultat de l'acquisition des métriques à une variable du programme principal, et de mettre à jour toutes les données du programme.

- Maj_temp(), maj_hum(), maj_press() :

  Méthodes permettant d'affecter la métrique correspondant à l'attribut du programme principal.

B. LECTURE DES METRIQUES DANS L'INTERFACE GRAPHIQUE

Une fois le capteur appelé pour la lecture des métriques, les métriques sont stockées dans une variable.
Chaque métrique est stockée séparément dans un attribut correspondant.
Ces trois attributs (température, humidité et pression) ayant des fonctions d'accession correspondantes (« Get ») et utilisable par l'interface graphique via des Q_PROPERTY, il est donc possible de récupérer la valeur de Température, Humidité et Pression et l'affiché dans un champ de texte.

C. ACTUALISATION DES METRIQUES

Un timer est programmé dans l'interface graphique. Tous les t millisecondes, ce timer s'active et appel une fonction update. Cette dernière permet :

- D'appeler la fonction de récupération des métriques (Refresh ()), qui réalise dans l'ordre de :
  - Récupérer les métriques
  - Réaliser les calculs météorologiques
  - Stocker les métriques
  - Mettre à jour les attributs du programme

- D'affecter au champ de texte correspondant la valeur des attributs m_temperature, m_pressure et m_humidity grâce à leurs accesseurs.

L'intégration du capteur au projet puis l'utilisation des métriques dans l'interface suit donc ce schéma :

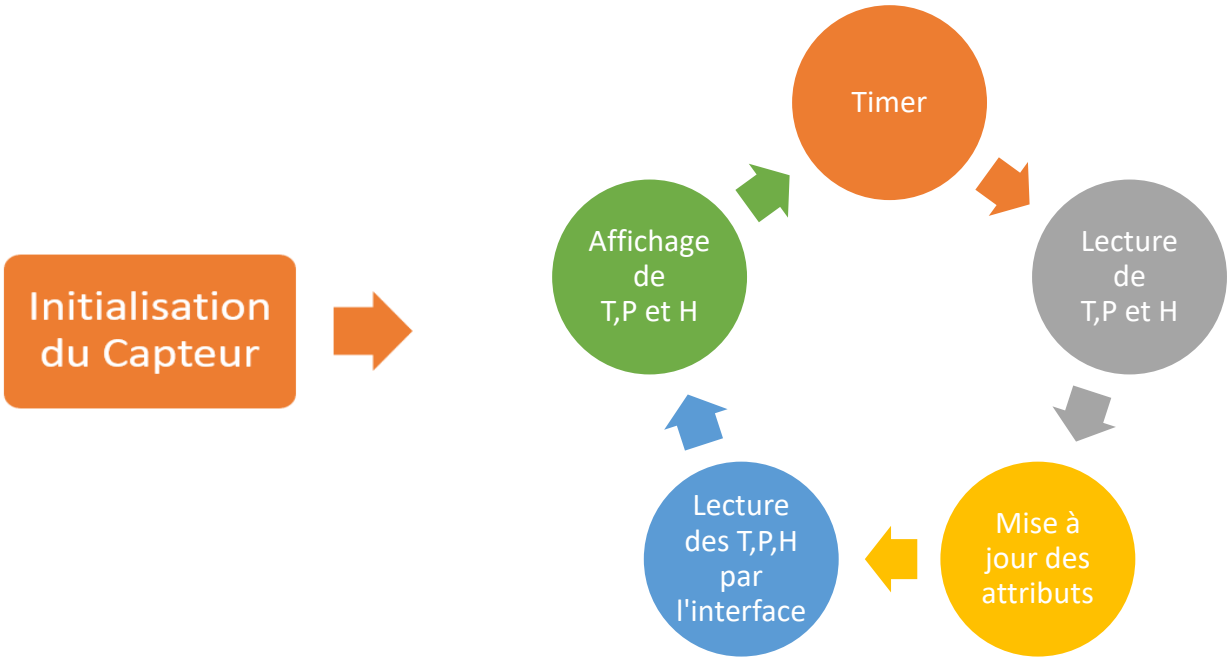

FIGURE 9: SCHEMA INTEGRATION DU CAPTEUR AU PROJET

IV/ CALCUL DES PREVISIONS METEOROLOGIQUES

A. ALGORITHME DE ZAMBRETTI

L'algorithme de Zambretti est un algorithme météo permettant de déterminer le temps en fonction de la Pression atmosphérique au niveau de la mer, du vent dominant et des tendances passées. Ici l'information concernant la direction des vents n'est pas disponible, le calcul final sera donc une approximation du calcul initial.

Les formules utilisées pour ce projet se basent sur une régression linéaire calculée à partir du cadran développé par Negretti et Zambra (1915). Ce Cadran permet de déterminer la prévision météorologique pour la Grande-Bretagne et à partir de la pression au niveau de la mer, des vents et de la tendance.

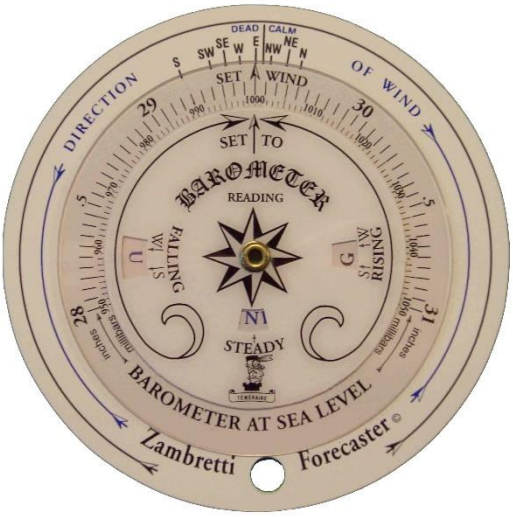

FIGURE 10: PHOTO DU PREDICTEUR METEOROLOGIQUE DE ZAMBRETTI

La tendance est considérée à la baisse si la différence entre la pression actuelle et la pression moyenne mesurée sur la dernière heure est inférieure à -2hPa.
Le coefficient de zambretti est alors calculé ainsi :

$$Z = 130 - 10 \times {}^{P}/_{81}$$

La tendance est considérée stable si la différence entre la pression actuelle et la pression moyenne mesurée sur la dernière heure est comprise entre 2hPa et -2hPa.
Le coefficient de zambretti est alors calculé ainsi :

$$Z = 147 - 50 \times {}^{P}/_{376}$$

La tendance est considérée à la hausse si la différence entre la pression actuelle et la pression moyenne mesurée sur la dernière heure est supérieure à 2hPa.
Le coefficient de Zambretti est alors calculé ainsi :

$$Z = 179 - 20 \times {}^{P}/_{81}$$

Avec P la pression au niveau de la mer, en hectoPascale.
Afin de compenser les basses pressions hivernales, le coefficient de Zambretti est majoré de 1 d'Octobre à Mars (inclus).

Le coefficient de Zambretti permet de faire une correspondance dans le tableau ci-dessous :

TABLEAU 2: RELATION COEFFICIENT DE ZAMBRETTI – METEO PREVISIONNELLE. (TABLEAU NON EXHAUSTIF)

Coefficient de Zambretti	Météo prévisionnelle	Image correspondante
1	Ensoleillé	Soleil.svg
3	Beau temps variable	Soleil_nuageux.svg
5	Averses	Averses.svg
17	Pluie fréquente	Pluvieux.svg
31	Orageux	Orageux.svg

B. CALCUL DE PRESSION AU NIVEAU DE LA MER

Dans l'algorithme de Zambretti les pressions utilisées sont les pressions au niveau de la mer. Il est donc nécessaire de convertir les données mesurées par le capteur pour pouvoir les utiliser. Dans cet objectif, la formule internationale du nivellement barométrique est utilisée:

$$P_{mer} = P_{alt} \times (\frac{1 - 0{,}0065 \times Altitude}{288{,}15})^{-5{,}255}$$

Avec P_{mer} et P_{alt} les pressions au niveau de la mer et en altitude, en hPa

Une altitude fixe a été choisi de 151m qui correspond à l'altitude moyenne du boulevard Koening (143m) majorée de 8 mètres pour les étages.

C. VERIFICATION DE L'ALGORITHME DE ZAMBRETTI

Avant d'ajouter l'algorithme de Zambretti au code principal, ce dernier a été testé dans un premier temps sous Microsoft Excel, afin de voir quelles variables sont nécessaire pour son calcul. Une fois cette vérification effectuée, l'algorithme a été codé en C++, dans une boucle afin de tester l'intégralité des valeurs possiblement mesurées par le capteur et leur calcul via l'algorithme.

V/ INTERFACE GRAPHIQUE

L'interface graphique a été réalisée en qml sous kdevelop à partir d'un projet QtQuick.
La fenêtre principale a par défaut une largeur de 600 pixels et une hauteur de 480 pixels. Afin de pouvoir organiser la fenêtre de manière modulaire, deux rectangles ont été dessinés : le premier contenant les informations en temps réel et le second présentant l'historique des données.
Dans chaque partie sont affichés différents items : du texte (variable ou non), et des images. Chacun étant positionné en fonction de la taille du rectangle 'parent ' afin de pouvoir agrandir et rétrécir la fenêtre sans modifier les positionnements.

A. AFFICHAGE DES VALEURS FIXES

Les textes de pression, température, et humidité sont affichées dans des items texte.
Le positionnement de chaque texte est réalisé en fonction de la taille de la fenêtre principale afin de rendre l'application plus ajustable. Pour les mêmes raison la valeur de la taille de la police a elle aussi été fixé en fonction de la taille du rectangle parent.
Dans un souci d'uniformité le style de police est fixé pour chaque texte en 'Linux Biolinum'.

B. AFFICHAGE DES VALEURS VARIABLES

Afin de faire varier les données affichées en temps réel (par exemple la pression, l'humidité, l'heure…), une fonction update a été intégrée au qml. Cette fonction se réinitialise toutes les secondes afin d'afficher des données actualisées. Ainsi les données récupérées par le capteur se rafraichissent en directe.
La date et l'heure sont également implémentées dans cette fonction update afin d'obtenir un affichage dynamique. La date et l'heure sont affichées à l'aide de la fonction Date de Qt sous la forme 'jour mois année' (ex : jeudi 2 Mai 2019) pour la date et 'heure:minutes:secondes' pour les heures.

C. AFFICHAGE DES ICONES

Les icônes météo doivent se mettre à jour à chaque nouveau calcul de Zambretti, dans notre cas il se calcule chaque heure. Une précision supérieure est non nécessaire car l'algorithme permet d'obtenir une prévision sur les quatre prochaines heures. Les icônes ont été téléchargées sur flaticon.com, elles sont libres de droit et en svg afin d'être ajustables dans l'interface, ainsi lorsque la taille de la fenêtre d'affichage est modifiée, elles s'agrandissent (ou se rétrécissent) en conséquence.

L’interface graphique présentée ci-dessous a été développée avec toutes ces données.

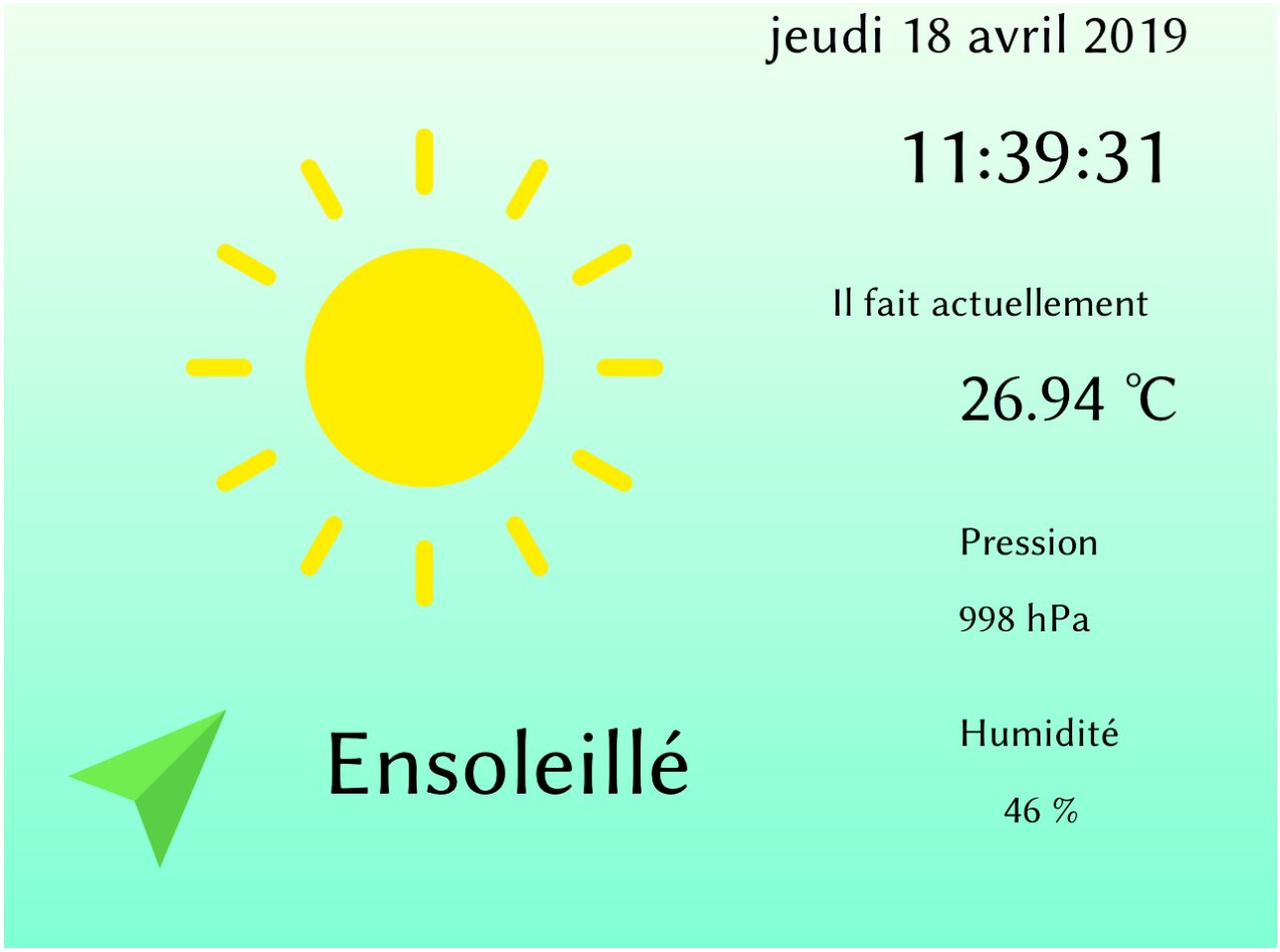

FIGURE 11: CAPTURE D'ECRAN DE L'INTERFACE GRAPHIQUE SANS LES FEATURES

VI/ FONCTIONNALITES

L'application ainsi construite présente diverses fonctionnalités afin de satisfaire l'utilisateur.

A. MODIFICATION DE L'INTERFACE JOUR/NUIT

Afin d'améliorer le confort de l'utilisateur, l'interface change de couleur en fonction de l'heure. Ainsi, de Mai à Septembre l'interface s'éclaircit à partir de 7h et s'obscurcit à 21h, et d'Octobre à Avril l'interface s'éclaircit à 9h et s'obscurcit à 19h.

B. AFFICHAGE DE LA PHASE LUNAIRE

L'affichage de la phase lunaire est une information utile pour certains utilisateur (ex : jardinier…), il a donc été décidé d'afficher une icône donnant cette information.
Dans ce but, nous avons choisi une date fixe (le 19/04/2019) où la lune était pleine, puis la date du jour est comparée à cette dernière grâce à un modulo, en sachant que la période synodique de la Lune a une durée moyenne de 29,53 jours calculé selon la formule découverte par Copernic :

$$Période_{lunaire} = \frac{1}{\frac{1}{27{,}322} - \frac{1}{365{,}25}}$$

Où 27,322 correspond à la période orbitale de la Lune et 365,25 celle de la Terre.

C. STOCKAGE DES METRIQUES POUR L'AFFICHAGE DE L'HISTORIQUE.

Avec la Weather Pi, il est possible d'afficher l'historique des valeurs mesurées sur les 4 dernières heures. L'application affiche les données Température, Pression, Humidité et icône météo enregistrées à l'heure précédente.

L'interface graphique a été modifiée pour prendre en compte ces nouvelles données, la version finales est visible ci-dessous (le mode nuit en figure 12 et le mode jour en figure 13).

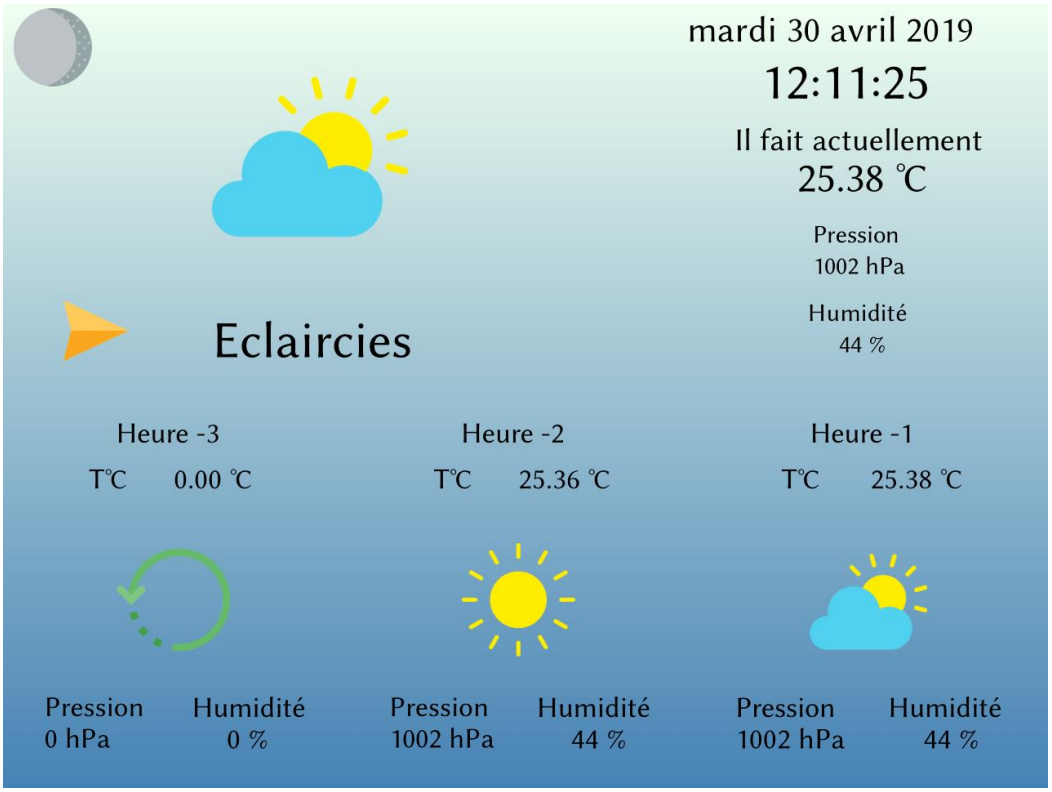

FIGURE 12: CAPTURE D'ECRAN DE L'INTERFACE GRAPHIQUE FINALE (EN MODE NUIT)

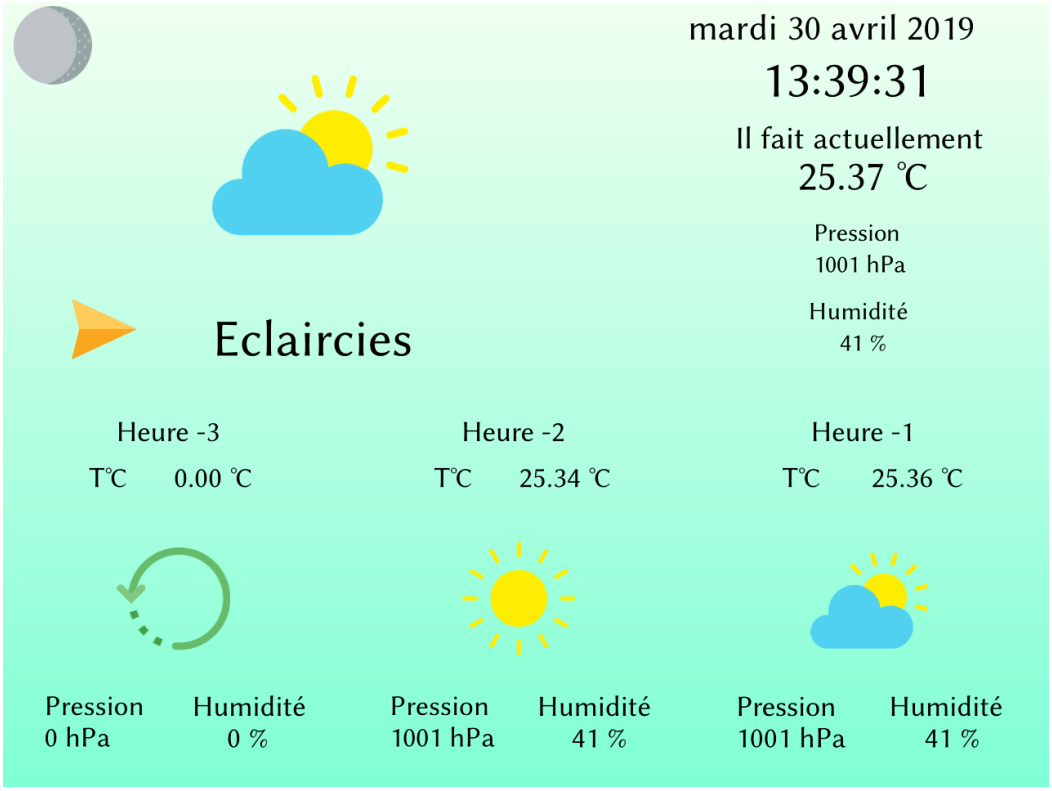

FIGURE 13: CAPTURE D'ECRAN DE L'INTERFACE GRAPHIQUE FINALE (EN MODE JOUR)

CONCLUSION

La Weather Pi est fonctionnelle, elle permet d'afficher l'ensemble des demandes initiales du client auxquelles nous avons ajouté quelques options pour les besoins de l'exercice (à savoir : appliquer les différents langages et méthodes appris au cours de la formation).
L'application fonctionne sur un Raspberry Pi 3b+, elle s'initialise automatiquement, et récolte des informations toutes les secondes. Elle possède un temps de primo-chargement d'une heure fixée afin de permettre à l'application de récupérer assez de données pour afficher une valeur fiable.
La Weather Pi possède des fonctionnalités supplémentaires telles que la variation de la couleur du fond d'écran en fonction de l'heure (mode jour/nuit), l'affichage de la phase de la Lune et l'affichage d'un historique des données (sur les quatre dernières heures).
L'interface a été présenté au client (dans ce cas nous avons demandé à notre référent projet : Sebastien HUSS) lors d'une réunion et ce dernier l'a validé.

En ce qui concerne l'organisation de notre travail, nous avons respecté le diagramme de Gant que nous avions élaboré en phase d'analyse. Nous avons respecté les règles du Scrum telles qu'elles nous ont été présentée (daily meeting, kick-off, réunion de fin de sprint…) et cela nous a permis une plus grande collaboration grâce à des échanges réguliers. La mise en place du GIT, nous a permis (outre la sauvegarde supplémentaire du projet) de pouvoir travailler sur différentes parties du projet en parrallèle en limitant l'impact sur le travail de l'autre.

Ce projet nous a permis d'appliquer une majorité des concepts appris au cours de ces trois mois de formation, car il intègre du langage C (pour la récupération des métriques), du langage C++ (pour le calcul de l'algorithme de Zambretti, des tendances et de l'historique), et du qml (pour la réalisation de l'interface graphique).

ANNEXES

Tableau 1 : Relation Coefficient de Zambretti – Météo prévisionnelle.

Coefficient de Zambretti	Météo prévisionnelle	Image correspondante
1	Ensoleillé	Soleil.svg
2	Beau temps	Soleil.svg
3	Beau temps variable	Soleil_nuageux.svg
4	Beau temps pluie à venir	Soleil_nuageux.svg
5	Averses	Averses.svg
6	Instable – Pluie à venir	Pluvieux.svg
7	Pluie – temps en dégradation	Pluvieux.svg
8	Pluie – temps très instable	Pluvieux.svg
9	Temps très instable, Pluie	Pluvieux.svg
10	Beau temps	Soleil.svg
11	Beau temps	Soleil.svg
12	Beau temps, rares averses	Soleil_nuageux.svg
13	Beau temps, possibles averses	Soleil_nuageux.svg
14	Averses intermittentes	Averses.svg
15	Variable, quelques averses	Averses.svg
16	Variable, pluie	Pluvieux.svg
17	Pluie fréquente	Pluvieux.svg
18	Pluie	Pluvieux.svg
19	Orageux, beaucoup de pluie	Orageux.svg
20	Beau temps	Soleil.svg
21	Beau temps	Soleil.svg
22	En amélioration	Soleil_nuageux.svg
23	Plutôt beau, en amélioration	Soleil_nuageux.svg
24	Plutôt beau, possible pluie	Soleil_nuageux.svg
25	Pluie, en amélioration	Pluvieux.svg
26	Temps instable	Nuageux.svg
27	Eclaircies	Soleil_nuageux.svg
28	Temps instable	Nuageux.svg
29	Rares averses	Averses.svg
30	Temps instable	Averses.svg
31	Orageux	Orageux.svg
32	Tempête	Orageux.svg