

Université de Maroua  
\*\*\*\*  
École Nationale Supérieure  
Polytechnique de Maroua  
\*\*\*\*  
Département d'Informatique et  
des Télécommunications  
\*\*\*\*



The University of Maroua  
\*\*\*\*  
National Advanced School  
Engineering of Maroua  
\*\*\*\*  
Department of Computer Science  
and Telecommunications  
\*\*\*\*

## INFORMATIQUE ET TÉLÉCOMMUNICATIONS

# SURVEILLANCE ENVIRONNEMENTALE A PARTIR DES SYSTEMES EMBARQUES ET ANALYSE DE DONNEES

Mémoire présenté et soutenu en vue de l'obtention du Diplôme d'INGÉNIEUR DE CONCEPTION EN INFORMATIQUE ET TÉLÉCOMMUNICATION OPTION GÉNIE LOGICIEL

par

**FEGUE EMBOLO ARMEL VIANNEY**

Ingénieur des travaux en Informatique de Gestion / Analyste Programmeur

**Matricule : 13Z094S**

sous la direction de : **Dr. WANSOUWE WANBITCHING**

Chargé de Cours

Devant le jury composé de :

**Président : Prof. MOTAPON OUSMANOU**

**Rapporteur : Dr. WANSOUWE WANBITCHING**

**Examinateur : Dr. HAYATOU OUMAROU**

Année Académique 2017 / 2018

**DEDICACE**

**A LA FAMILLE EMBOLO**

# REMERCIEMENTS

Au terme de ce travail, nous tenons à remercier très sincèrement :

- **Prof. MOTAPON OUSMANOU** pour m'avoir fait l'honneur d'accepter de présider ce travail.
- **Dr HAYATOU** qui m'a fait l'honneur d'accepter d'examiner ce travail. Je le remercie fortement pour toutes les remarques et conseils apportés lors de ma soutenance.
- **Dr. WANSOUWE WANBITCHING** pour son apport dans finalisation du travail ainsi que pour sa grande disponibilité;
- Toute ma famille, en particulier mon papa **EMBOLO NOAH**, ma maman **MBAZOA CLAUTILDE**, de continuer à m'apporter leur soutien dans mes choix scolaires et professionnels;
- **Dr. STINCKWISH SERGE, Dr. HAYATOU** avoir encadré et suivi l'évolution de mon travail au sein de l'IRD.
- **FOUDA ROSARIO (Technicien principal d'analyse médicale)** d'avoir pris sur lui de me présenter les méthodes préparation et d'évaluation de qualité des prélèvements dans le domaine médicale.
- Les enseignants de l'Ecole Supérieure Polytechnique de Maroua et en particulier ceux du département Informatique et Télécommunication pour leurs enseignements tout au long de notre formation ;
- Tous mes amis et camarades pour leur soutien.

# Table des matières

LISTE DES ILLUSTRATIONS	VIII
LISTE DES TABLEAUX	X
LISTE DES EQUATIONS ET FORMULES	XII
LISTE DES CODES	XIII
RESUME	XIV
ABSTRACT	XV
INTRODUCTION GÉNÉRALE	1
CHAPITRE I : CONTEXTE ET PROBLEMATIQUE	2
Introduction	2
I.     Présentation de l'entreprise d'accueil : IRD France	2
1- Historique	2
2- Organigramme	3
3- Départements	4
4- Présence dans le monde	5
II.    CONTEXTE	6
III.   PROBLEMATIQUE	7
IV.    OBJECTIFS	8
CHAPITRE II : GÉNÉRALITÉS SUR SDA	9
Introduction	9
I.     NOTION SUR LES CAPTEURS	9
1- Classification des capteurs	9
2- Caractéristique des capteurs	10
3- Type de capteurs	10
II.    NOTIONS SUR LE RASPBERRY	11
1- Spécifications générales Raspberry Pi	11
2- Modèle de Raspberry Pi	12
3- Pins du Raspberry Pi	13
III.   NOTION DE BASE DE L'ÉLECTROTECHNIQUE	16
1.   Loi fondamentales	18
2.   Composants de base	19
IV.    INTRODUCTION A PHARO	21
1.   Les caractéristiques de Pharo	21
2.   Les présentations de Pharo	22
V.     GÉNÉRALITÉS SUR LA MODÉLISATION MATHÉMATIQUE	23
1.   Variables aléatoires et agrégats sur la moyenne	24

2.	Validation des échantillons	25
3.	Détermination du modèle	28
4.	Validation du modèle	31
<b>CHAPITRE III : MODÉLISATION ET CONCEPTION</b>		<b>33</b>
Introduction		33
I.	PRÉSENTATION DE SDA	33
1.	Réseau de capteurs	34
2.	Raspberry et Serveur	34
3.	Serveur ThinkSpeak	34
4.	Serveur SDA	35
II.	ANALYSE DES BESOINS	35
1.	Les besoins fonctionnels	35
2.	Les besoins non fonctionnels	47
III.	MÉTHODE DE TRAVAIL ET TECHNOLOGIE UTILISÉES	49
1.	Méthode de travail	49
2.	Les technologies utilisées	51
3.	La Méthodologie de codage	53
IV.	MODÉLISATION	55
1.	Quelques définitions	55
2.	Modélisation de l'infrastructure	56
3.	Modélisation sur le Raspberry	66
4.	Modélisation sur ThinkSpeak	67
5.	Modélisation de l'application mobile	58
6.	Modélisation de l'application web SDA	59
V.	IMPLEMENTATION	67
1.	Implémentation du Réseau de capteurs	67
2.	Implémentation du Raspberry	72
3.	Configuration sur ThinkSpeak	78
4.	Implémentation de l'application Mobile	79
5.	Implémentation de la plateforme Web	81
<b>CHAPITRE IV : RESULTATS OBTENUS</b>		<b>85</b>
I.	RESULTATS SUR L'APPLICATION MOBILE	85
1.	Initialisation de l'application et Accueil	85
2.	Liste des champs et des alertes	86
3.	Configuration de l'application	86
1.	Détail d'une source et push de données	87
II.	RESULTATS SUR L'APPLICATION WEB	88

1.	L'accueil de la plateforme web SDA	88
2.	Information du propriétaire de l'image SDA	89
3.	Le module laboratoire SDA	90
4.	Le module Capteur SDA	93
5.	Le module Alerte SDA	96
6.	Le module Analyse SDA	98
	CONCLUSION ET PERSPECTIVES	104
	BIBLIOGRAPHIE	105
	INDEX	106

# **LISTE DES ABREVIATIONS**

API: Applications Programming Interface;

BD: Base de données ;

BTS: Base Transmission System;

CSS: Cascading Style Sheets;

CVS: Concurrent Versions System ;

DAJ : Direction des affaires juridiques ;

DCPI : Direction de la communication et du partage de l'information ;

DDUNI : Direction pour le développement des usages numériques innovants

DF : Direction des Finances ;

DISCO : Département Dynamiques Internes et de Surface des Continents ;

DMRID : Département Mobilisation de la recherche et de l'innovation pour le développement ;

DRH : Direction des Ressources Humaines ;

DRIE : Département Relations internationales et européennes ;

DRIF : Délégation régionale Ile-de-France ;

DROc : Délégation régionale Occitanie ;

DROu : Délégation régionale Ouest ;

DRSE : Délégation régionale Sud-Est ;

ECOBIO : Département Écologie, Biodiversité et Fonctionnement des Ecosystèmes Continentaux ;

EDI: Environnement de Développement Intégré;

GPIO : Général Purpose Input/Output ;

HTML: HyperText Markup Language;

IDE: Integrated Development Environment;

IP: Internet Protocol;

LED : Light emitting diode ;

MAC : Mission accompagnement au changement ;

MAS : Mission d'appui à la science ;

MCST : Mission culture scientifique et technologique ;

MEPR : Mission d'évaluation et de programmation de la recherche ;  
MIDN : Mission Infrastructures et données numériques ;  
MPII : Mission pour la promotion de l'interdisciplinarité et de l'intersectorialité ;  
MSST : Mission santé et sécurité au travail ;  
OCEANS : Département Océans, climat et ressources ;  
PDF: Portable Document Format;  
SAS : Département Santé et sociétés ;  
SGBD: Systèmes de Gestion de Bases de Données ;  
SOC : Département Sociétés et Mondialisation ;  
UML: Unified Modeling Langage;  
URL: Uniform Resource Locator;  
XHTML: Extensible HyperText Markup Language;  
XML: eXtensible Markup Language.

# LISTE DES ILLUSTRATIONS

Figure 1 : Logo IRD .....	2
Figure 2 : Organigramme de l'IRD .....	4
Figure 3 : Capteur infrarouge .....	9
Figure 4 : Raspberry Pi modèle 3B .....	12
Figure 5 : résistors en série .....	20
Figure 6 : résistors en parallèle .....	20
Figure 7 : Exemple d'interface de Pharo .....	23
Figure 8 : Représentation de Levey-Jennings.....	26
Figure 9 : Règles de WESTGUARD .....	27
Figure 10 : Exemple de nuage de points .....	29
Figure 11 : Architecture générale de SDA .....	33
Figure 12 : Patron MVC .....	54
Figure 13 : Diagramme de déploiement de l'infrastructure .....	57
Figure 14 : Diagramme cas d'utilisation du Raspberry.....	66
Figure 15 : Diagramme cas d'utilisation ThinkSpeak.....	67
Figure 16 : Diagramme de classe de l'application mobile.....	59
Figure 17 : Diagramme de classe de la couche Modèle SDA .....	60
Figure 18 : Diagramme de classe de la couche Controller SDA.....	61
Figure 19 : SDA Modèle Vue.....	Erreur ! Signet non défini.
Figure 20 : Diagramme de classe de la couche vue SDA.....	62
Figure 21 : Cas d'utilisation LaboratoireView SDA .....	63
Figure 22 : Cas d'utilisation CapteurView SDA .....	64
Figure 23 : Cas d'utilisation Analyse d'échantillon SDA .....	65
Figure 24 : Diagramme cas d'utilisation Analyse des données combinatoires .....	65
Figure 25 : Réseau de capteurs .....	68
Figure 26 : Capteur d'Humidité température .....	68
Figure 27 : Branchement d'un DHT22 .....	69
Figure 28 : Préparer la récupération .....	70
Figure 29 : Capteur de luminosité .....	70
Figure 30 : Schéma électrique du capteur de luminosité .....	71
Figure 31 : Schéma électrique de notre réseau de capteurs .....	71
Figure 32 : Connexion Réseau de capteurs à le Raspberry .....	72
Figure 33 : Interface de commande de PharoThing Server à partir du Client .....	76
Figure 34 : Système de ThinkSpeak.....	78
Figure 35 : Logo Android Studio .....	79
Figure 36 : Schéma de l'application Mobile .....	80
Figure 37 : Logo SeaSide.....	81
Figure 38 : Initialisation de l'application .....	85
Figure 39 : Page d'accueil .....	85
Figure 40 : Liste des champs de la chaine .....	86
Figure 41 : Liste des Alertes Signalées .....	86
Figure 42 : Configuration du la source principale .....	87
Figure 43 : Configuration des Api et alertes.....	87
Figure 44 : Visualisation d'une source.....	88
Figure 45 : Envoie de données en ligne.....	88
Figure 46 : Page d'accueil Web SDA .....	89
Figure 47 : Page Profil utilisateur .....	90
Figure 48 : Ajout d'un laboratoire .....	90

Figure 49 : Liste des laboratoires .....	91
Figure 50 : Page détail d'un Laboratoire .....	92
Figure 51 : Ajouter un capteur à un laboratoire .....	93
Figure 52 : Ajouter un capteur .....	94
Figure 53 : Liste des capteurs .....	95
Figure 54 : Détail du capteur .....	96
Figure 55 : Ajouter un modèle d'alerte .....	97
Figure 56 : Liste des modèles d'alerte.....	98
Figure 57 : Sélectionner un capteur pour une validation .....	98
Figure 58 : Analyse des échantillons .....	99
Figure 59 : Correction due à l'interpolation .....	100
Figure 60 : Choix d'un couple de capteurs .....	101
Figure 61 : Relation entre données – modèle.....	102
Figure 62 : Relation entre les données - Tableau marginale.....	103
Figure 63 : Indépendance entre deux variables.....	103

# LISTE DES TABLEAUX

Tableau 1 : Type de capteurs.....	11
Tableau 2 : Tableau comparatifs des modèles de Raspberry .....	13
Tableau 3 : Pins modèle 3B.....	14
Tableau 4 : exemple de BreadBoard .....	15
Tableau 5 : Exemple d'utilisation sur une led .....	16
Tableau 6 : Effet du courant sur le corps humain.....	17
Tableau 7 : Schéma électrique .....	17
Tableau 8 : Illustration de la loi d'Ohm.....	18
Tableau 9 : Illustration de la loi des mailles.....	19
Tableau 10 : Illustration de la loi des mailles.....	19
Tableau 11 : Code couleur des résistors .....	20
Tableau 12 : Classification des couleurs des LEDs.....	21
Tableau 13 : Tableau marginal de la variable aléatoire X et Y .....	29
Tableau 14 : intensité d'un nuage de points.....	30
Tableau 15 : forme d'un nuage de points .....	30
Tableau 16 : sens d'un nuage de points .....	31
Tableau 17 : Connexion à sur le Raspberry.....	36
Tableau 18 : Contrôler l'alimentation des capteurs.....	36
Tableau 19 : Interpréter les données venant des capteurs .....	37
Tableau 20 : Charger du code dans le Raspberry.....	37
Tableau 21 : Importer les données .....	37
Tableau 22 : Interpréter les données venant des capteurs .....	38
Tableau 23 : Lister les laboratoires .....	38
Tableau 24 : Ajouter un laboratoire.....	39
Tableau 25 : Supprimer un laboratoire .....	39
Tableau 26 : Ajouter un capteur à un laboratoire.....	39
Tableau 27 : Supprimer un capteur à un laboratoire.....	40
Tableau 28 : Visualiser en un graphe les données de ses capteurs .....	40
Tableau 29 : Lister les Capteurs .....	40
Tableau 30 : Ajouter un capteur .....	41
Tableau 31 : Supprimer un capteur.....	41
Tableau 32 : Ajouter une signalisation d'alerte un capteur .....	41
Tableau 33 : Supprimer une signalisation d'alerte un capteur .....	42
Tableau 34 : Visualiser l'évolution des données de ses capteurs .....	42
Tableau 35 : Initialiser les données du capteur en important à partir d'une feuille de calcul .....	42
Tableau 36 : Lister les modèles d'Alertes.....	43
Tableau 37 : Ajouter un Alerte .....	43
Tableau 38 : Supprimer un Alerte .....	43
Tableau 39 : Lister les capteurs (Validation d'échantillon) .....	44
Tableau 40 : Effectuer une validation selon Levey-Jennings.....	44
Tableau 41 : Effectuer une validation selon les règles de WESTGRAD .....	44
Tableau 42 : Corriger les données par interpolation linéaire .....	45
Tableau 43 : Annuler une interpolation linéaire .....	45
Tableau 44 : Lister les couples de capteurs (Corrélation des séries) .....	45
Tableau 45 : Dresser le tableau marginal des séries.....	46
Tableau 46 : Dresser le tableau de dépendances des séries.....	46

Tableau 47 : Dresser le nuage des points pondéré correspondant aux deux séries .....	46
Tableau 48 : Dessiner les tableaux du modèle et du graphe moyen du nuage de point.....	47
Tableau 49 : Modifier le modèle .....	47
Tableau 50 : Disponibilité.....	47
Tableau 51 : Testabilité .....	48
Tableau 52 : Facilite de maintenance .....	48
Tableau 53 : Facilite de d'utilisation .....	48
Tableau 54 : Adaptation aux écrans.....	49
Tableau 55 : Format des données du DHT22 .....	69

# **LISTE DES EQUATIONS ET FORMULES**

Équation 1 : Loi d'Ohm .....	18
Équation 2 : Loi des mailles .....	19
Équation 3 : Loi des nœuds .....	19
Équation 4 : Résistance et code couleur .....	20
Équation 5 : Résistances en série .....	20
Équation 6 : Résistances en parallèle .....	20
Équation 7 : Moyenne d'une série .....	24
Équation 8 : Ecart type d'une série .....	25
Équation 9 : Probabilité $P(X=x)$ .....	28
Équation 10 : Probabilité $P(Y=y)$ .....	28
Équation 11 : Covariance de X et Y .....	31
Équation 12 : Variance de Y .....	31
Équation 13 : Variance de X .....	31
Équation 14 : Coefficient de corrélation .....	31

## **LISTE DES CODES**

Code 1 : Installation de PharoThing Server .....	75
Code 2 : Installation de PharoThing Client .....	75
Code 3 : Connexion à PharoThing Server .....	75
Code 4 : Récupération des 40bits du capteur DHT22 .....	77
Code 5 : Appel d'une Activité Android .....	80
Code 6 : Ajout des dépendances pour les Graphes Android.....	81
Code 7 : Appel d'une activité avec paramètre .....	81
Code 8 : Génération du code HTML SeaSide.....	82
Code 9 : SeaSide avec les appels Callback.....	82
Code 10 : Installation de SeaSide Stable .....	83
Code 11 : Bootstrap Pharo .....	83
Code 12 : Installation de DataFrame.....	83
Code 13 : Installation de HighChart .....	84

# RESUME

Ce travail présente la conception et la réalisation d'un prototype de surveillance environnementale au moyens des systèmes embarqués et d'analyse de ces données à partir des outils logiciels et mathématiques; ce prototype implémente donc une version de l'infrastructure SDA avec une approche plutôt automatique allant de la collecte au rendu des résultats. En effet l'intervention forte humaine dans la surveillance ralenti le processus et favorise l'apparition des erreurs ; le déploiement d'une solution automatisant le processus sera dans la plupart des cas couteux car les équipements auront un certain cout pour des travaux dont la taille n'est généralement pas proportionnelle. Suite à cette observation nous avons établi un cahier de charge qui propose une solution globale et générique à ce problème ; et pour son implémentation nous avons choisi d'utiliser pour le cas de figurer deux capteurs que nous connectons à un nano ordinateur, puis nous envoyons les données sur le Cloud, ensuite nous suivons l'évolution de ces données à partir d'une application mobile et enfin nous faisons la préparation de ces données et leurs analyses sur une plateforme web développée en Pharo.

**Mot clés :** électronique, mathématique, modélisation, Pharo, Cloud.

# **ABSTRACT**

This work presents the design of an alternative solution, to facilitate the search process through the electronic tools, software and mathematics that the SDA; with a rather automatic approach going from the collection to the results result. The strong human intervention slowed down the process and favored the appearance of errors; the deployment of a solution automating the process will be in a number of customs for the work of the equipment. Following this observation we have established a specification which proposes a global and generic solution to this problem; We have tracked the evolution of this data from a mobile application, and we track the evolution of this data from a mobile application. and finally we prepare the data and their analysis on a web platform developed in Pharo.

**Key words:** electronics, mathematics, modeling, Pharo, Cloud.

# INTRODUCTION GÉNÉRALE

Dans de nombreux domaines de recherche, nous avons besoin de prendre en compte les propriétés physiques (manière périodique) environnementales qui influencent l'évolution de l'objet d'étude. Cette collecte de données, lorsqu'elle est effectuée se fait **ici** généralement de manière manuelle ce qui impacte fortement la quantité et la fiabilité des prélèvements ; quand bien même celle-ci est automatisée, les coûts de la mise en place d'un système de collecte ne sont pas négligeables; par conséquent elle n'est pas à la portée de tous.

Les données collectées doivent ensuite être analysées pour aboutir à des conclusions utilisables dans une expérience, utilisée dans le cadre d'une étude. Les effets d'une collecte manuelle, auront un impact sur le temps d'analyse, et augmenteront considérablement le risque d'erreurs, mettant en cause la fiabilité du résultat.

**Sensor Data Analytic** est une solution qui fournit un ensemble complet d'outils qui faciliteront la collecte des données sur les différents sites d'études, le traitement et l'analyse de celles-ci en vue d'aboutir à des conclusions fiables.

Notre tâche ici consistera à apporter une solution matérielle et logiciel pour la réalisation de cette solution ; tout en mettant en avant les technologies utilisées pour la réaliser.

Pour cela nous procéderons ainsi : au **chapitre I**, nous présenterons le contexte dans lequel nous comptons œuvrer, et de là nous tirerons une problématique qui justifie l'utilité de notre solution; ensuite au **chapitre II**, nous présenterons quelques généralités sur notre sujet ainsi que les différents sujets qui y gravitent ; puis au **chapitre III**, nous présenterons les étapes de modélisation et l'implémentation de notre outils; au **chapitre IV**, nous présenterons les résultats et enfin nous présenterons la conclusions et les perspectives.

# CHAPITRE I : CONTEXTE ET PROBLEMATIQUE

## Introduction

Ce chapitre présente le contexte de l'étude de notre projet et apporte une vue générale de ce qui sera développé le long de notre travail. Une attention particulière est accordée à la présentation de l'entreprise porteuse du projet, puis celle de l'environnement ciblé par l'application et enfin ressortir la problématique.

### I. Présentation de l'entreprise d'accueil : IRD France



Figure 1 : Logo IRD

L'Institut de recherche pour le développement (IRD) est un établissement public à caractère scientifique et technologique (EPST) français sous la tutelle des ministères chargés de la Recherche et de la Coopération, en remplaçant l'Office de la recherche scientifique et technique outre-mer (ORSTOM) créé le 11 Octobre 1943.

Cet organisme participe à des recherches scientifiques et techniques par le biais d'accords signés entre la France et certains pays en développement (exemples : Cameroun, Bénin, Burkina Faso, Côte d'Ivoire, Ghana...).

Ces programmes de recherche destinés à apporter une aide au développement des pays du Sud, s'orientent autour des sciences humaines et sociales (géographie, sociologie...), des sciences de la santé (maladies infectieuses, grandes endémies, nutrition, environnement) et des sciences de la nature (hydrologie, pédologie, géophysique, ichtyologie...).

Le siège de l'IRD est situé à Marseille (à Paris jusqu'en septembre 2008). [1]

#### 1- Historique

Les principales étapes de l'histoire de l'Institut sont les suivantes :

- 1937 : création du Comité consultatif des recherches scientifiques de la France d'outre-mer et du Conseil supérieur de la recherche scientifique pour la coordination de la recherche nationale, institutions chargées de l'organisation de la "science coloniale" qui s'était déjà développée aux lendemains de la première guerre mondiale.

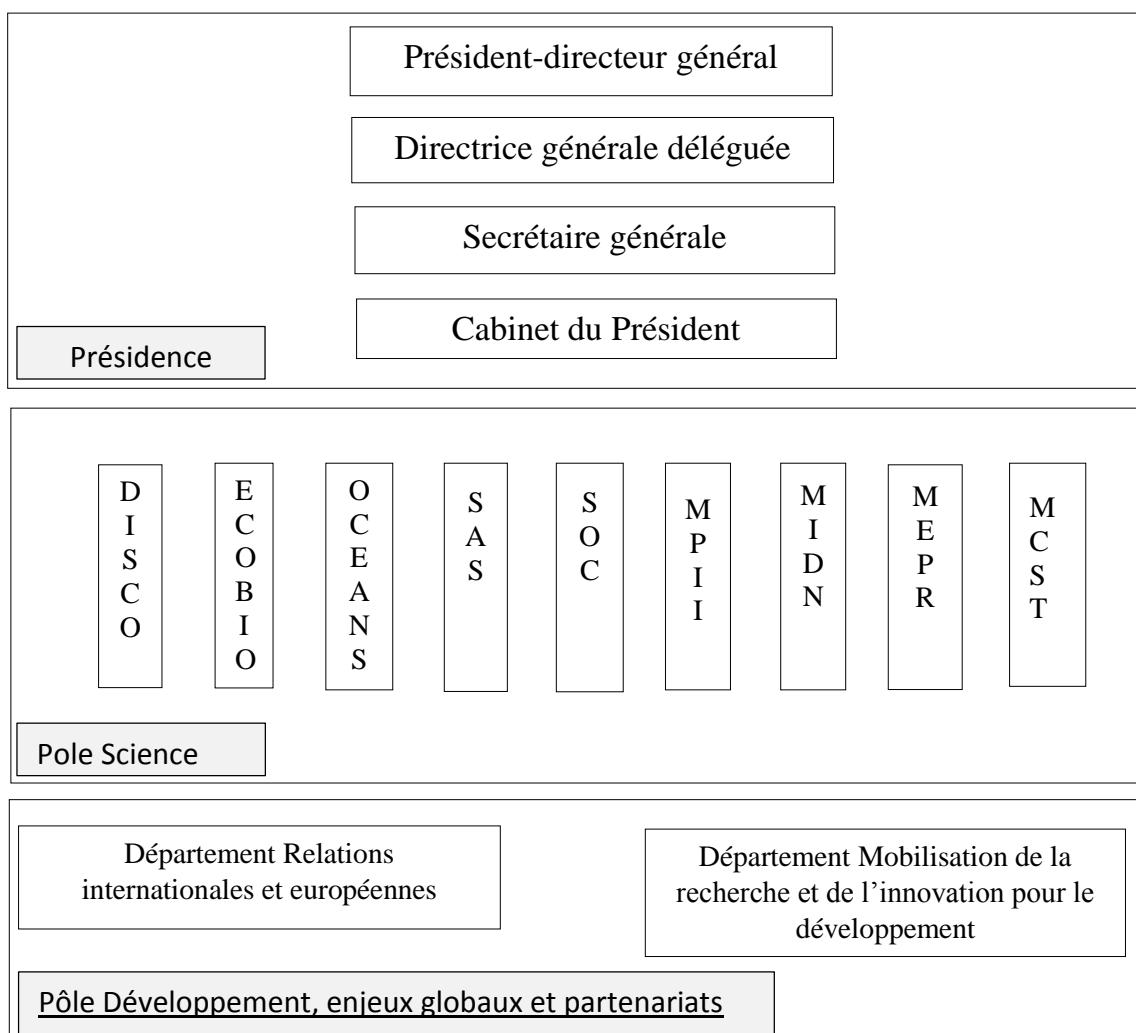
- 1944-1953 : l'Office change deux fois d'appellation, tout d'abord ORSOM (Office de recherche scientifique d'outre-mer) puis ORSTOM (Office de la recherche scientifique et technique outre-mer) qui est rattaché au ministère de la France d'outre-mer ; son conseil d'administration est présidé par le Secrétaire d'État à la recherche [2].

- Le décret du 5 novembre 1998 modifie le décret organique de juin 1984 ainsi que ses textes d'accompagnement et introduit la nouvelle appellation d'Institut de recherche pour le développement (IRD).

Depuis l'année 1998, l'IRD a vu son statut et ses missions modifiées prenant ainsi en compte l'évolution de la recherche en France et dans le monde.

## 2- Organigramme

Actuellement (juin 2018) l'IRD est hiérarchisée ainsi [3] :



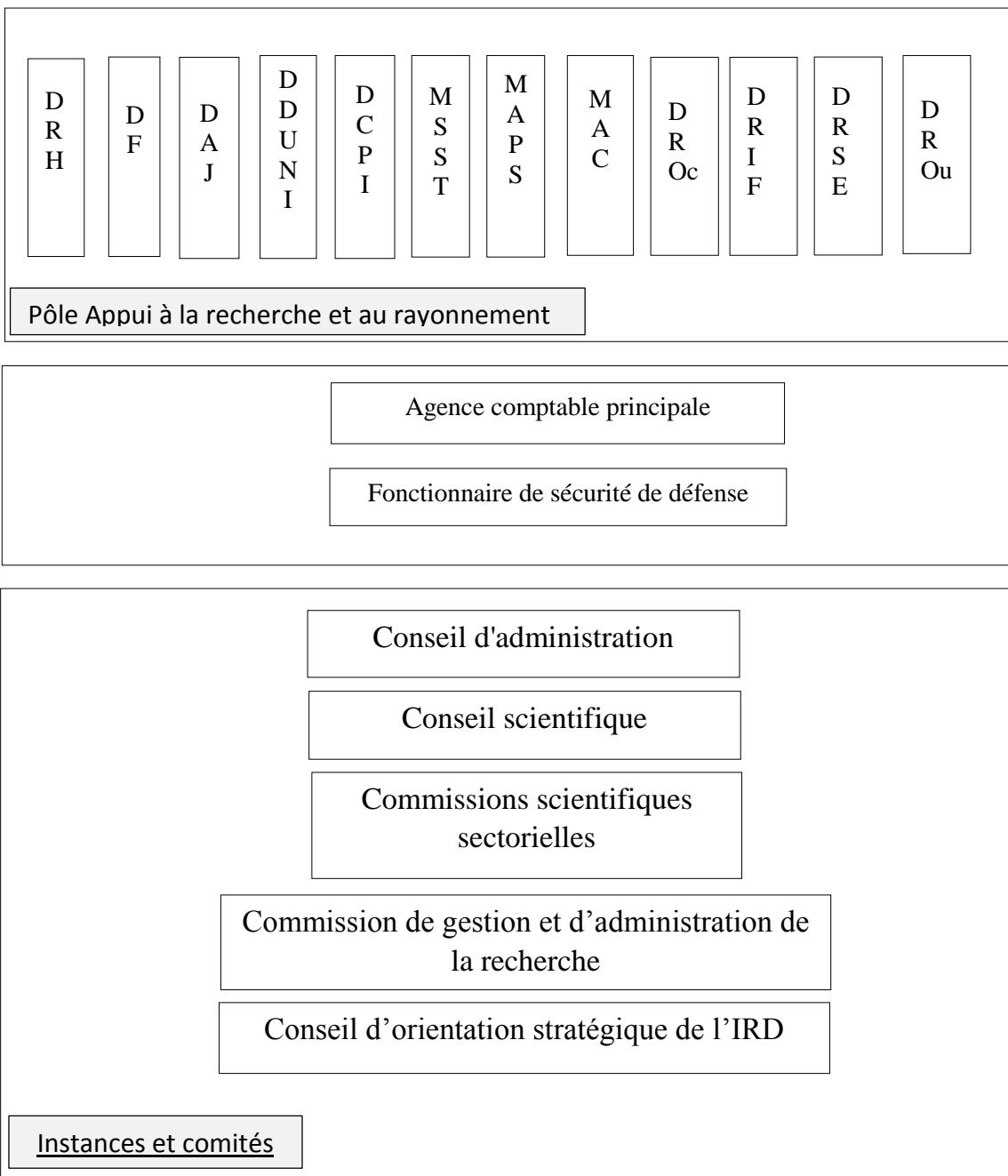


Figure 2 : Organigramme de l'IRD

Cet organigramme est divisé en plusieurs parties hiérarchiquement positionnées.

- La présidence gère décide des grandes orientations de l'Institut ;
- L'agence comptable principale gère la préparation, l'analyse et l'exécution budgétaire et finance les unités, les centres et les représentations ;
- La Direction des Ressources Humaines contribue à la définition de la politique de l'emploi de l'Institut et assure sa mise en œuvre ;
- Le pôle de développement gère scientifiques gère les programmes scientifiques, l'expertise, la valorisation et la formation des pays du Sud.

### 3- Départements

Le Pôle Science est animé par le président-directeur général. Les structures qui le composent soutiennent et mettent en œuvre la politique scientifique de l'Institut et regroupe plusieurs départements à savoir :

- Département Dynamiques Internes et de Surface des Continents (DISCO) ;
- Département Ecologie, Biodiversité et Fonctionnement des Ecosystèmes Continentaux (ECOBIO) ;
- Département Océans, climat et ressources (OCEANS) ;
- Département Santé et sociétés (SAS) ;
- Département Sociétés et Mondialisation (SOC) ;
- Mission pour la promotion de l'interdisciplinarité et de l'intersectorialité ;
- Mission Infrastructures et données numériques ;
- Mission culture scientifique et technologique [4].

#### **4- Présence dans le monde**

Outre le siège de Marseille, l'IRD est présent dans plus de 90 pays sous forme de centres ou de missions temporaires :

- Afrique subsaharienne
  - o Bénin, Burkina, Faso, Cameroun, Niger, Mali, Sénégal, Guinée, Ghana, etc...
- Afrique du Nord
  - o Égypte, Maroc, Tunisie, etc...
- Amérique latine
  - o Bolivie, Brésil, Chili, Pérou, Équateur, Mexique, Colombie, Argentine, etc...

## II. CONTEXTE

L'IRD est un institut encourageant la recherche scientifique en vue du développement ; en tant que telle, elle regroupe de nombreux projets de recherche dans des domaines variés tel que : maladies infectieuses, grandes endémies, nutrition, environnement et autres. L'IRD parraine des projets gouvernementaux des entreprises de recherche et des JEAI (Jeunes équipes associées à l'IRD).

Parmi les entreprises partenaires de l'IRD au Cameroun, on peut citer :

- Centre Pasteur du Cameroun (**CPC**) : c'est un partenariat avec le gouvernement du Cameroun et qui dispose d'une autonomie financière ;
- Centre de recherche sur les filarioSES et autres maladies tropicaLES (**CRFilMT**) ;
- Agence nationale de recherche sur le sida et les hépatites virales (**ANRS**)

Pour ce qui est des JEAI, l'IRD vise l'émergence ou le renforcement des jeunes équipes de recherche dans les pays du Sud ; On peut distinguer dans la zone de l'Afrique centre et de l'Ouest une vingtaine de groupes parmi lesquels :

- Le projet IMPALE-Cameroun : IL s'agit des Indicateurs de lutte contre le paludisme au Cameroun [5] ;
- EPIRETRYP – Cameroun : le thème de ce projet est Epidémiologie de la résistance aux trypanocide ;
- MODELCAF - Côte d'Ivoire : Modélisation de l'architecture et de la croissance du cafier ;
- RI3M – Mauritanie : Recherches Intégrées sur les Moustiques et les Maladies infectieuses en Mauritanie etc...

L'IRD aimeraient pouvoir suivre les évolutions du travail des Jeunes Equipes et leurs donner des outils qui facilitent leur processus de recherche. En effet les partenariats avec les grandes structures ont cet avantage que la logistique est déjà disponible ; ce qui n'est pas toujours le cas avec les JEAI. Pour former une équipe, il y a quelques conditions à remplir :

- Constituée d'au moins trois chercheurs du Sud,
- Implantée dans un pays du Sud,
- Associée à une unité de recherche de l'IRD,
- Sur une thématique de recherche liée aux grands enjeux sociétaux sanitaires et environnementaux actuels.

La jeune équipe va donc suivre un nombre d'étapes pour aboutir à des résultats fiables :

- **Première étape** : On part d'un problème, pour émettre une hypothèse ; le reste des étapes aura pour principale rôle de valider ou d'invalider cette hypothèse ;
- **Seconde étape** : On choisit une population à étudier, puis on prélève un échantillon de cette population (ou une collecte de données). C'est principalement sur ces données que seront portées les futures analyses ;
- **Troisième étape** : On prépare les données pour les analyse ; cette préparation vise à déterminer qualité et la précision de la collecte ;
- **Quatrième étape** : on effectue des analyses statistiques sur les données en croisant les variables entre elles et étudiant les relations qui les lie ;
- **Cinquième étape** : dans la mesure du possible on détermine le model qui permet d'expliquer le phénomène et effectuer des prévisions.

Une fois que ces étapes ont été suivies et que l'on a abouti à des résultats (validation d'une hypothèse), on va lancer une série d'actions qui vont contribuer à résoudre le problème préalablement posé.

### III. PROBLEMATIQUE

Le but d'une recherche (dans ce contexte) est de valider une hypothèse à partir des observations qui permettent de définir un modèle. Ce modèle permettra de faire des analyses et simulations pour pouvoir effectuer des prédictions sur l'objet de la recherche.

Il faut pour cela une quantité importante de données prélevées (Observations) avec une marge d'erreur très faible pour établir un modèle fiable. Il est à noter que des données erronées conduisent un l'échec certain d'une expérience ; car tout analyse faite, ne sera pas celle de l'environnement étudié et ainsi toutes prédictions qui en découlera aura une grande probabilité d'échec.

La phase d'observation terminée il faut procéder à l'analyser les données et le faire à la main (outils de Bureautique et feuilles de calcul) ou par report des données dans une plateforme comportent aussi une grande probabilité d'erreur humaine ; Sans compter la lenteur dans le traitement de ces données. Une erreur de calcul aura un impact sur les résultats (pouvant impacter les résultats d'étude en vue de l'éradication d'une épidémie).

En résumant cette situation, nous constatons un besoin évident qui se fait ressentir. Nous avons besoin d'automatiser les collectes des données lors des observations ; de faire le suivi de cette collecte ; d'automatiser les calculs sur les données observées ; Et mettre en place ce système nécessitant de petits moyens de déploiement, qui soit tout aussi facile à installer.

Nous avons donc opté à une solution générique et globale. Il s'agit de mettre en place :

- Un système sur lequel on puisse greffer des capteurs adéquats pour automatiser la collecte de données ;
- Suivre l'évolution de ces cette collecte ;
- Faciliter le processus de préparation des données ;
- Enfin faciliter l'analyse de ces données.

En apportant toutes ces valeurs ajoutées, elle doit pouvoir être facile à déployer (côté technique) et à faible cout (côté financier).

## IV. OBJECTIFS

Les objectifs de manière basique visent à faciliter le processus de recherche d'une jeune équipe qui dispose de son kit, et dont la collecte est automatisable (ce cas sera meilleur) ou non (la collecte se fera manuellement). Mais d'une manière un peu plus fonctionnelle, on peut diviser nos objectifs en deux tranches :

- L'objectif principal vise à pouvoir effectuer des opérations suivantes.
  - o Réaliser une collecte de données de manière automatique ;
  - o Contrôler le flux de collecte ;
  - o Implémenter les plates-formes de récupération de traitement et d'analyse de ces données.
- L'objectif secondaire concerne plus l'aspect pratique ; ce projet doit permettre de fournir une solution en Pharo et de disposer d'une implémentation qui sera libre, qui évoluera avec les nouvelles contributions des développeurs de la communauté.
  - o Implémenter des API (en Smalltalk dans la mesure du possible) pour la communication avec les capteurs ;
  - o Implémenter la plateforme web avec un Framework Smalltalk ;
  - o Faire évoluer la bibliothèque OpenSource DataFrame avec des nouvelles implémentations des fonctions statistiques sur des séries.

# CHAPITRE II : GÉNÉRALITÉS SUR SDA

## Introduction

SDA est une infrastructure mettant ensemble une plateforme logicielle et électronique ; permettant de recueillir des données provenant soit d'un sujet expérimental à travers les capteurs, soit à travers une source externe; et faisant sur celles-ci une modalisation statistique. Pour comprendre les contours de ce travail, il est nécessaire d'avoir des notions assez générale sur les notions comme les capteurs, le **Raspberry**, la **breadboard**; l'électronique de base et des notions en probabilités/statistique.

### I. NOTION SUR LES CAPTEURS

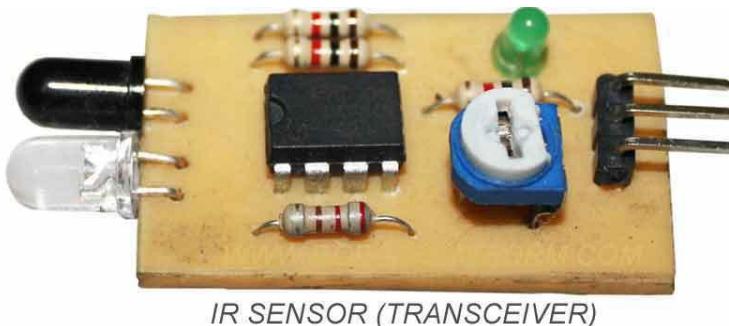


Figure 3 : Capteur infrarouge

Selon Georges Asch [6], un Capteur est un dispositif transformant l'état d'une grandeur physique observée en une grandeur utilisable. Ainsi lorsque l'on veut automatiser la quantification d'une propriété on fait appel aux capteurs. IL existe presque autant de propriété physique que de capteurs ainsi on distingue plusieurs variétés de capteurs.

#### 1- Classification des capteurs

On peut classer les capteurs en fonction de leur apport en énergie, en fonction du type de données qu'ils produisent en sortie ou en fonction du type de détection.

En fonction de leur apport en énergie on distingue deux types:

- **Les capteurs passifs:** ceux-ci ont besoin d'un apport externe d'énergie; ils fonctionnent grâce à une impédance. La variation de la propriété physique provoque une variation au niveau de cette impédance ;
- **Les capteurs actifs:** Ici le phénomène a déjà effectué la transformation en grandeur électrique.

En fonction de leur type de sortie on distingue trois types:

- **Les capteurs analogiques :** La sortie est une grandeur électrique dont la valeur est une fonction de la grandeur physique mesurée par le capteur. La sortie peut prendre une infinité de valeurs continues (tension, courant).

- **Les capteurs numériques** : La sortie de ce type de capteur est un état logique que l'on note 0 ou 1; définissant l'existence ou non d'une propriété.
- **Les capteurs logiques** : La sortie est une séquence d'états logiques qui, en se suivant, forment un nombre. La sortie peut prendre une infinité de valeurs discrètes (série de code numérique binaire).

## 2- Caractéristique des capteurs

La grande variété de capteurs, possède des caractéristiques communes qui permettent de différencier les capteurs les uns des autres. On distingue donc plusieurs caractéristiques parmi les plus courants, entre autre:

- **La grandeur physique** : elle représente la propriété physique que le capteur doit mesurer (température, humidité, lumière etc...);
- **Étendue de la mesure** : représente le domaine de variation possible de la valeur à mesurer;
- **Sensibilité** : désigne la variation de la valeur en sortie par rapport à celle fournie en entrée;
- **Résolution**: est la plus petite valeur en entrée qui produit un résultat perceptible en sortie;
- **Le temps de réponse**: Il représente le temps qu'il faut au capteur de produire une sortie en fonction de l'entrée.

## 3- Type de capteurs

Propriété	Description	Exemples
Angle	Le codeur rotatif est un capteur de position angulaire lié mécaniquement à l'axe de rotation du système sur lequel on travaille.	Capteur ENA28,
Contraintes	Mesure de la variation d'allongement	Corde vibrante, piézo-électrique, plot magnétique
Courant	Mesure du courant électrique	Capteur de courant à effet de Hall, Shunt, fluxgate, Capteur de courant à effet de Néel.
Champ magnétique	l'élément de base ici est le magnétomètre	Capteur magnétique à effet de Hall, Capteur magnétique à effet de Faraday , fluxgate fluxmètre etc...
Débit	Ceci peut s'agir du débit d'écoulement d'un fluide à un autre type de débit	tube de Pitot, débitmètre à effet vortex, débitmètre ionic etc...

Déplacement	mesure le déplacement, la distance ou la position d'un objet à mesure	Capteur de déplacement
Force	Convertir une force appliquée sur un objet en grandeur électrique	Capteur de force
Inertiel	Mesure la position du centre d'inertie d'un objet	accéléromètre, inclinomètre, gyromètre etc...
Lumière	Mesure le niveau de lumière dans un milieu	photodiode, cellule photoélectrique etc...
Niveau	il peut s'agir du niveau du son, d'une onde ou autre	capteur à pression différentielle, à ultra son, à flotteur etc...
Position	détermine la position d'un objet dans une surface	Souris, codeur, corde vibrante, capteur de proximité
Pression	Convertir des vibrations de tensions en tension électrique	tube de bourdon, baromètre, hypsomètre etc...
Son	Transforme un signal acoustique en signal électrique	microphone, hydrophone etc...
Température	Mesure la température dans un milieu	thermomètre, thermistance, thermocouple etc...

Tableau 1 : Type de capteurs

La plupart des capteurs n'intègre pas directement un écran affichage et une unité de traitement des données recueillies; pour cela on greffe généralement un capteur à un micro-ordinateur disposant des pic comme ARDUINO, RASPBERRY ou autre.

## II. NOTIONS SUR LE RASPBERRY

**Le Raspberry Pi** est un *nano ordinateur* (Petites tailles), *nano carte* (seule la carte est fournie avec ses extensions) créé par **David BRABEN**. Cet ordinateur avait été créé pour favoriser l'apprentissage de l'informatique, car le coût se retrouvait réduit, il fallait donc acheter les périphériques en fonction de l'usage que l'on veut en faire en fonction des extensions disponibles sur le modèle.

Le Raspberry Pi a évolué au fil des années et comptait après sa cinquième année à plus de 13 millions d'utilisateurs; ce qui a fait évoluer sa communauté. Et de nos jours plusieurs autres extensions ont été ajoutées pour faire du nano ordinateur une entité performante et partiellement complète.

### 1- Spécifications générales Raspberry Pi



Figure 4 : Raspberry Pi modèle 3B

En 2018 les Raspberry Pi en disponibles disposent de composants suivant:

- un processeur ARM11 à 700MHz de la fondation britannique ARM;
- un, deux ou trois ports USB;
- un port Ethernet RJ45;
- une carte Wifi (Optionnelle);
- une carte Bluetooth (Optionnelle);
- une mémoire vive variant de 256 Mo à 1G;
- un circuit graphique BMC VideoCore.

## 2- Modèle de Raspberry Pi

Depuis 2012 on compte 3 grands modèles de Raspberry Pi (A, B et Zéro). Le tableau [7] ci-dessous illustre des différents modèles ainsi que des ajouts effectués en fonction des précédents modèles:

	<b>Modèle A</b>	<b>Modèle A+</b>	<b>Modèle 1 B</b>	<b>Modèle 1 B+</b>	<b>Modèle Zero<sup>72</sup></b>	<b>Modèle Zero W</b>	<b>Modèle Zero WH</b>	<b>Modèle 2 B</b>	<b>Modèle 3 B</b>	<b>Modèle 3 B+</b>
Prix de lancement :	25 \$ US <sup>1</sup>	20 \$ US	35 \$ US <sup>73, 74</sup>		5 \$ US	10 \$ US	15 \$ US	35 \$ US <sup>75</sup>		35 \$ US
Date de lancement :	Février 2013	Novembre 2014	Avril-Juin 2012	Juillet 2014	Novembre 2015	Février 2017	Janvier 2018	Février 2015	Février 2016	Mars 2018
SoC :			Broadcom BCM2835 (CPU, GPU, DSP, SDRAM, et 1 port USB) <sup>76</sup>					Broadcom BCM2836 <sup>77</sup>	Broadcom BCM2837	Broadcom BCM2837B0
CPU :		700 MHz ARM1176jZF-S core (ARM11) <sup>76</sup>		1 GHz ARM1176jZF-S core (ARM11)			900 MHz quadricœur ARM Cortex-A7 (jeu d'instructions ARM v7) <sup>77</sup>		1,2 GHz quadricœur ARM Cortex-A53	1,4 GHz quadricœur ARM Cortex-A53
GPU :		Broadcom VideoCore IV <sup>78</sup> , OpenGL ES 2.0, MPEG-2 et VC-1 (avec licence), 1080p30 h.264/MPEG-4 AVC high-profile decodeur et encodeur <sup>76, 77</sup>								
Mémoire (SDRAM) :	256 Mo (intégré avec GPU)	512 Mo (intégré avec GPU) au 15 octobre 2012 <sup>64</sup>						1 Go <sup>77</sup>		
Nombre de ports USB 2.0 <sup>79</sup> :	1 (directement sur BCM2835 chip)		2	4 <sup>64</sup>	1 (Micro-USB)				4	
Sorties vidéos <sup>1</sup> :	HDMI et Composite (via une Prise RCA)	HDMI et Composite (via un connecteur Jack)	HDMI et Composite (via une Prise RCA)	HDMI et Composite (via un connecteur Jack) <sup>80</sup>	Mini HDMI et Composite (via soudures)			HDMI et Composite (via un connecteur Jack)		
Sorties audio <sup>1</sup> :		stéréo Jack 3,5 mm (sortie son 5.1 sur la prise HDMI) et Composite		stéréo Jack 3,5 mm (sortie son 5.1 sur la prise HDMI) et Composite		Mini-HDMI		stéréo Jack 3,5 mm (sortie son 5.1 sur la prise HDMI) et Composite		
Unité de lecture/écriture :	SD / MMC / fente pour carte SDIO (3,3 V)	MicroSD <sup>64</sup>	SD / MMC / fente pour carte SDIO (3,3 V)			MicroSD <sup>64</sup>				
Carte/connectique réseau <sup>1</sup> :	Non		10/100 Ethernet		Non	Wifi 802.11n, Bluetooth 4.1	10/100 Ethernet	10/100 Ethernet, Wifi 802.11n, Bluetooth 4.1	10/100/1000 Ethernet (Max. 300Mbps <sup>81</sup> ), Wifi 802.11ac, Bluetooth 4.2	
Périphériques bas niveau:	8 × GPIO, UART, I <sup>2</sup> C bus, SPI bus avec deux chip selects, I <sup>2</sup> S audio <sup>82</sup> , +3.3 V, +5 V <sup>83</sup>	17 × GPIO, UART, I <sup>2</sup> C bus, SPI bus avec deux chip selects, I <sup>2</sup> S audio, +3.3 V, +5 V <sup>83</sup>	8 × GPIO, UART, I <sup>2</sup> C bus, SPI bus avec deux chip selects, I <sup>2</sup> S audio, +3.3 V, +5 V <sup>83</sup>	17 × GPIO, UART, I <sup>2</sup> C bus, SPI bus avec deux chip selects, I <sup>2</sup> S audio, +3.3 V, +5 V <sup>83</sup>	GPIO à souder	17 × GPIO, UART, I <sup>2</sup> C bus, SPI bus avec deux chip selects, I <sup>2</sup> S audio, +3.3 V, +5 V <sup>83</sup>				
Puissance nominale :	300 mA (1,5 W) <sup>84</sup>	200 mA (1 W) <sup>85</sup>	700 mA (3,5 W)	600 mA (3 W)	~160 mA (0,8 W)			800 mA (4 W)		
Consommation maximale mesurée <sup>86</sup> :	320 mA	230 mA	480 mA	330 mA	140 mA <sup>87</sup>	?	180 mA	350 mA	720 mA <sup>88</sup>	Non précisé
Source d'alimentation <sup>1</sup> :			5 volt via Micro-B USB ou GPIO header							Micro-B USB ou GPIO header ou Power Over Ethernet
Dimensions :	85,60 mm × 53,98 mm × 17 mm <sup>89</sup>	65 mm × 53,98 mm × 17 mm	85,60 mm × 53,98 mm × 17 mm <sup>77, 89</sup>	65 mm × 31 mm × 5 mm	65 mm × 31 mm × 13 mm		85,60 mm × 53,98 mm × 17 mm			
Poids :	45 g <sup>1</sup>	23 g <sup>1</sup>	45 g <sup>1</sup>	9 g	12 g			45 g		
Systèmes d'exploitation :	Debian GNU/Linux, Raspbian OS, Fedora, Arch Linux ARM <sup>1</sup> , RISC OS, FreeBSD, Plan 9, Kali Linux							idem modèle 1 + Snappy Ubuntu Core <sup>90</sup> , SolydX RPi <sup>91</sup> [archive], Windows 10 IoT <sup>77</sup>		

Tableau 2 : Tableau comparatifs des modèles de Raspberry

Dans tous ces modèles de Raspberry Pi, on dispose des Pins qui permettent de lui intégrer des composantes électroniques et manipuler ces dernières.

### 3- Pins du Raspberry Pi

Chaque Raspberry possède des pins qui lui permettent d'intégrer des composantes électrotechniques qui ne sont pas pris en compte par les extensions par défaut. Les dispositions des broches peuvent varier d'un modèle de Raspberry à un autre.

### a. Présentation des Pins

Les différents modèles possèdent entre 26 pattes et 40 pattes, parmi les plus utilisées dans le monde de l'électronique on distingue deux grands groupes :

- **Alimentation :**

- le Raspberry dispose d'une ou plusieurs alimentations à **5 Volts** pour les composants nécessitant beaucoup de tension;
- elle dispose aussi d'une ou plusieurs alimentations **3 Volts** pour celles qui consomment un peu moins;
- enfin **Ground** : ou terre qui constitue à la tension nulle

- **GPIO (Général Purpose Input/Output)** ce sont des ports qui peuvent être configurés en entrée (lit la valeur à son entrée) ou sortie (peut faire varier la tension qu'elle génère de 0 à 3Volts).

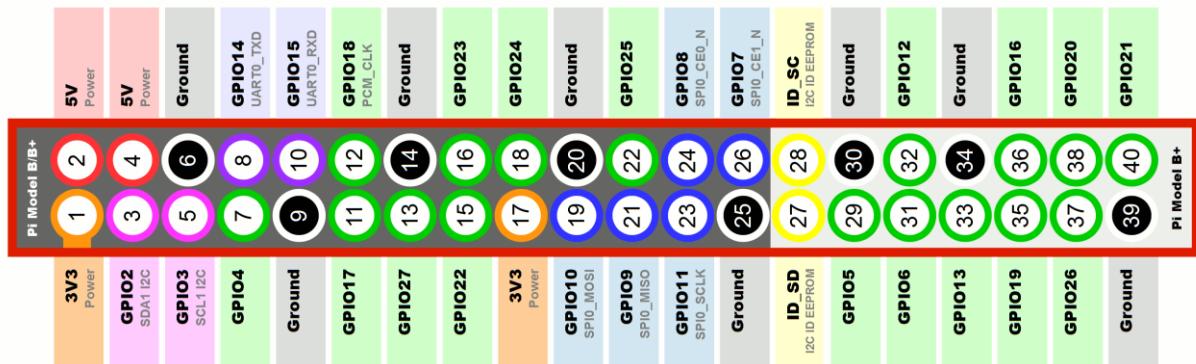


Tableau 3 : Pins modèle 3B

Sur la figure ci-dessus est illustrée la disposition des pins sur le modèle 3B des Raspberry, dessus on peut distinguer les alimentations de 5V (en rouge), les alimentations 3V (en orange), les bornes terre (en gris) et les ports d'entrée sortie (bleu et vert).

### b. Exemple d'utilisation

Pour utiliser, on adapte le composant directement sur les pins à travers des fils que l'on relie à la breadBoard.

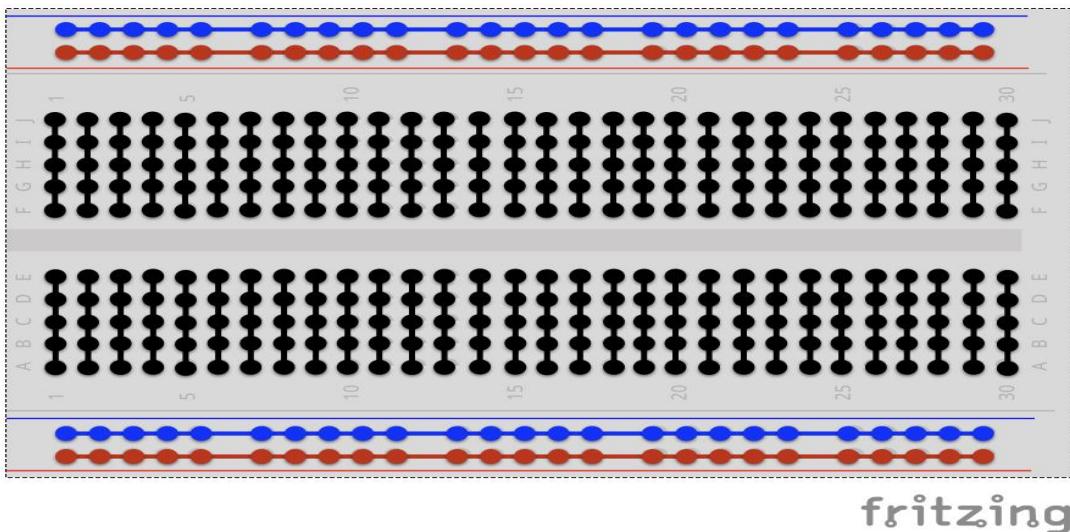


Tableau 4 : exemple de BreadBoard

Un Breadboard (platine de prototypage) est un dispositif permettant de réaliser le prototype d'un circuit électronique et de le tester. La figure ci-dessus montre l'exemple d'une platine de prototypage.

- **Les petits points circulaires** représentent des points de contact ou l'on insère des fils pour alimenter ou recueillir des tensions ; les points noirs sont indexés dans un plan à deux dimensions où les lettres (dans ce cas d'exemple) vont de A à I (en ordonnée) et de 1 à 30 (en abscisse). On retrouve un point sur le Breadboard à travers un couple (abscisse, ordonnée);
- **La ligne horizontale Rouge**: elle représente la ligne qui doit être alimentée par une tension positive non nulle. Lorsque l'on fournit une tension sur l'un des points d'une ligne, toute la ligne est automatiquement alimentée par cette tension (elle peut être branchée soit sur une patte 5V, soit sur une patte 3V ou encore à une patte GPIO);
- **La ligne horizontale Bleue** : elle représente la ligne qui doit être alimentée par une tension nulle. Elle sert de borne négative à tout composant qui sera branché sur sa ligne (elle peut être branchée soit sur une patte Ground soit sur une patte GPIO à tension nulle);
- **La ligne verticale Noire** : elle représente la ligne sur laquelle on branche les composants électroniques, si un point dans la ligne 1 est alimenté, alors tous les points de la ligne 1 sont alimentés (elle est généralement branchée sur une patte GPIO en fonction des besoins);

#### Exemple d'utilisation :

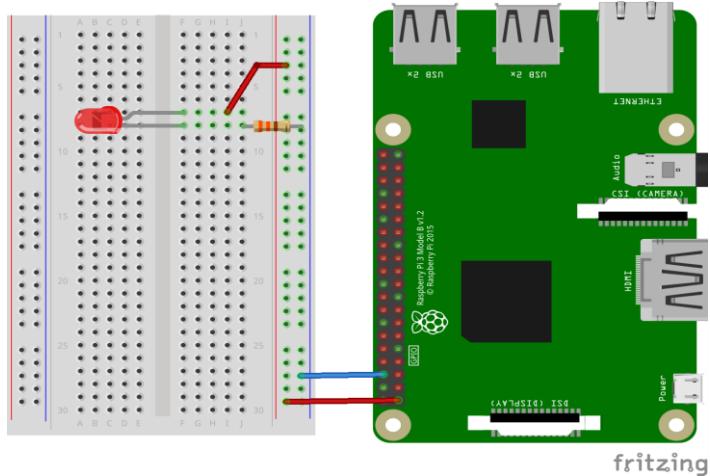


Tableau 5 : Exemple d'utilisation sur une led

Dans l'exemple ci-dessus on considère une LED rouge que l'on branche sur les points (12, F) pour la borne négative et (13, F) pour la borne positive. On effectue un branchement en série avec une résistance que l'on alimente grâce à la patte de 3volts du Raspberry.

Cet exemple est assez simple, et explique de façon basique le fonctionnement d'une platine de prototypage; mais pour des prototypes plus compliqués et pour tenir compte de la sensibilité des équipements, il est nécessaire d'avoir des notions en électronique de base.

### III. NOTION DE BASE DE L'ÉLECTROTECHNIQUE

Lorsque nous manipulons des circuits électriques, la non maîtrise de certaines notions d'électronique peut faire courir plusieurs risques à savoir:

- Des risques sur nous:** le fait de traiter des tensions fortes peut nous faire générer des coupes circuits qui peuvent dangereusement affecter notre santé et le tableau ci-dessous l'illustre.

Effets	Courant continu	courant alternatif
<i>Légère sensation de picotement</i>	0,6 - 1,0 mA	0,3 - 0,4 mA
<i>Sensation évidente</i>	3,5 - 5,2 mA	0,7 - 1,1 mA
<i>Douleurs, mais contrôle musculaire conservé</i>	41 - 62 mA	6 - 9 mA
<i>Douleurs et incapacité de lâcher le métal</i>	51 - 76 mA	10 - 16 mA
<i>Difficultés respiratoires (paralysie de la cage thoracique)</i>	60 - 90 mA	15 - 23 mA
<i>Fibrillations cardiaques (dans les 3 secondes)</i>	500 mA	65 - 100 mA

Tableau 6 : Effet du courant sur le corps humain

- **Des risques sur les équipements** : certains équipements (parfois coûteux) sont très sensibles aux tensions auxquelles on les soumet; et une surtension peut provoquer la destruction de ces derniers;
- **Des risques sur le fonctionnement du prototype**: un prototype peut ne pas fonctionner à cause d'un composant qui a été monté maladroitement et causer plus tard des dysfonctionnements sur tout le prototype.

Soit un schéma simple comportant un générateur des dipôles et une masse, définissons quelques notions:

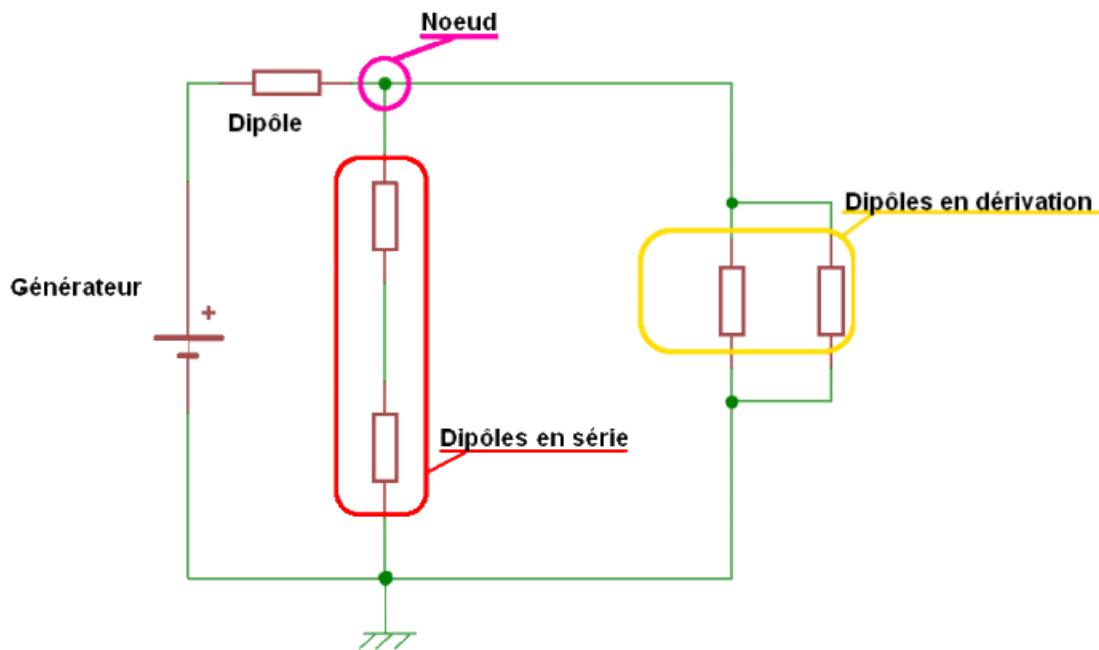


Tableau 7 : Schéma électrique

- Les composants d'un circuit sont :
  - le **générateur**, un dipôle qui fournit de l'énergie électrique dans un circuit ;
  - le **récepteur** qui reçoit quant à lui l'énergie électrique dans un circuit ;
  - le **dipôle**, un élément du circuit qui possède deux bornes. En général, les dipôles sont des récepteurs.
- Lois sur les fils et leurs liaisons :
  - un **nœud** est une connexion qui relie au moins trois fils ;
  - une **branche** est une portion de circuit (un fil) comprise entre deux nœuds consécutifs ;
  - une **maille** est un chemin fermé, formé d'un ou de plusieurs fils (ou branches) et de dipôles dans un circuit électrique.
- Lois concernant les dipôles :
  - deux dipôles sont en **série** lorsqu'ils appartiennent à la même branche ;
  - deux dipôles sont en **dérivation** (ou en **parallèle**) lorsqu'ils forment une maille.

Nous allons fixer quelques paramètres pour la suite:

**R** : Résistance d'un dipôle;

**U** : Différence de potentiel aux bornes d'un dipôle;

**I** : Intensité qui parcourt un dipôle.

## 1. Loi fondamentales

### a. Loi d'Ohm

Elle est une loi fondamentale de l'électronique et elle s'énonce comme suit : « *La différence de potentiel ou tension U (en volts) aux bornes d'un résistor de résistance R (en ohms) est proportionnelle à l'intensité du courant électrique I (en ampères) qui la traverse* » [8].

Cette loi nous permet de déterminer l'intensité qui parcourt un dipôle en lui appliquant une certaine tension; Pour illustrer cette loi on va utiliser un schéma assez simple.

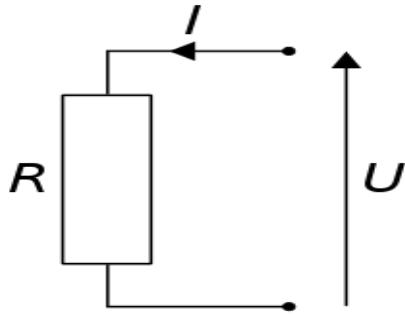


Tableau 8 : Illustration de la loi d'Ohm

$$U = R \cdot I$$

Équation 1 : Loi d'Ohm

### b. Loi des mailles

Elle est une loi fondamentale de l'électronique et elle s'énonce comme suit : « *la somme algébrique des tensions le long de la maille est constamment nulle* ». [9]

Cette loi nous permet de déterminer la tension totale qui parcourt une maille; Pour illustrer cette loi on va utiliser le schéma ci-dessous.

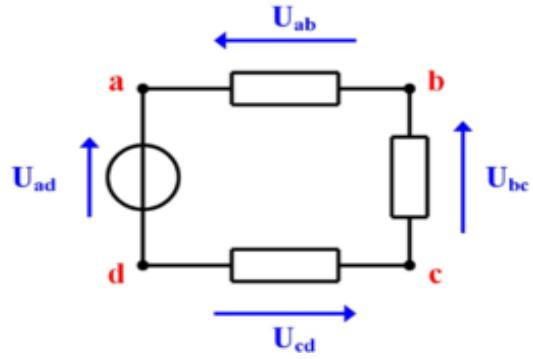


Tableau 9 : Illustration de la loi des mailles

$$\sum U_i = 0$$

Équation 2 : Loi des mailles

### c. Loi des nœuds

Elle est tout aussi une loi fondamentale de l'électronique et elle s'énonce comme suit : « *La somme des intensités des courants qui entrent par un nœud est égale à la somme des intensités des courants qui sortent du même nœud* ». [8]

Cette loi nous permet de déterminer le courant qui parcourt une branche du nœud. Pour illustrer cette loi nous allons aussi utiliser un schéma.

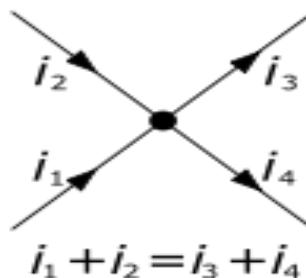


Tableau 10 : Illustration de la loi des mailles

$$\sum I_e = \sum I_s$$

Équation 3 : Loi des noeuds

## 2. Composants de base

### a. Les Résistances

Une **résistance** est un composant électronique ou électrique dont la principale caractéristique est d'opposer une plus ou moins grande résistance (*mesurée en ohms*) à la circulation du courant électrique.

Nous pouvons être face à deux situations dans un montage, soit nous avons à déterminer la résistance d'un résistor (on procède par identification par le code couleur), soit déterminer la résistance équivalente d'un groupe de résistances (on utilise des conséquences de la loi d'Ohm et celle des nœuds).

- Détermination par le code de couleurs [10]: les résistors portent un code de couleurs constitué d'environ quatre couleurs consécutives et la lecture de ce code suit certaines règles:

	<i>A</i>	<i>B</i>	<i>C</i>	<i>Tolérance</i>
<i>Noir</i>	0	0	0	
<i>Marron</i>	1	1	1	1%
<i>Rouge</i>	2	2	2	2%
<i>Orange</i>	3	3	3	
<i>Jaune</i>	4	4	4	
<i>Vert</i>	5	5	5	0.5%
<i>Bleu</i>	6	6	6	0.25%
<i>Violet</i>	7	7	7	0.1%
<i>Gris</i>	8	8	8	0.05%
<i>Blanc</i>	9	9	9	

Tableau 11 : Code couleur des résistors

$$R = (A + 10^B) \cdot 10^C$$

Équation 4 : Résistance et code couleur

- Groupement de résistances : soit deux résistors de résistance respective  $R_1$  et  $R_2$ ; ces deux résistors peuvent être montés soit en parallèle soit en série.



Figure 5 : résistors en série

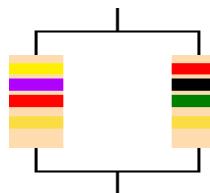


Figure 6 : résistors en parallèle

$$\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2}$$

Équation 5 : Résistances en série

$$\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2}$$

Équation 6 : Résistances en parallèle

## b. Les LEDs

Les **LEDs** (*Light emitting diode*) ou en français, Diodes électroluminescentes sont des dispositifs optoélectronique capables d'émettre de la lumière lorsqu'ils ont parcouru par un courant électrique dans le sens positif (polarisation directe).

On l'utilise généralement comme voyant lumineux pour signifier du fonctionnement ou du disfonctionnement d'un composant. Il en existe plusieurs type et de plusieurs couleurs; pour choisir une couleur de diode il faut prendre en compte plusieurs facteurs, la longueur d'onde souhaitée (comprise entre **700 nm** et **400 nm** pour avoir une lumière visible), et la tension à laquelle la diode sera soumise, le tableau ci-dessous illustre mieux ces propos.

Couleur	Longueur d'onde (nm)	Tension de seuil (V)
<i>Infrarouge</i>	$\lambda > 760$	$\Delta V < 1,63$
<i>Rouge</i>	$610 < \lambda < 760$	$1,63 < \Delta V < 2,03$
<i>Orange</i>	$590 < \lambda < 610$	$2,03 < \Delta V < 2,10$
<i>Jaune</i>	$570 < \lambda < 590$	$2,10 < \Delta V < 2,18$
<i>Vert</i>	$500 < \lambda < 570$	$2,18 < \Delta V < 2,48$
<i>Bleu</i>	$450 < \lambda < 500$	$2,48 < \Delta V < 2,76$
<i>Violet</i>	$400 < \lambda < 450$	$2,76 < \Delta V < 3,1$
<i>Ultraviolet</i>	$\lambda < 400$	$\Delta V > 3,1$
<i>Blanc</i>	Chaud à froide	$\Delta V = 3,5$

Tableau 12 : Classification des couleurs des LEDs

Après avoir parcouru les contours de ce qui compose l'architecture physique de notre produit, nous allons nous intéresser à la partie logicielle et mathématique.

## IV. INTRODUCTION A PHARO

**Pharo** est un langage OpenSource sous licence MIT et apache, inspiré du langage Smalltalk; créé par S DUCASSE et M. DENKER en avril 2008. En plus de la philosophie de codage qu'elle inspire à ses utilisateurs, ce langage actuellement à sa version 6.2 stable, procède de nombreux atouts dont la plupart sont hérités de Smalltalk.

### 1. Les caractéristiques de Pharo

Pharo est un langage qui a pour objectif d'avoir en un petit système un environnement puissant, innovant et gratuit ouvert à tous; ses caractéristiques contribuent fortement à atteindre cet objectif.

- **Pharo est un langage OpenSource** : cette politique oblige à ses contributeurs à accepter de publier leurs codes sous licence MIT.
- **Pharo est purement Orienté Objet**: en Pharo on dit généralement, tout est objet. En effet Pharo à travers son organisation et ses règles de codage encourage fortement l'encapsulation des données. Ainsi tout ce qui est codé doit être pensé dans une philosophie Orientée Objet.
- **Pharo est réflexif**: Pharo à cette capitale qu'en pleine exécution du code, l'on puisse examiner, voir modifier la structure interne de ce code. Ce qui permet de maintenir des systèmes directement en production sans recompiler tout le programme.
- **Pharo est un langage à typage dynamique**: les variables dans les programmes en Pharo n'ont pas des types statiques ; le type d'une variable, en fonction des besoins d'exécution peut s'adapter au fur et à mesure que l'on avance dans le code.
- **Pharo est simple d'apprentissage**: l'apprentissage d'un langage peut influer sur la volonté d'utiliser ce langage ; La syntaxe de Pharo est très minimale et <<peut tenir sur une carte postale>>.
- **Pharo est un puissant IDE**: Pharo est aussi un environnement de développement intégré assez puissant; ce dernier a poussé le débogage des applications à un autre niveau et de plus il propose des outils comme (Finder, System browser...) qui sont très utile lors d'un développement ; ainsi qu'une implémentation des tests assez simple et intuitive.
- **Pharo est multi plateforme**: Pharo tourne sur une machine virtuelle, ce qui permet de l'utiliser sur les différents systèmes d'exploitation.
- **Pharo possède de nombreux Framework**: Lorsque vous êtes face à un besoin particulier dans Pharo, il est probable qu'il y a une contribution qui propose une solution à votre problème; cela peut aller des mathématiques, aux statistiques en passant par les plateformes web et de l'intelligence artificielle.
- **Pharo a une communauté accessible**: La liste de diffusion de Pharo compte plus de 600 contributeurs actifs, qui vous répondent au moindre problème soulevé; en plus de cela il est enseigné dans plusieurs Universités à travers le monde.

## 2. Les présentations de Pharo

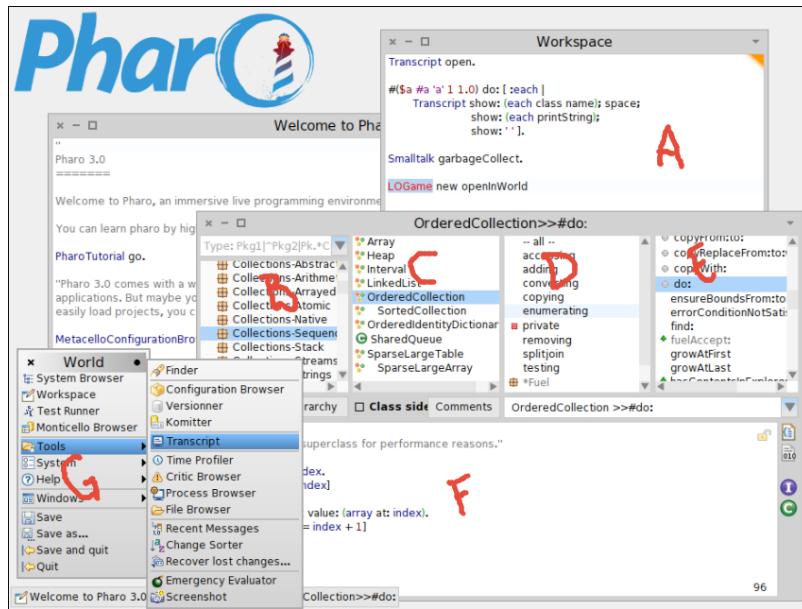


Figure 7 : Exemple d'interface de Pharo

Pharo présente une interface semblable à celle du haut, elle regroupe plusieurs parties qui sont-elles même des Objets et qui sont assez facile d'accès. Parmi les plus utilisé on note:

- **A:** Il s'agit du **Workspace**, dans certaines versions **Playground**, ce dernier permet d'une manière générale d'exécuter du code directement dans la console ;
- **B:** il s'agit de la liste des **Packages**. Tous les codes de votre image vous sont présentés, et dans un package vous pouvez créer des sous package ou **tag** ;
- **C:** Dans cette section on retrouve toutes les **classes** et les tests des classes du package sélectionné dans la précédente section.
- **D:** La liste des **protocoles**; un protocole étant ici un regroupement de classes qui traitent du même domaine. Ainsi pour l'initialisation on aura **initialize**, pour les getters et les setters on aura **accessing**, pour les méthodes qui permettent de faire le rendu on aura **render**.
- **E:** Ici on retrouve les **méthodes** du protocole précédemment sectionnés.
- **F:** c'est l'espace principale d'édition de code.
- **G:** Il s'agit du menu; il présente les outils que vous avez ajouté pour votre métier et ses outils par défaut.

## V. GÉNÉRALITÉS SUR LA MODÉLISATION MATHÉMATIQUE

**La modélisation mathématique** est une traduction d'une observation dans le but de lui appliquer les outils, les techniques et les théories mathématiques, puis généralement,

en sens inverse, la traduction des résultats mathématiques obtenus en prédictions ou opérations dans le monde réel. Cette définition assez longue peut être résumée en ces mots « La modélisation mathématique c'est le fait de prédire grâce aux outils mathématique des événements à partir des observations faites sur le sujet à partir d'un modèle ».

Le but central ici ressort donc comme étant **la prédition** d'un sujet en fonction de certaines observations. Ceci peut aller de la prédition météorologique, aux prédictions environnementales (pollution, désertification ...), dans le domaine de la santé etc... Pour effectuer cette modélisation plusieurs approches sont possibles entre autre:

- **Approche au pire des cas:** Cette approche consiste à prendre en compte la pire des situations et ainsi trouver la manière optimale. Cette approche suppose de travailler sur un grand ensemble de données et la plus part du temps la recherche se fait de manière gloutonne (théorie des jeux).
- **Approche au cas moyen:** On va se dire que si les observations sont aléatoires et qu'elles n'ont pas été biaisées, les valeurs normales devraient tourner autour d'une certaine moyenne. Toutes les études vont donc être centrées vers la moyenne.

Dans ce qui va suivre nous prendrons en compte seulement le cas d'une approche de la moyenne, nous allons donc commencé par définir nos variable et quelques mesure de tendance centrale puis, nous allons présenter les différentes étapes d'une modélisation à savoir la validation des observations, la modélisation et la validation du modèle.

## 1. Variables aléatoires et agrégats sur la moyenne

Dans la suite nous allons considérer:

**x,y:** observation scalaire;

**x,y:** observation vectorielle;

**X,Y:** variable aléatoire scalaire ou vectorielle selon le contexte;

**X,Y:** matrice d'observation.

### a. La moyenne

**La moyenne** est une mesure statistique caractérisant les éléments d'un ensemble de quantité : elle exprime la grandeur qu'aurait chacune des valeurs s'ils avaient été égaux sans changement de la dimension globale.

$$M = \frac{1}{n} \sum_{i=0}^{i < n} x_i$$

*Équation 7 : Moyenne d'une série*

Cette formula exprime la moyenne arithmétique non pondéré de la variable aléatoire X.

Exemple: moyenne ( $\{4, 1, 7\}$ )=4.

### b. La médiane

La médiane est un critère de position par rapport à la moyenne qui permet de diviser une série ordonnée en deux sous ensemble de taille identique (de manière théorique).

Exemple: médiane ( $\{3, 1, 7\}$ )=3; médiane ( $\{5, 2, 4, 1\}$ )=3.

### c. Le mode

La mode permet de distinguer la valeur la plus représentée dans une série statistique.

Exemple: mode ( $\{4, 2, 4, 3, 2, 2\}$ )=2.

### d. La variance / écart-type

L'écart-type d'une série représente la dispersion par rapport à la moyenne ou encore la moyenne quadratique des écarts par rapport à la moyenne.

$$\delta = \left[ \frac{1}{n} \sum_{i=0}^{i < n} (x_i - M)^2 \right]^{1/2}$$

*Équation 8 : Ecart type d'une série*

## 2. Validation des échantillons

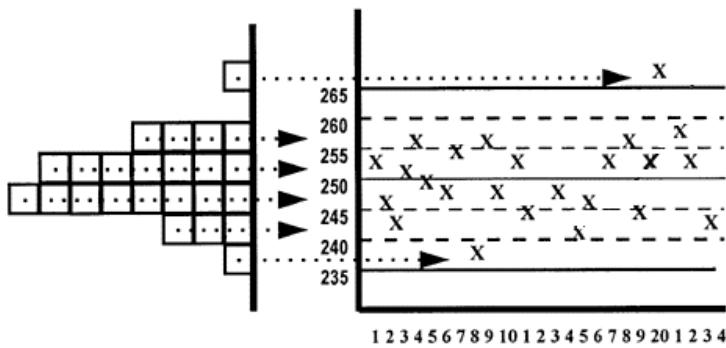
Une modélisation n'est possible que lorsqu'on dispose d'un ensemble de données qui forment une base de connaissances ; et dans une série aléatoire les valeurs normales ne doivent pas trop s'éloigner de la moyenne, dans l'éventualité où l'on retrouve une valeur qui est très à l'extrême, on considère qu'il s'agit d'un cas isolé (laisser cette valeur telle quelle est risque influencer la moyenne et ainsi tous les calculs se feront sur une fausse moyenne). Pour résoudre ce problème de cas isolé on peut utiliser deux approches:

- On retire ce cas de la liste des échantillons;
- Soit on modifie sa valeur en lui donnant une valeur qu'elle aurait dû avoir.

Pour pouvoir savoir la valeur qui est trop éloignée de la moyenne on peut procéder par plusieurs méthodes et ici nous utiliserons l'approche de WESTGARD qui sera illustré par les graphes de la représentation de Levey-Jennings.

### a. Représentation de Levey-Jennings

Pour évaluer une série par rapport au reste des mesures, il est plus facile d'avoir une représentation qui tienne compte de l'historique des mesures. On représente alors chaque point en fonction du jour auquel la mesure a été faite. C'est la représentation de Levey-Jennings. Pour faciliter l'interprétation d'un tel graphe, il est courant de représenter les points en fonction de l'écart à la moyenne par rapport à la déviation standard.



Sur cette représentation on a la moyenne (trait fin continu), et les niveaux d'écart type; +/- 1 écart type (trait interrompu fin); +/- 2 écart type (trait interrompu fort); +/- 3 écart type (trait continu fort). Ainsi on représente les observations sur ce graphe et ainsi en fonction de la position et de la sensibilité choisie on peut éliminer une série. Et les règles de WESTGRAD définissent mieux les conditions de validation d'une série.

### b. Règles de WESTGRAD

L'utilisation de règles permet pour chaque nouveau point de décider si ce point peut être considéré comme acceptable ou non. Ces règles donnent des moyens objectifs de valider techniquement une série. Selon une habitude répandue, on abrège ces règles sous la forme A L (parfois écrits A:L) où A représente le nombre de mesures prises en compte et L représente la limite utilisée. Par exemple : 1 2s (ou 1:2s) indique une mesure excédant 2s (2 déviations standard). Les règles principales sont décrites dans le tableau ci-dessous :

Règles	Description

1:2s	Cette règle est en général considérée comme avertissement et non pas comme critère de rejet d'une série. Dans l'application des règles que nous verrons, elle est utilisée comme critère d'utilisation des autres règles.	
1:3s	Cette règle, très commune, fait partie des règles de Westgard. Elle stipule que la série de mesure doit être rejetée lorsqu'une mesure excède la moyenne de plus ou de moins de 3 écart-types.	
2:2s	Peut être utilisée avec un contrôle ou avec deux. Si on utilise deux contrôles, la comparaison est faite sur une même série. Avec un contrôle, la comparaison est faite dans la même série.	
R:4s	Uniquement avec deux contrôles. La règle est violée si l'écart entre les deux contrôles est supérieur à 4s.	
4:1s	Avec un contrôle : violation si quatre valeurs (de quatre séries) sont en dessus ou en dessous de 1s. Avec deux contrôles, la comparaison se fait sur deux séries.	
10moy	Violation lorsque 10 valeurs qui se suivent se trouvent sur le même côté de la moyenne.	

Figure 9 : Règles de WESTGUARD

Ce tableau présente les règles de WESTGUARD, la première colonne nous présente la règle, dans la seconde nous présente la description de la règle et la troisième nous donne une illustration de la règle.

### 3. Détermination du modèle

*Information disponible* : des observations de  $X : (x_1, \dots, x_n)$ , ou des observations de  $(X, Y) : (x_1, y_1), \dots, (x_n, y_n)$ .

Soit  $D_n := \{(X_1, Y_1), \dots, (X_n, Y_n)\}$  un ensemble de données d'entraînement.

Les  $X_i$  sont des variables d'entrées à valeur dans un ensemble  $X$ . De même les  $Y$  sont des variables de sortie à valeur dans un ensemble  $Y$ .

**Sortie** : Une solution du problème de prévision est un prédicteur.

$f : X \rightarrow Y$  Soit  $F = F(X, Y)$  l'ensemble des prédicteurs.

Si on a une nouvelle observation  $X_{n+1} \in X$  alors  $f(X_{n+1})$  est un candidat pour prévoir la valeur de  $Y_{n+1}$  (non observée).

#### a. Tableau Marginal

Nous pouvons nous retrouver à croiser des variables de nature différente (la température et le niveau d'oxygène; l'âge et le poids) il faut donc normaliser ces valeurs sur une même base, nous allons introduire des notions sur les probabilités, dresser un tableau marginal, puis construire un nuage de points représentant la dispersion des points enfin nous allons de manière graphique dresser la fonction motrice de notre modèle.

- Probabilité que  $X$  prenne la valeur  $x$  et  $Y$  celle de  $y$  :

$$P(X=x, Y=y) = (\text{nombre d'apparitions de } (x,y) \text{ dans } D) / (\text{nombre d'éléments dans } D)$$

- la distribution marginale de  $X$  est la probabilité de  $X$  lorsqu'elle prend la valeur  $x$  est :

$$P(X = x) = \sum_y P(X = x, Y = y)$$

Équation 9 : Probabilité  $P(X=x)$

- la distribution marginale de  $y$  : est la probabilité de  $Y$  lorsqu'elle prend la valeur  $y$  est :

$$P(Y = y) = \sum_x P(X = x, Y = y)$$

Équation 10 : Probabilité  $P(Y=y)$

On obtient un tableau comme celui-ci:

$X/Y$	$x_1$	$x_2$	...	$x_n$	<b>Total</b>
$y_1$	$P(X = x_1, Y = y_1)$	$P(X = x_2, Y = y_1)$	...	$P(X = x_n, Y = y_1)$	$P(Y = y_1)$
$y_2$	$P(X = x_1, Y = y_2)$	$P(X = x_2, Y = y_2)$	...	$P(X = x_n, Y = y_2)$	$P(Y = y_2)$
...	...	...	...	...	...
$y_n$	$P(X = x_1, Y = y_n)$	$P(X = x_2, Y = y_n)$	...	$P(X = x_n, Y = y_n)$	$P(Y = y_n)$
<b>Total</b>	$P(X = x_1)$	$P(X = x_2)$	...	$P(X = x_n)$	<b>1</b>

Tableau 13 : Tableau marginal de la variable aléatoire X et Y

### b. Nuage de points

Un **nuage de points** est une représentation de données dépendant de plusieurs variables. Il permet de mettre en évidence le degré de corrélation entre au moins deux variables liées.

Les différentes observations des nuages de points permettent de déterminer :

- Des tendances et dépendances entre les variables aléatoires;
- Des relations positives, négatives, directes, indirectes ou inverses.
- Des répartitions plus ou moins homogènes.
- Des données aberrantes s'écartant de la moyenne.

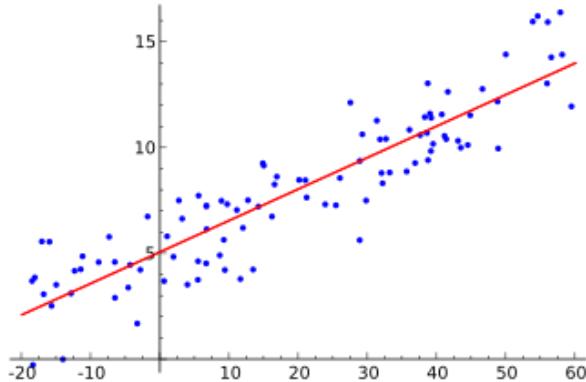


Figure 10 : Exemple de nuage de points

Sur cette figure on voit une représentation en nuage de points d'une série de données, on peut remarquer que ces deux distributions ont une relation positive qui a tendance à suivre une droite. Dans ce cas de figure on peut déterminer (de manière graphique) une droite de régression (droite de couleur rouge). Ainsi la fonction principale du modèle sera l'équation de droite de cette droite de régression.

### c. Étude de la corrélation

Étudier la corrélation entre deux variables revient à étudier l'intensité de la liaison qui peut exister entre ces deux variables.

Nous allons étudier le nuage de points, pour expliquer quelques critères qualitatifs d'appréciation de la corrélation entre deux séries:

– intensité de la relation

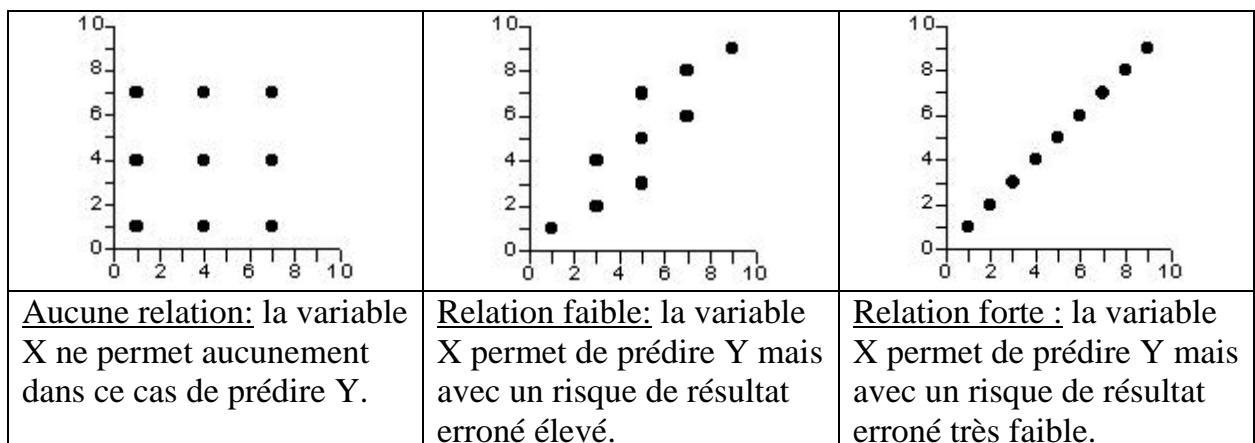


Tableau 14 : intensité d'un nuage de points

– forme de la relation :

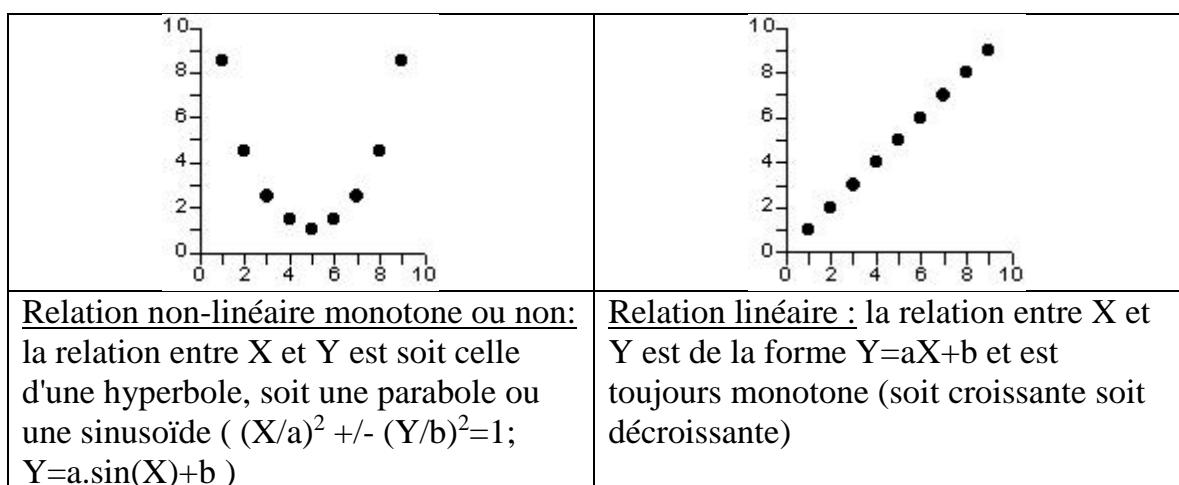
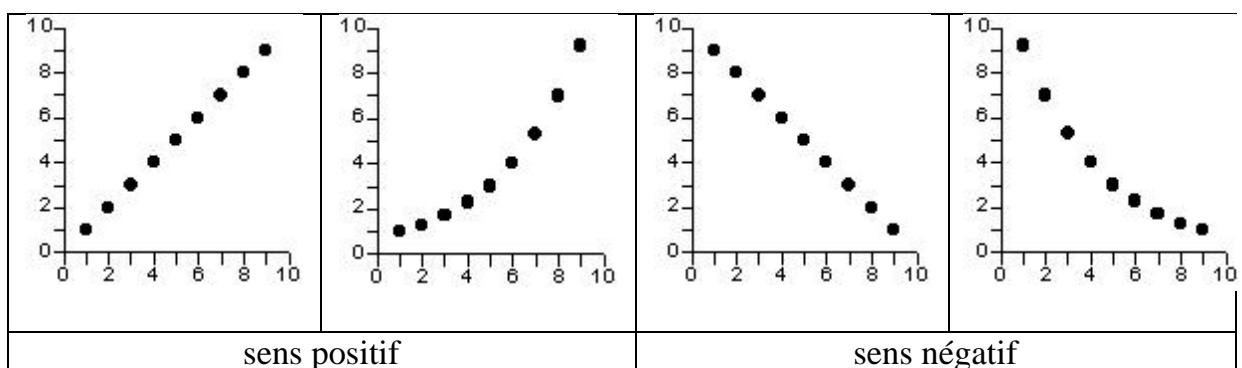


Tableau 15 : forme d'un nuage de points

– sens de la relation



X et Y évolue dans le même sens; les grandes valeurs de X correspondent aux grandes de Y	X et Y évolue dans des sens contraire; les grandes valeurs de X correspondent aux petites de Y		
<u>Linéaire</u>	<u>Non linéaire</u>	<u>Linéaire</u>	<u>Non linéaire</u>

Tableau 16 : sens d'un nuage de points

Pour ce qui est de l'étude quantitative de la corrélation entre deux variables, on utilise le coefficient de corrélation qui donne en pourcentage l'intensité de corrélation entre les deux variables. Il se calcule à partir de la covariance des deux variables et de leurs variances (écart type) respectives:

$$Cov(X, Y) = \left[ \frac{1}{N} \sum_i (Xi - Yi) \right] - Mx * My$$

Équation 11 : Covariance de X et Y

$$var(X) = \frac{1}{N} \sum_i (Xi - Mx)^2$$

$$var(Y) = \frac{1}{N} \sum_i (Yi - My)^2$$

Équation 13 : Variance de X

Équation 12 : Variance de Y

$$r(X, Y) = \frac{Cov(X, Y)}{\sqrt{var(X)*var(Y)}}$$

Équation 14 : Coefficient de corrélation

Un nuage de point linaire et ayant une intensité forte aura un coefficient de corrélation proche de 1 tandis qu'un qui n'a aucune relation sera proche de zéro. Le coefficient est aussi utilisé pour déterminer la dépendance entre deux variables aléatoires.

#### d. Étude de la dépendance

On dit que deux variables X et Y sont indépendantes lors que X ne peut permettre de déterminer Y; dans ce cas, une situation de dépendance linéaire aura un **coefficent de corrélation nul**.

- X et Y sont indépendants cela veut dire que  $Cov(X, Y)=0$ ;
- X et Y sont indépendants cela veut dire que  $P(X=x_i, Y=y_i) = P(X=x_i)*P(Y=y_i)$

#### 4. Validation du modèle

Supposons que l'on cherche à vérifier une loi du type  $y=f(x)$  prévue par un modèle théorique à partir de n mesures  $(y_i, x_i)$ .

Tester la validité de la loi  $y=f(x)$  c'est répondre à deux questions :

- Si l'on suppose la loi valide, quelle est la courbe d'ajustement (courbe de régression) qui présente une correspondance maximale avec les données expérimentales ?
- Une fois l'ajustement effectué, peut-on dire si les écarts entre la courbe de régression et les points sont significatifs ?
  - o S'ils sont imputables aux erreurs de mesure, alors rien ne permet de réfuter la loi.
  - o Si le désaccord est significatif, il faut en chercher l'origine (problème de calcul, hypothèse du modèle à réfuter, etc).

De manière plus formelle, on va définir une fonction  $c_{out}$  qui va représenter la qualité de prédiction de notre modèle.

Soit  $c : Y \times Y \rightarrow R$  telle que :  $\forall (y, y_0) \in Y \times Y$

- $c(y, y_0) \geq 0$
- $c(y, y) = 0$

Cette fonction est la fonction de perte ou  **$c_{out}$** . L'objectif est de déterminer  $f \in Y$  telle que  $c(f(X_{n+1}), Y_{n+1})$  soit petit en moyenne.

# CHAPITRE III : MODÉLISATION ET CONCEPTION

## Introduction

Après avoir présenté notre contexte et ressorti notre problématique, nous avons exploré les différents aspects qui interviennent ou influencent le projet SDA. Nous entrons ainsi dans le vif du sujet en présentant le projet SDA proprement dit, puis analyser ses besoins, ensuite définir une méthode de travail et déterminer les technologies à utiliser, enfin passer à la modélisation de la version de base.

## I. PRÉSENTATION DE SDA

Une infrastructure est définie par wikipédia, comme « une ensemble d'éléments interconnectés qui fournissent le cadre pour supporter la totalité de la structure ». En tant qu'infrastructure la SDA interconnecte à travers des interconnexions réseaux plusieurs équipements physiques dont le contrôle et l'administration sont gérés par des outils logiciels.

Pour être plus clair SDA a pour but de fournir à des laborantins, chercheurs et autres la possibilité d'effectuer des prélèvements périodique et automatique sur leurs cible; puis à partir d'une plate-forme logiciel récupérer ces données les épurer puis en faire une analyse.

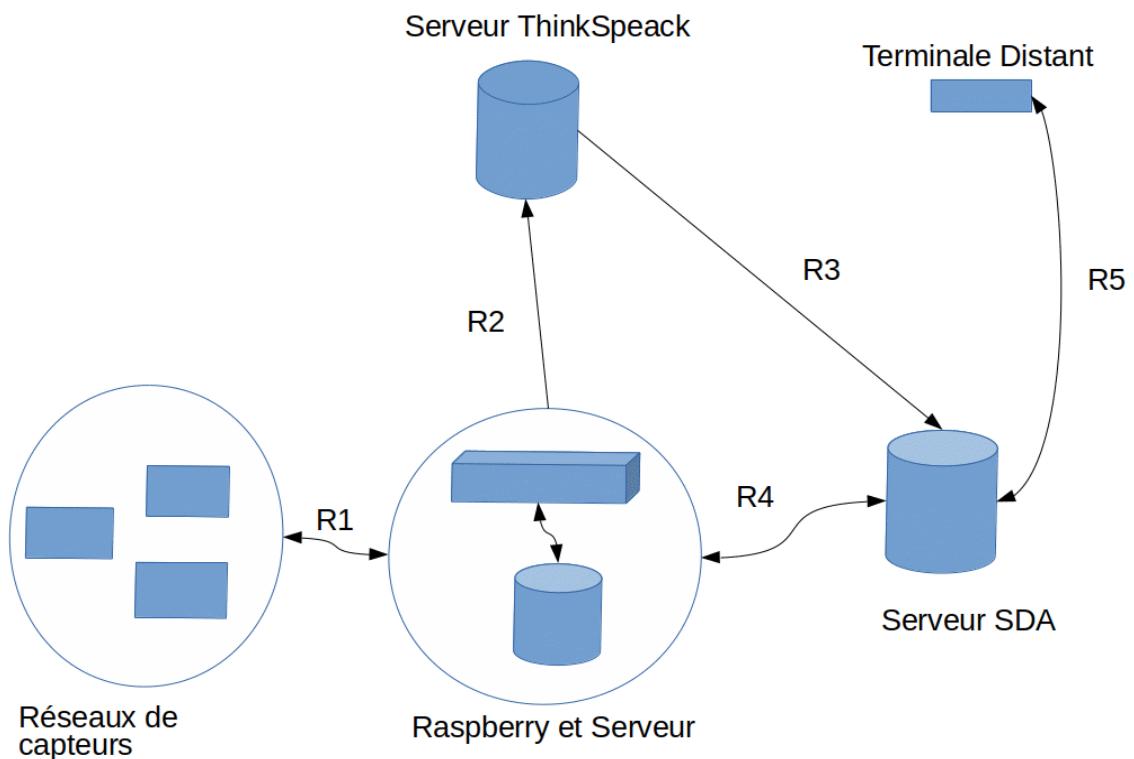


Figure 11 : Architecture générale de SDA

La figure ci-dessus présente l'architecture fonctionnelle de SDQ; avec les interconnexions existantes entre les différents équipements.

## 1. Réseau de capteurs

Tout commence à ce niveau, on branche sur le sujet les capteurs adéquats pour la récupération des informations ciblées, ces capteurs doivent communiquer avec le Raspberry Pi et être contrôlés au travers de cette dernière (**Liaison R1**). Il faut mettre sur pied un schéma électrique qui va permettre de faire fonctionner les différents capteurs, en les alimentant avec une bonne tension et en redirigeant la sortie vers un pin GPIO du Raspberry.

## 2. Raspberry et Serveur

Le Raspberry va se connecter aux différents capteurs à travers une filaire établie sur les Pins du Raspberry (**Liaison R1**). Cette connexion aura pour rôle:

- Implémenter les pilotes pour pouvoir comprendre les informations que lui transmettent les différents capteurs en fonction de leurs caractéristiques;
- Brancher les capteurs sur la bonne tension d'alimentation;
- Allumer et éteindre un capteur à partir des pins et d'une plateforme d'administration.

Le Raspberry va se connecter au serveur de la SDA par une connexion Ethernet ou Wifi en fonction du modèle du Raspberry (**Liaison R4**). Cette connexion aura pour rôle:

- Implémenter un protocole pour pouvoir envoyer les données au Serveur de la SDA;
- Implémenter un protocole pour permettre qu'on puisse contrôler le réseau de capteurs; à travers le serveur local qui va attendre des connexions et commandes du côté du serveur SDA;
- Permettre de configurer le Raspberry directement depuis le Serveur SDA;

Le Raspberry va se connecter au serveur de ThinkSpeak par une connexion Ethernet ou Wifi en fonction du modèle du Raspberry (**Liaison R2**). Cette connexion aura pour rôle:

- Implémenter un protocole pour pouvoir envoyer les données des capteurs au Serveur de ThinkSpeak

## 3. Serveur ThinkSpeak

ThinkSpeak est une plateforme en ligne qui permet d'envoyer des données des capteurs, de pouvoir les récupérer à partir de ses APIs et de pouvoir les exporter dans un format de votre choix. Ses principaux rôles sont d'importer les données venant du

Raspberry et permettre au serveur SDA de récupérer ces données de d'en envoyer d'autres si nécessaire.

#### 4. Serveur SDA

C'est sur ce serveur qu'est implantée la plateforme d'épuration des données et d'analyse. Pour permettre l'analyse, il doit pouvoir:

- Configurer le Raspberry et les capteurs (**Liaison 4**);
- Récupérer les informations du Raspberry directement (**Liaison 4**);
- Extraire les informations du Serveur de ThinkSpeak (**Liaison 3**).
- Répondre aux requêtes des clients distants (**Liaison 5**).

Une fois que le processus de récolte a été mis sur pied, la plateforme devrait utiliser ces données pour permettre à son utilisateur d'effectuer des analyses statistiques dessus pour s'en servir dans le cadre de son étude. Nous présentons par la suite les fonctionnalités et le cahier de charge que devra respecter la future infrastructure.

## II. ANALYSE DES BESOINS

Les besoins ici définissent les attentes exprimées que devront respecter le produit final. On peut en distinguer deux principales à savoir : les fonctionnels et les non-fonctionnels. Chaque exigence possède les attributs suivants:

- **un identifiant unique** : qui définit de façon unique un besoin ;
- **la catégorie** : qui permet de savoir si le besoin est fonctionnel ou non ;
- **la description** ;
- **la liste des termes** utilisés pour la description d'un besoin ;
- **la justification** : qui permet de ressortir la présence de ce besoin ;
- **la priorité** : les valeurs possibles pour ce champ sont, haute, moyenne ou faible ;
- **la vérification** : permet de justifier si le besoin a été implémenté tel que décrit afin d'éviter que le test dépende de notre implémentation.

### 1. Les besoins fonctionnels

Ceux-ci sont les besoins qui se rapportent aux tâches spécifiques de l'infrastructure. Par soucis d'organisation nous avons segmenté l'infrastructure en fonction des différents composants et leurs interconnexions:

## a) Entre le Raspberry et les Capteurs

Dans cette partie il s'agit des besoins liés à la partie électronique, mais aussi logiciel. Il s'agit entre autre de la récupération et de l'envoie des données:

- Connexion à sur le Raspberry

<i>Identifiant</i>	<i>F1</i>
<i>Catégorie</i>	Fonctionnel
<i>Description</i>	Le système devra permettre de se connecter sur le Raspberry, pour pouvoir la configurer et retirer des données.
<i>Liste des termes</i>	Capteurs, Raspberry, PharoThing, SSH, Pharo
<i>Justification</i>	le Raspberry est un nano-ordinateur et on doit pouvoir l'utiliser à travers une autre machine sans avoir à installer sur lui tous les périphériques
<i>Priorité</i>	haute
<i>Vérification</i>	Se connecter à distance par SSH à le Raspberry

Tableau 17 : Connexion à sur le Raspberry

- Contrôler l'alimentation des capteurs

<i>Identifiant</i>	<i>F2</i>
<i>Catégorie</i>	Fonctionnel
<i>Description</i>	Le Raspberry doit pouvoir programmer l'allumage ou l'extension des capteurs
<i>Liste des termes</i>	Capteur, Raspberry, PharoThing, GPIO
<i>Justification</i>	les capteurs doivent être allumés lorsqu'on veut récupérer les données ou le mettre en veille lorsqu'il ne récupère rien.
<i>Priorité</i>	Moyenne
<i>Vérification</i>	Éteindre l'alimentation d'un capteur à partir du Raspberry.

Tableau 18 : Contrôler l'alimentation des capteurs

- Interpréter les données venant des capteurs

<i>Identifiant</i>	<i>F3</i>
<i>Catégorie</i>	Fonctionnel
<i>Description</i>	Récupéré les prélèvements des capteurs
<i>Liste des termes</i>	Capteur, Raspberry, PharoThing, GPIO
<i>Justification</i>	Permettre à qu'à partir du Raspberry on ait accès aux prélèvements des capteurs.
<i>Priorité</i>	Moyenne
<i>Vérification</i>	Visualiser les données des capteurs

Tableau 19 : Interpréter les données venant des capteurs

### b) Liaison entre le Raspberry et le Serveur SDA

Dans cette section on gère la configuration du Raspberry, pour que celle-ci puisse faire au mieux la récupération et l'envoie des données:

- Charger du code dans le Raspberry

<i>Identifiant</i>	<i>F4</i>
<i>Catégorie</i>	Fonctionnel
<i>Description</i>	Charger des codes sur le Raspberry
<i>Liste des termes</i>	Capteurs, Raspberry, PharoThing, SSH, Pharo
<i>Justification</i>	Le Raspberry doit pouvoir comprendre les données des capteurs, pour cela il faut pour chaque type de capteurs intégré le pilote qui va permettre d'interpréter les données pour ce capteurs.
<i>Priorité</i>	haute
<i>Vérification</i>	Se connecter à distance par SSH à le Raspberry

Tableau 20 : Charger du code dans le Raspberry

- Importer les données

<i>Identifiant</i>	<i>F5</i>
<i>Catégorie</i>	Fonctionnel
<i>Description</i>	Le serveur doit récupérer directement les données du capteur depuis le Raspberry
<i>Liste des termes</i>	Capteur, Raspberry, PharoThing, GPIO
<i>Justification</i>	Il peut y avoir la connexion internet qui est disfonctionnelle il faut donc importer les données en local et les traiter directement.
<i>Priorité</i>	Moyenne
<i>Vérification</i>	Éteindre l'alimentation d'un capteur à partir du Raspberry.

Tableau 21 : Importer les données

- Interpréter les données venant des capteurs

<i>Identifiant</i>	<i>F6</i>
<i>Catégorie</i>	Fonctionnel
<i>Description</i>	Importer les données prises par les capteurs sur le serveur
<i>Liste des termes</i>	Capteur, Raspberry, PharoThing, GPIO

<i>Justification</i>	Permettre à qu'à partir du Raspberry on ait accès aux prélèvements des capteurs.
<i>Priorité</i>	Moyenne
<i>Vérification</i>	Importer les données dans la plateforme et les analyser.

Tableau 22 : Interpréter les données venant des capteurs

### c) Serveur SDA

Vu l'ensemble élevé des fonctionnalités de la plateforme web de l'infrastructure, on va diviser le reste de fonctionnalités en module; en fonction des grands axe que suit le projet.

Nous distinguons plusieurs parties entre autre:

- **Les laboratoires d'analyse:** le laboratoire représente ici le milieu dans lequel l'expérience a lieu; on peut y travailler avec plusieurs capteurs, répartis dans plusieurs zones;
- **Les capteurs:** les données peuvent être directement importées, ou peuvent venir d'une plateforme en ligne. Chaque capteur est utilisé dans une zone d'un unique laboratoire ; et peut générer des alertes en fonction des limites fixées;
- **Les alertes:** Ces dernières permettent de savoir si des données ne respectent pas les normes ; et l'action que l'on doit effectuer dans le cas où il y a une situation qui se présente;
- **Analyse:** C'est la partie qui doit effectuer les validations et les traitements sur les données recueillies.

#### i. Fonctionnalités du module Laboratoire

- Lister les laboratoires

<i>Identifiant</i>	<i>F7</i>
<i>Catégorie</i>	Fonctionnel
<i>Description</i>	Avoir une liste complète de ses laboratoires
<i>Liste des termes</i>	Capteurs, Raspberry, SeaSide , web , Pharo
<i>Justification</i>	Chaque instance de la plateforme est unique, on doit donc pouvoir voir la liste des laboratoires pour y ajouter les capteurs et visualiser les données de ces dernières
<i>Priorité</i>	Moyenne
<i>Vérification</i>	Voir sur l'interface web la liste des laboratoires enregistrés

Tableau 23 : Lister les laboratoires

- Ajouter un laboratoire

<i>Identifiant</i>	<i>F8</i>
<i>Catégorie</i>	Fonctionnel
<i>Description</i>	Ajouter un nouveau laboratoire expérimental
<i>Liste des termes</i>	Capteur, Raspberry, SeaSide, Pharo
<i>Justification</i>	Ajouter un capteur est indispensable pour pouvoir regrouper les capteurs et mieux organiser son travail.
<i>Priorité</i>	Moyenne
<i>Vérification</i>	Ajouter le capteur à partir du formulaire et vérifier qu'il existe dans la liste.

Tableau 24 : Ajouter un laboratoire

- Supprimer un laboratoire

<i>Identifiant</i>	<i>F9</i>
<i>Catégorie</i>	Fonctionnel
<i>Description</i>	Supprimer un laboratoire existant expérimental
<i>Liste des termes</i>	Capteur, Raspberry, SeaSide, Pharo
<i>Justification</i>	Pouvoir supprimer un capteur est utile pour l'organisation de son travail.
<i>Priorité</i>	Moyenne
<i>Vérification</i>	Supprimer le capteur à partir de la liste et vérifier qu'il existe dans la liste.

Tableau 25 : Supprimer un laboratoire

- Ajouter un capteur à un laboratoire

<i>Identifiant</i>	<i>F10</i>
<i>Catégorie</i>	Fonctionnel
<i>Description</i>	Ajouter un capteur existant à un laboratoire
<i>Liste des termes</i>	Capteur, Raspberry, SeaSide, Pharo
<i>Justification</i>	Pouvoir ajouter un capteur à un laboratoire est toujours dans l'optique d'une bonne organisation du travail.
<i>Priorité</i>	Haute
<i>Vérification</i>	Ajouter le capteur à partir de la liste des capteurs disponibles et vérifier qu'il existe dans la liste des capteurs du laboratoire.

Tableau 26 : Ajouter un capteur à un laboratoire

- Supprimer un capteur à un laboratoire

<i>Identifiant</i>	<i>F11</i>
<i>Catégorie</i>	Fonctionnel
<i>Description</i>	Supprimer un capteur existant d'un laboratoire
<i>Liste des termes</i>	Capteur, Raspberry, SeaSide, Pharo

<i>Justification</i>	Pouvoir ajouter un capteur à un laboratoire est toujours dans l'optique d'une bonne organisation du travail.
<i>Priorité</i>	Moyenne
<i>Vérification</i>	Supprimer le capteur à partir de la liste et vérifier qu'il existe n'existe plus dans la liste.

Tableau 27 : Supprimer un capteur à un laboratoire

- Visualiser en un graphe les données de ses capteurs

<i>Identifiant</i>	<i>F12</i>
<i>Catégorie</i>	Fonctionnel
<i>Description</i>	Visualisé le graphe présentant les n données des capteurs
<i>Liste des termes</i>	Capteur, Raspberry, SeaSide, Pharo, Js
<i>Justification</i>	Il est plus facile d'interpréter visuellement un graphe qu'un ensemble de données numériques
<i>Priorité</i>	Moyenne
<i>Vérification</i>	Voir sur le tableau de bord le visuels des données de ses des capteurs.

Tableau 28 : Visualiser en un graphe les données de ses capteurs

## ii. Fonctionnalités du module Capteur

- Lister les Capteurs

<i>Identifiant</i>	<i>F13</i>
<i>Catégorie</i>	Fonctionnel
<i>Description</i>	Avoir une liste complète de ses capteurs
<i>Liste des termes</i>	Capteurs, Raspberry, SeaSide, web, Pharo
<i>Justification</i>	Chaque instance de la plateforme est unique, on doit donc pouvoir voir la liste des capteurs pour y ajouter les capteurs et visualiser les données de ces dernières
<i>Priorité</i>	Moyenne
<i>Vérification</i>	Voir sur l'interface web la liste des capteurs enregistrés

Tableau 29 : Lister les Capteurs

- Ajouter un capteur

<i>Identifiant</i>	<i>F14</i>
<i>Catégorie</i>	Fonctionnel
<i>Description</i>	Ajouter un nouveau capteur expérimental
<i>Liste des termes</i>	Capteur, Raspberry, SeaSide, Pharo
<i>Justification</i>	Ajouter un capteur est indispensable pour pouvoir regrouper les capteurs et mieux organiser son travail.

<i>Priorité</i>	Moyenne
<i>Vérification</i>	Ajouter le capteur à partir du formulaire et vérifier qu'il existe dans la liste.

Tableau 30 : Ajouter un capteur

- Supprimer un capteur

<i>Identifiant</i>	<i>F15</i>
<i>Catégorie</i>	Fonctionnel
<i>Description</i>	Supprimer un capteur existant expérimental
<i>Liste des termes</i>	Capteur, Raspberry, SeaSide, Pharo
<i>Justification</i>	Pouvoir supprimer un capteur est utile pour l'organisation de son travail.
<i>Priorité</i>	Moyenne
<i>Vérification</i>	Supprimer le capteur à partir de la liste et vérifier qu'il n'existe plus dans la liste.

Tableau 31 : Supprimer un capteur

- Ajouter une signalisation d'alerte un capteur

<i>Identifiant</i>	<i>F16</i>
<i>Catégorie</i>	Fonctionnel
<i>Description</i>	Ajouter une signalisation d'alerte à un capteur
<i>Liste des termes</i>	Capteur, Raspberry, SeaSide, Pharo
<i>Justification</i>	Cela permet d'avoir des signalisations d'erreur lors de l'importation des données, pour savoir si les données respectent vos conditions initiales.
<i>Priorité</i>	Haute
<i>Vérification</i>	Ajouter une alerte à partir de la liste des modèles disponibles et vérifier qu'elle dans la liste des alertes du capteur.

Tableau 32 : Ajouter une signalisation d'alerte un capteur

- Supprimer une signalisation d'alerte un capteur

<i>Identifiant</i>	<i>F17</i>
<i>Catégorie</i>	Fonctionnel
<i>Description</i>	Supprimer une signalisation d'alerte d'un capteur
<i>Liste des termes</i>	Capteur, Raspberry, SeaSide, Pharo
<i>Justification</i>	Cela permet de supprimer une signalisation d'erreur lorsque celle-ci n'importe plus ou importe peu.
<i>Priorité</i>	Haute
<i>Vérification</i>	Supprimer une alerte à partir de la liste des modèles disponibles et vérifier qu'elle n'existe plus dans la liste des alertes du capteur.

Tableau 33 : Supprimer une signalisation d'alerte un capteur

- Visualiser l'évolution des données de ses capteurs

<i>Identifiant</i>	<i>F18</i>
<i>Catégorie</i>	Fonctionnel
<i>Description</i>	Visualisé le graphe présentant les n données des capteurs
<i>Liste des termes</i>	Capteur, Raspberry, SeaSide, Pharo, Js
<i>Justification</i>	Il est plus facile d'interpréter visuellement un graphe plutôt qu'un ensemble de données numériques
<i>Priorité</i>	Moyenne
<i>Vérification</i>	Voir sur le tableau de bord le visuel des données de ses capteurs.

Tableau 34 : Visualiser l'évolution des données de ses capteurs

- Initialiser les données du capteur en important à partir d'une feuille de calcul

<i>Identifiant</i>	<i>F19</i>
<i>Catégorie</i>	Fonctionnel
<i>Description</i>	Pouvoir importer les données récupérées et listées dans un fichier Excel
<i>Liste des termes</i>	Capteur, Raspberry, SeaSide, Pharo, Excel, DataFrame
<i>Justification</i>	Dans certains cas de figure, on va récupérer des données déjà observée disponibles dans une feuille de calcul et qu'il faudra analyser ces données.
<i>Priorité</i>	Moyenne
<i>Vérification</i>	Préparer un fichier qui suit le modèle puis importer à partir de l'interface Web ou dans Pharo directement

Tableau 35 : Initialiser les données du capteur en important à partir d'une feuille de calcul

### iii. Fonctionnalités du module Alerte

- Lister les modèles d'Alertes

<i>Identifiant</i>	<i>F20</i>
<i>Catégorie</i>	Fonctionnel
<i>Description</i>	Avoir une liste complète de ses modèles Alertes
<i>Liste des termes</i>	Alertes, Raspberry, SeaSide, web, Pharo
<i>Justification</i>	On a plusieurs types d'alertes, ceci permet de savoir combien sont enregistrés et combien d'entre eux on peut utiliser
<i>Priorité</i>	Moyenne

<i>Vérification</i>	Voir sur l'interface web la liste des modèles d'Alertes enregistrés
---------------------	---

Tableau 36 : Lister les modèles d'Alertes

- Ajouter un Alerta

<i>Identifiant</i>	<i>F21</i>
<i>Catégorie</i>	Fonctionnel
<i>Description</i>	Ajouter un nouveau modèle d'Alerte
<i>Liste des termes</i>	Alerte, Raspberry, SeaSide, Pharo
<i>Justification</i>	Ajouter un modèle d'Alerte est indispensable pour pouvoir voir si une liste de données présente certaines données qui ne suivent pas les normes de mesure.
<i>Priorité</i>	Moyenne
<i>Vérification</i>	Ajouter le Alerta à partir du formulaire et vérifier qu'il existe dans la liste.

Tableau 37 : Ajouter un Alerta

- Supprimer un Alerta

<i>Identifiant</i>	<i>F22</i>
<i>Catégorie</i>	Fonctionnel
<i>Description</i>	Supprimer un modèle d'Alerte existant s'il n'est plus utile
<i>Liste des termes</i>	Alerte, Raspberry, SeaSide, Pharo
<i>Justification</i>	Pouvoir supprimer un modèle d'Alerte est utile pour l'organisation de son travail et permet d'éviter des données qui ne sont pas utilisées
<i>Priorité</i>	Moyenne
<i>Vérification</i>	Supprimer un modèle d'Alerte à partir de la liste et vérifier qu'il n'existe plus dans la liste.

Tableau 38 : Supprimer un Alerta

#### iv. Fonctionnalités du module Analyse

On retrouve dans cette partie deux grands ensembles, la validation des échantillons et l'analyse des données proprement dite.

- Lister les capteurs (Validation d'échantillon)

<i>Identifiant</i>	<i>F23</i>
<i>Catégorie</i>	Fonctionnel
<i>Description</i>	Avoir une liste des capteurs
<i>Liste des termes</i>	Alertes, Raspberry, SeaSide, web, Pharo

<i>Justification</i>	Pendant la validation des échantillons, on traite les données des capteurs indépendamment des autres. Ceci permet d'avoir la liste des capteurs possédant des données pour vérification
<i>Priorité</i>	Moyenne
<i>Vérification</i>	Voir sur l'interface web la liste des capteurs enregistrés ayant plusieurs données en local.

Tableau 39 : Lister les capteurs (Validation d'échantillon)

- Effectuer une validation selon **Levey-Jennings**

<i>Identifiant</i>	<b>F24</b>
<i>Catégorie</i>	Fonctionnel
<i>Description</i>	Ceci permet d'encadrer de manière graphique des valeurs.
<i>Liste des termes</i>	Alerte, Raspberry, SeaSide, Pharo
<i>Justification</i>	Ceci permet de visualiser les données atypiques qui peuvent s'afficher
<i>Priorité</i>	Moyenne
<i>Vérification</i>	représenter sur le graphe limitant les l'intervalle de validation des données en fonction de la sensibilité choisie.

Tableau 40 : Effectuer une validation selon Levey-Jennings

- Effectuer une validation selon les règles de **WESTGRAD**

<i>Identifiant</i>	<b>F25</b>
<i>Catégorie</i>	Fonctionnel
<i>Description</i>	Voir les quelles des règles de Wesgrad ont été respectées.
<i>Liste des termes</i>	Alerte, Raspberry, SeaSide, Pharo
<i>Justification</i>	Permet d'automatiser les tests de WESTGRAD
<i>Priorité</i>	Moyenne
<i>Vérification</i>	Vérifier que les règles validées le sont effectivement

Tableau 41 : Effectuer une validation selon les règles de WESTGRAD

- Corriger les données par interpolation linéaire

<i>Identifiant</i>	<b>F26</b>
<i>Catégorie</i>	Fonctionnel

<i>Description</i>	Faire par interpolation que les données soient cohérentes
<i>Liste des termes</i>	Alerte, Raspberry, SeaSide, Pharo
<i>Justification</i>	Pouvoir rectifier on ne supprime pas les données ou on ne les donne pas la moyenne on leur donne une valeur fonction de ses voisins
<i>Priorité</i>	Moyenne
<i>Vérification</i>	Vérifie que la variation entre une donnée et ses voisins n'est pas extrême comme avant la rectification.

Tableau 42 : Corriger les données par interpolation linéaire

- Annuler une interpolation linéaire

<i>Identifiant</i>	<i>F27</i>
<i>Catégorie</i>	Fonctionnel
<i>Description</i>	Faire un retour arrière sur une interpolation
<i>Liste des termes</i>	Alerte, Raspberry, SeaSide, Pharo
<i>Justification</i>	Pouvoir revenir en arrière au cas où l'on ne veut pas garder les valeurs issues de l'interpolation
<i>Priorité</i>	Moyenne
<i>Vérification</i>	Effectuer une interpolation linaire, l'annuler et vérifier que rien n'a finalement été modifié sur les données originelles.

Tableau 43 : Annuler une interpolation linéaire

- Lister les couples de capteurs (Corrélation des séries)

<i>Identifiant</i>	<i>F28</i>
<i>Catégorie</i>	Fonctionnel
<i>Description</i>	Avoir une liste des capteurs
<i>Liste des termes</i>	Alertes, Raspberry, SeaSide , web , Pharo
<i>Justification</i>	Permettre de choisir deux capteurs et effectuer une analyse donnée sur le couplage de leurs données.
<i>Priorité</i>	Moyenne
<i>Vérification</i>	Choisir deux capteurs et vérifier que ce sont les bons capteurs qui ont été pris en charge

Tableau 44 : Lister les couples de capteurs (Corrélation des séries)

- Dresser le tableau marginal des séries

<i>Identifiant</i>	<i>F29</i>
<i>Catégorie</i>	Fonctionnel
<i>Description</i>	Tableau de probabilités marginales
<i>Liste des termes</i>	Alertes, Raspberry, SeaSide , web , Pharo

<i>Justification</i>	Présenter un tableau qui présente les valeurs marginales du croisement des deux séries respectives des capteurs
<i>Priorité</i>	Moyenne
<i>Vérification</i>	Vérifier les probabilités individuelles et les leurs sommes.

Tableau 45 : Dresser le tableau marginal des séries

- Dresser le tableau de dépendances des séries

<i>Identifiant</i>	<i>F30</i>
<i>Catégorie</i>	Fonctionnel
<i>Description</i>	Tableau de dépendance entre les deux séries
<i>Liste des termes</i>	Alertes, Raspberry, SeaSide, web, Pharo
<i>Justification</i>	Présenter un tableau qui présente les dépendances entre les deux séries
<i>Priorité</i>	Moyenne
<i>Vérification</i>	Vérifier les probabilités individuelles et les leurs sommes.

Tableau 46 : Dresser le tableau de dépendances des séries

- Dresser le nuage des points pondéré correspondant aux deux séries

<i>Identifiant</i>	<i>F31</i>
<i>Catégorie</i>	Fonctionnel
<i>Description</i>	Dresser le nuage de points pondérés des deux séries
<i>Liste des termes</i>	Alertes, Raspberry, SeaSide, web, Pharo
<i>Justification</i>	Cette représentation permet de déterminer de manière graphique et approximative le modèle que suit la série.
<i>Priorité</i>	Moyenne
<i>Vérification</i>	Vérifier que c'est une représentation qui correspond au tableau marginale.

Tableau 47 : Dresser le nuage des points pondéré correspondant aux deux séries

- Dessiner les tableaux du modèle et du graphe moyen du nuage de point

<i>Identifiant</i>	<i>F32</i>
<i>Catégorie</i>	Fonctionnel
<i>Description</i>	Dessiner sur un même graphe le croisement des données et le modèle partiellement déterminé.
<i>Liste des termes</i>	Alertes, Raspberry, SeaSide, web, Pharo
<i>Justification</i>	Ceci permet de savoir quel est le taux de correspondance entre nos données et le modèle à déterminer.
<i>Priorité</i>	Moyenne

## Vérification

Vérifier l'effectivité du graphe du modèle.

Tableau 48 : Dessiner les tableaux du modèle et du graphe moyen du nuage de point

- Modifier le modèle

<i>Identifiant</i>	F33
<i>Catégorie</i>	Fonctionnel
<i>Description</i>	Modifier les paramètres de l'équation du modèle ainsi que le taux d'erreur accepté.
<i>Liste des termes</i>	Alertes, Raspberry, SeaSide, web, Pharo
<i>Justification</i>	Ceci permet d'améliorer la correspondance entre le modèle et le nuage de points.
<i>Priorité</i>	Moyenne
<i>Vérification</i>	Vérifier que l'équation u graphe respecte celle de notre équation mathématique.

Tableau 49 : Modifier le modèle

Pour ce qui est des besoins fonctionnels ; nous allons nous limiter à ceux-là. Passons maintenant aux non fonctionnels.

## 2. Les besoins non fonctionnels

Pour citer **Wikipédia**, « Un autre type d'exigence spécifie quelque chose sur le système lui-même, et de quelle manière il exécute ses fonctions. De telles exigences s'appellent souvent «exigences non fonctionnelle». On a donc la **disponibilité**, la **testabilité**, la **facilité de maintenance**, la **facilité d'utilisation**, l'**adaptation aux différents périphériques**. »

- Disponibilité

<i>Identifiant</i>	F34
<i>Catégorie</i>	Non Fonctionnel
<i>Description</i>	On l'obtient en divisant la durée durant laquelle SDA est opérationnel par la durée totale durant laquelle on aurait souhaité qu'il le soit.
<i>Liste des termes</i>	performance, disponibilité, temps, fonctionnement
<i>Justification</i>	SDA doit toujours être accessible lorsque son utilisateur en a besoin
<i>Priorité</i>	moyen
<i>Vérification</i>	Lancer le serveur et vérifier la disponibilité des services de SDA

Tableau 50 : Disponibilité

- Testabilité

<i>Identifiant</i>	<i>F35</i>
<i>Catégorie</i>	Non Fonctionnel
<i>Description</i>	Il doit être facile de réaliser des tests de fonctionnalité sur SDA.
<i>Liste des termes</i>	performance, test, modules
<i>Justification</i>	Pour isoler une panne, il faut pouvoir faire des tests unitaires.
<i>Priorité</i>	Haute
<i>Vérification</i>	Effectuer les tests de manière isolée des autres, recueillir des données en sorties qui confirmeront d'un problème à ce niveau ou pas.

Tableau 51 : Testabilité

- Facilite de maintenance

<i>Identifiant</i>	<i>F36</i>
<i>Catégorie</i>	Non Fonctionnel
<i>Description</i>	Il doit être possible d'isoler un problème et de le résoudre aisément.
<i>Liste des termes</i>	performance, test, modules, maintenance
<i>Justification</i>	Vu que le system doit être et doit fournir des résultats fiables, il faudrait que les pannes éventuelles soient facilement maintenables.
<i>Priorité</i>	haute
<i>Vérification</i>	Si on à un problème sur un calcul on lance la liste des tests si il y a un qui échoue-t-on ne modifie que la méthode défaillante.

Tableau 52 : Facilité de maintenance

- Facilite de d'utilisation

<i>Identifiant</i>	<i>F28</i>
<i>Catégorie</i>	Non Fonctionnel
<i>Description</i>	L'utilisateur doit pouvoir s'adapter facilement à l'interface ; et se guider de manière intuitive.
<i>Liste des termes</i>	SDA, performance, visibilité
<i>Justification</i>	Un utilisateur ne doit pas avoir du mal à réaliser une tâche, les images doivent être descriptives et les expressions doivent être explicites.
<i>Priorité</i>	moyen
<i>Vérification</i>	Exécuter les actions pour une première fois doit prendre moins d'une minute pour comprendre comment faire

Tableau 53 : Facilité de d'utilisation

- Adaptation aux écrans

<i>Identifiant</i>	F29
<i>Catégorie</i>	Non Fonctionnel
<i>Description</i>	L'interface d'administration doit pouvoir s'adapter aux différents écrans et réarranger ses éléments automatiquement.
<i>Liste des termes</i>	SDA, performance, visibilité
<i>Justification</i>	De nos jours on utilise des téléphones pour aller sur le net, des téléviseurs, des tablettes. Il est nécessaire de toucher toutes ses cibles.
<i>Priorité</i>	moyen
<i>Vérification</i>	Ouvrir l'interface avec un smartphone et constater l'adaptation des éléments.

Tableau 54 : Adaptation aux écrans

### III. MÉTHODE DE TRAVAIL ET TECHNOLOGIE UTILISÉES

Pour mener à bien ce projet nous avons opté pour une méthode de travail et choisi les technologies qui permettent d'effectuer le travail efficacement.

#### 1. Méthode de travail

Pour un usage optimal du temps et des ressources disponibles (humaines et matérielles) ; il est nécessaire de se fixer sur un processus approprié pour le développement. Parmi les Modèles de développement on présente les plus connu.

##### a. Processus de développement

Processus de développement se définit comme toutes les activités et résultats se rapportant à la construction d'un logiciel. Il regroupe quatre principaux axes les spécifications (fonctionnalités et contraintes), la conception et implémentation, la validation (vérifications et tests) et la maintenance (améliorations et adaptations futures).

Nous distinguons tout aussi quatre processus de développement entre autre.

- **Modèle en cascade** : il force une documentation et des validations par des tests après chaque phase. Mais vu qu'elle dépend d'une documentation celle-ci doit être stable et non modifiable et cela est peu réaliste;
- **Modèle incrémentaux** : il est un peu proche du modèle en cascade. il fournit vite une documentation de base du logiciel et les parties restantes du logiciel viendront s'y accrocher. Mais son architecture est difficile à maintenir surtout si le module de base est erroné.
- **Modèle en spirale** : il est formé de boucle chacune étant constituée de quatre parties (définir les objectifs, évaluer les risques, développer et valider, planifier la prochaine boucle). Plus on s'éloigne du centre de la spirale et plus on se rapproche de la livraison du produit. Vu que la boucle n'est pas unique

(boucle de faisabilité, boucle de développement...), il est difficile de lier le projet;

- **Modèle agile** : celui-ci est plus pragmatique, évolutif car le client participe en quelques sortes au développement ; mais nécessite une profonde implication des programmeurs.

Le modèle agile correspond aux attentes donc c'est suivant ce processus qu'on évoluera.

### b. Méthodes agiles

Les méthodes agiles sont des méthodologies essentiellement dédiées à la gestion de projets informatiques. Elles reposent sur des cycles de développement itératifs et adaptatifs en fonction des besoins évolutifs du client. Elles permettent notamment d'impliquer l'ensemble des collaborateurs ainsi que le client dans le développement du projet.

Ces méthodes permettent généralement de mieux répondre aux attentes du client en un temps limité (en partie grâce à l'implication de celui-ci) tout en faisant monter les collaborateurs en compétences. Ces méthodes constituent donc un gain en productivité ainsi qu'un avantage compétitif tant du côté client que du côté du fournisseur.

Parmi les méthodes agiles les plus populaires, nous avons **Extreme Programming** et **Scrum**. Ces différentes méthodes ont en commun quatre valeurs fondamentales :

- **L'équipe** : Dans les méthodes agiles, l'équipe est plus important que les outils ou les procédures de fonctionnalités. Il est primordial d'avoir une équipe soudée et qui communique;
- **L'application** : il est vital que l'application fonctionne. Le reste, et notamment la documentation technique, est une aide précieuse mais non un but en soi. Une documentation précise est utile comme moyen de communication. La documentation représente une charge de travail importante;
- **La collaboration** : Le client doit être impliqué dans le processus de développement. Pour être sûre de développer le produit en souhaité et non notre imagination;
- **L'acceptation du changement** : la planification initiale et la structure du logiciel doivent être flexibles afin de permettre l'évolution de la demande du client tout au long du projet. Les premières livraisons du logiciel vont souvent provoquer des demandes d'évolution.

### c. Implémentation de la méthode Scrum

**La méthode Scrum** est une méthode agile, créée en 2002, dont le nom est un terme emprunté au rugby qui signifie « la mêlée ». Elle s'appuie sur le découpage des

projets en itérations encore nommées « sprints ». Un sprint peut avoir une durée qui varie généralement entre deux semaines et un mois.

La méthode Scrum définit trois rôles pour un projet.

- **Le Scrum master** : Il s'agit d'une personne chargée de veiller à la mise en application de la méthode et au respect de ses objectifs. Il ne s'agit pas d'un chef de projet, mais d'une personne chargée de lever les obstacles éventuels qui empêcheraient l'avancement de l'équipe et du projet pendant les différents sprints;
- **Le product owner** : Il s'agit du représentant officiel du client au sein d'un projet Scrum. Il est l'interlocuteur principal du Scrum Master et des membres de l'équipe. Il définit les besoins du produit et rédige les spécifications. Il peut se faire aider de responsables fonctionnels pour la rédaction des spécifications. Il est également chargé de définir les priorités des users stories pour chaque sprint;
- **L'équipe (« team members »)** : Elle est constituée des personnes chargées de la réalisation du sprint et d'un produit utilisable en fin de sprint. Il peut s'agir de développeurs, architectes, personnes chargées de faire des tests fonctionnels...

Dans notre contexte, *le Scrum master* est l'encadreur professionnel, le **Dr Stinckwich Serge**; Se chargeait de nous recadrer dans l'objectif principal du travail, approuver les spécification et évolutions du travail et enfin proposer des solutions aux différents problèmes rencontré pendant le développement du produit.

*La team members* était constitués principalement de **ARMEL EMBOLLO**; dont le rôle était d'effectuer la conception, l'implémentation et la proposition de fonctionnalités sur le produit.

Les Sprints variaient entre une et deux semaines; pendant ces Sprint nous devions implémenter les objectifs fixés et en discuter si un problème se pose. Et à la fin du Sprint nous retrouver pour éliminer les obstacles ou approuver les modifications et passer au prochain Sprint.

## 2. Les technologies utilisées

Pour atteindre nos objectif et réaliser nos besoins fonctionnels et non, il faut fixer les technologies adéquates qui faciliteront le travail.

### a. Les langages utilisés

Parmi les langages utilisés; certains sont imposés et d'autres sont ajoutés comme des extensions pour pouvoir réaliser des fonctionnalités supplémentaires.

- **Le HTLM (HyperText Markup Language)** : qui est un langage de balisage, qui permet d'éditer des pages web;
- **Le CSS (Cascading Style Sheets)** : c'est un langage qui permet de mettre en forme les documents web.

- **Le JS (JavaScript)** : c'est un langage de programmation généralement utilisé sur le web pour dynamiser le contenu et interagir avec l'utilisateur et le serveur.
- **Le Smalltalk** : c'est un langage de programmation purement Objet sur lequel est basé Pharo et celui-ci intervient du côté Serveur.
- **le C** : Le C est un langage compilé et impératif utilisé ici pour une programmation de bas niveau.
- **Le Java (Android)** : c'est un langage de programmation orienté objet de haut niveau; utilisé dans notre cas pour le développement Android natif.

### b. Les principales Bibliothèques

- **Le Boostap pour Pharo:** Boostap est une bibliothèque Open Source pour le développement HTML, CSS et JS. Sa version pour un usage dans Pharo est implémentation Objet de Boostap dans le sens de Smalltalk;
- **JQuery pour Pharo:** le JQuery est une bibliothèque JS, libre facilitant l'écriture des scripts coté client dans le code HTML. Son implémentation pour un usage dans Pharo est tout aussi orientée objet;
- **SeaSide :** Est un framework utilisé pour développer les applications web avancées en Smalltalk ;
- **DataFrame :** Il s'agit d'une bibliothèque en Pharo qui offre des fonctionnalités avancées sur le traitement des données statistiques;
- **PharoThing :** est une bibliothèque basé sur **TéléPharo**, permet de coder en live sur une machine distante et utilise aussi WiringPi;
- **WiringPi :** est une bibliothèque codé en C qui permet de contrôler les PINs sur des équipements comme les Raspberry;
- **HighChart :** est une bibliothèque développée en Pharo qui permet de manipuler des graphes;
- **MPAndroidChart:** est une bibliothèque développée en JAVA qui permet de manipuler des graphes dans une application Android;

### c. Les principaux Frameworks

En programmation informatique, un Framework ou structure logicielle est un ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d'une partie d'un logiciel (architecture).

Un Framework se distingue d'une simple bibliothèque logicielle principalement par :

- son caractère générique, faiblement spécialisé, contrairement à certaines bibliothèques ; un Framework peut à ce titre être constitué de plusieurs bibliothèques chacune spécialisée dans un domaine. Un Framework peut néanmoins être spécialisé, sur un langage particulier, une plateforme spécifique, un domaine particulier;
- le cadre de travail (traduction littérale de l'anglais : Framework) qu'il impose de par sa construction même, guidant l'architecture logicielle voire conduisant le développeur à respecter certains patterns ; les bibliothèques le constituant sont alors organisées selon le même paradigme.

Les Frameworks sont donc conçus et utilisés pour modeler l'architecture des logiciels applicatifs, des applications web et des composants logiciels.

Parmi les Framework, ceux que nous avons utilisés dans ce travail sont:

- **Pharo:** Il s'agit du Framework principale de notre travail, il permet de développer en Smalltalk et intègre de nombreuses bibliothèques qui nous seront utiles.
- **Android Studio :** est l'environnement de travail officiel pour la programmation des applications mobiles Android natives, en utilisant comme principal langage, le JAVA.

### 3. La Méthodologie de codage

Nous allons dans les principaux axes de notre travail respecter certaines règles pour avoir un travail maintenant, efficace et compréhensible à la fin:

#### a. Le Patron architectural

Dans notre travail, peu importe le Framework nous allons mettre en œuvre ce patron pour faciliter la structuration et la lecture de notre code :

« Le modèle **MVC** est un modèle destiné à répondre aux besoins des applications interactives en séparant les problématiques liées aux différents composants au sein de leur architecture respective. » Wikipédia. D'après la figure suivant on constate que le modèle MVC permet d'établir les limites des différentes couches et permet de mieux structurer son code, par conséquent de mieux gérer la maintenance.

- **Modèle :** Il s'agit de la couche inférieur, il regroupe les données gérées par l'application et gère la notion d'intégrité sur les données.
- **La vue :** il s'agit de la couche supérieure, elle permet d'afficher les données sous un format précis et peut non seulement interroger le modèle et récupérer des données ; mais aussi donner des instructions au Controller pour modifier un élément au niveau du modèle.

- **Le Controller** : Il s'agit de la couche intermédiaire entre les deux précédentes. Il permet de manipuler les éléments du modèle et d'interagir avec la vue.

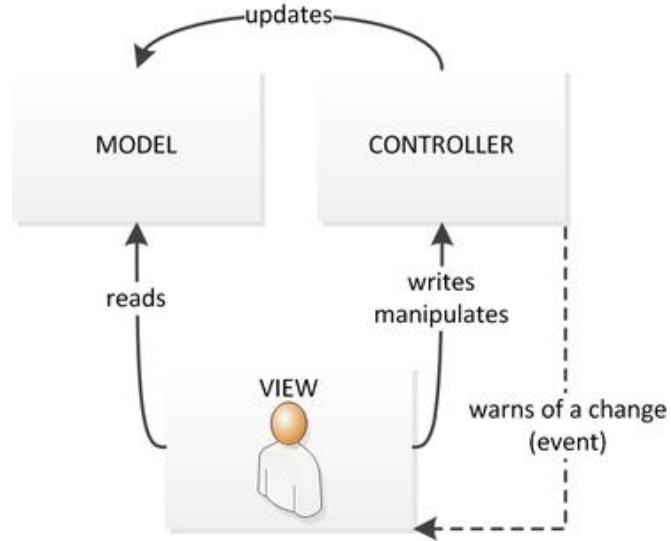


Figure 12 : Patron MVC

### b. Les paradigmes de la Programmation Orientée Objet

La grande partie de notre travail sera développée avec des langages Orientés Objet; il est donc conseiller de respecter ses paradigmes tout au long de l'implémentation. Les notions sur lesquelles nous allons plus nous tourner seront :

- **Encapsulation** : C'est l'une des premières règle, elle veut que chaque objet manipule ses attributs et l'extérieur communique avec lui à travers des messages;
- **Polymorphisme** : Elle nous permet de donner aux héritiers d'un objet un comportement différent en réponse aux même messages;
- **Héritage** : Elle permet aux sous objets qui héritent d'un objet héritent aussi de ses méthodes et attributs.

### c. Ergonomie des interfaces

Pour que l'infrastructure et les différentes applications soient le plus facile d'utilisation, et d'une compréhension intuitives ; les interfaces doivent suivre une certaine ergonomie qui sera caractérisée par les traits suivants :

- **Interface de type One Page** : Il s'agit d'un style de plus en plus répandu; le principe est simple on a une page uniforme et l'on fait changer seulement le contenu principale, sans modifier toute la page. Cette manière de faire facilite la familiarisation de l'utilisateur avec la plateforme.

## d. Charte graphique

Dans cette section nous choisissons des formes et principalement une palette de couleurs harmonie. Le fait que ces éléments soient le plus uniforme possible, permet à l'utilisateur de ne pas être surchargé par les informations non prioritaires et non fonctionnelles renvoyées.

# IV. MODÉLISATION

Pour mieux expliquer la modélisation de cette infrastructure nous devrons partir du général au particulier; autrement commencer par donner une représentation conceptuelle de l'infrastructure puis détailler celle de ses composantes. Pour éviter des légendes excessives sur les diagrammes (principalement UML), nous allons d'abord définir les principaux.

## 1. Quelques définitions

### a. Diagramme de déploiement

Il s'agit dans notre contexte d'une représentation statique de la manière dont les infrastructures physiques sont utilisées par le système ainsi que les relations existantes entre ses différentes composantes ainsi que leur répartition. De manière graphique on distingue les nœuds (sous forme de cube; il peut s'agir des ordinateurs, routeurs...), des connexions (lignes qui interconnectent: si elles sont fléchées ce sont des dépendances, sinon ce sont des associations), les composantes (surfaces rectangulaires) et les artefacts (boîtes rectangulaires).

### b. Diagramme de cas d'utilisation

Il permet de représenter les utilisateurs du système (label: **acteur**, représenté par un bonhomme), les cas d'utilisations du système (label : **use case**, représenté par un cercle) et les relations entre ces dernières. On distingue trois types de relations (soient deux cas d'utilisations A et B):

- **Inclusion** : représenté par un trait interrompu, avec le label **include**. A -> B, cela veut dire que l'exécution de A n'est possible que si B a eu lieu;
- **Extension** : représenté par un trait continu, avec le label **extend**. A -> B, cela veut dire que l'exécution de A peut être provoquée par B;
- **Généralisation** : représenté par un trait continu avec flèche en triangle. A -> B, cela veut dire que l'exécution de A est un cas de figure de B.

### c. Diagramme de classe

Il s'agit d'un diagramme qui montre l'interaction entre les différentes entités (généralement des classes composé d'un label, des attributs et des méthodes; elles sont représentés par des rectangles) qui interviennent dans le système. On distingue plusieurs types de relations (soient deux classes A et B):

- **Héritage** : représenté par un trait fléché par un triangle. A -> B, A généralise B;
- **Association** : représenté par un trait fléché ou pas;
- **Agrégation** : représenté par un trait continue avec flèche en losange vide. A -> B, B est constitué de plusieurs A;
- **Composition** : représenté par un trait continue avec flèche en losange plein. A -> B, est une agrégation avec cycle de vie dépendant entre A et B (A ne peut exister sans B, B est constitué des A).
- **Dépendance** : représenté par un trait interrompu. A -> B, certaines méthodes de A utilisent B.

*NB:* Les diagrammes qui seront utilisés dans la suite pour la plupart suivent les règles de la modélisation UML et ont été montés sur l'application UMBRELLO sur un système UBUNTU 18.04 64Bits.

## 2. Modélisation de l'infrastructure

L'infrastructure est composé de plusieurs composantes interconnectées et chacune œuvrant dans un sous-système de l'infrastructure. Nous avons trouvé que l'illustration de sa modélisation sera plus optimale avec un diagramme de déploiement:

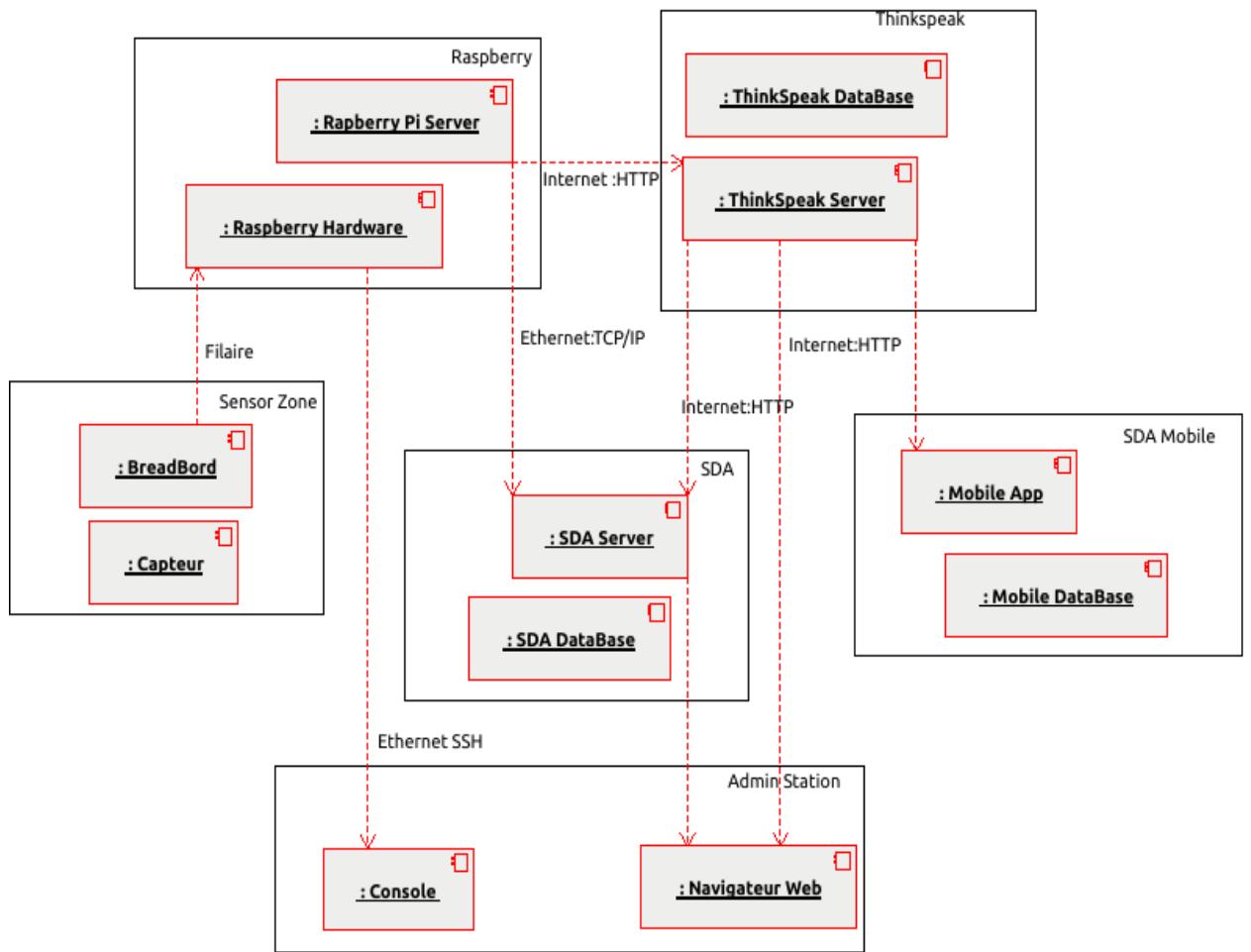


Figure 13 : Diagramme de déploiement de l'infrastructure

Dans ce diagramme nous avons représenté six sous-systèmes parmi lesquels:

### **a. Le réseau de Capteurs**

Elle représente le réseau de capteurs qui permettront de faire les relevés automatique des données; ces capteurs seront disposés dans le cas expérimental sur des BreadBoard ou seront implémentés les schémas électriques correspondants. Ce système ne sera en relation direct qu'avec le Raspberry par une simple connexion filaire.

### **b. Le Raspberry**

Dans ce système nous avons un serveur local qui va permettre de gérer les requêtes afin de faire communiquer PharoThing avec une machine distante; nous avons aussi le coté hardware qui est principalement constitué des PIN qui permettent la connexion avec le réseau de capteurs. Ce système communique avec les autres sous-systèmes ainsi:

- Connexion Filaire: avec le champ de capteur et utilise le protocole des PIN à travers les GPIO;
- Connexion internet : elle est établie principalement par le Wifi et suis le protocole HTTP pour la communication avec le serveur de ThinkSpeak;
- Connexion Ethernet: Elle utilise cette connexion avec la station d'administration qui y accède pour y effectuer des contrôlés avancés;
- Connexion SSH: elle dépend fortement de la dernière et permet à l'administrateur de s'identifier sur le Raspberry et de la contrôler.

### **c. La plateforme Web SDA**

Cette plateforme est constituée d'un serveur et d'une base de données locale pour l'importation des données; elle est connectée à le Raspberry comme expliqué plus haut. Est aussi connecté au serveur de ThinkSpeak par une connexion internet pour la visualisation des données en ligne.

### **d. Le Serveur ThinkSpeak**

Cette plateforme nous permet de stocker des données en ligne et n'est disponible ici que par une connexion internet usant du protocole HTTP. Elle tire ses données du Raspberry et des importations manuelles; et les renvoie à la plateforme Web SDA et la plateforme Mobile.

### **e. L'application Mobile**

Son rôle est de permettre le suivi des données venant du Cloud, l'envoie des données manuelles. Cette application pour tirer les données du Serveur ThinkSpeak a besoin d'une connexion internet et des API qui sont fournis.

## **3. Modélisation de l'application mobile**

Dans le cas de figure de l'application Mobile qui sera développée, pour le suivi des données; nous devons nous intéresser aux données venant du Cloud. Et pour cela nous devons configurer les accès aux APIs, les champs, les chaînes. Pour illustrer cela, nous présentons ci-après les relations entre les différentes entités, ainsi que les cas d'utilisation de l'application.

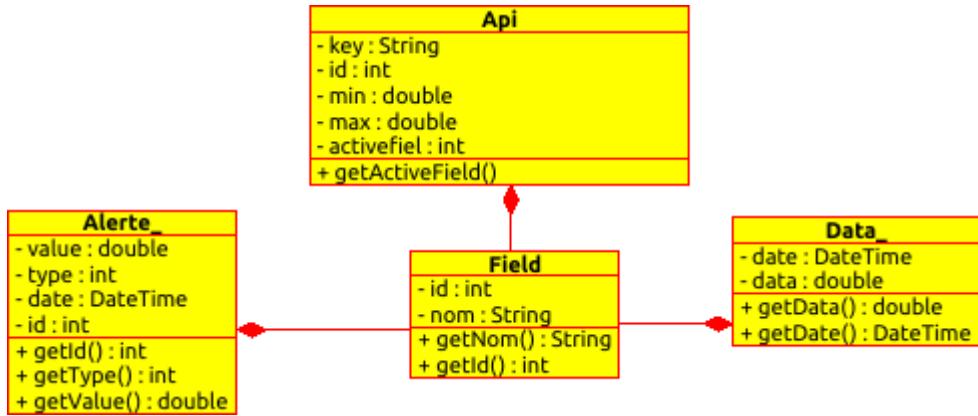


Figure 14 : Diagramme de classe de l'application mobile

Nous nous retrouvons avec cinq entités, entre autre:

- **Api**: qui est constitué de la clé **key** (la clé qui donne un accès en écriture à la chaîne) et l'identifiant (identifiant de la chaîne); en plus de ces informations on lui a défini deux attributs qui permettent de considérer un champ parmi les champs disponibles comme principale, et définir une borne maximale et minimale;
- **Alerte**: une alerte est générée lorsque l'application reçoit des données et que certaines données ne sont pas dans l'intervalle de sûreté fixé au niveau de l'API. Ainsi elle nous dira le **type** de l'alerte (si elle a dépassé la valeur maximale ou si elle est inférieure à la limite minimale). Elle permet aussi de préciser la valeur en question ainsi que la date et l'heure à laquelle elle a été prise;
- **Field**: représente un champ ou une source de données, elle a un nom, un identifiant et une liste de données;

**Data**: représente les données récupérées correspondant à un capteur en particulier.

## 4. Modélisation de l'application web SDA

Nous développons ici le noyau de l'architecture dans un langage qui est purement objet, nous allons donc éléver la modélisation sur les trois couches du patron MVC.

## a. Modélisation de la couche Modèle

Dans la couche Modèle, nos principales classes qui font l'application sont: Laboratoire, Capteur, Contact, Alertes et Data; les autres classes de cette couche qui n'influencent pas directement le projet n'apparaîtront pas ici. Nous nous retrouvons donc avec le diagramme de classe suivant:

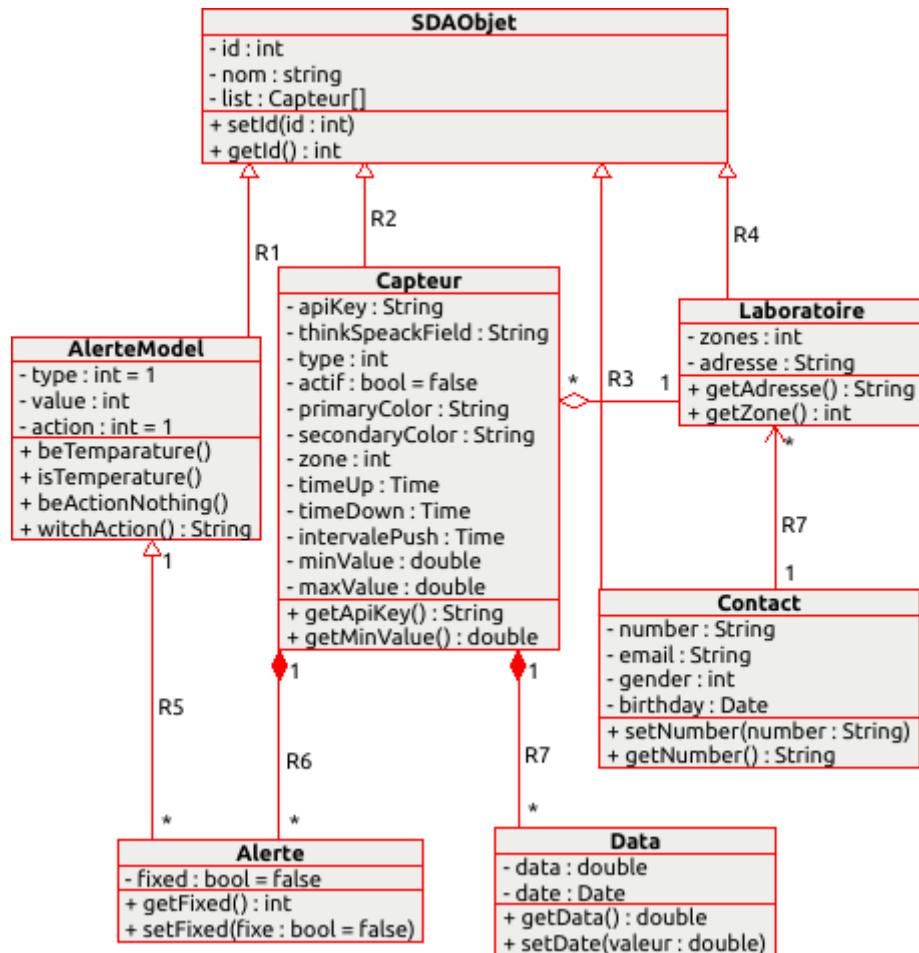


Figure 15 : Diagramme de classe de la couche Modèle SDA

Nous allons pour mieux comprendre ce diagramme, expliquer l'existence des différentes entités:

- **SDAOBJET** : On se rend compte que les entités de notre systèmes possèdent des particularités communes, qui ne demandent pas des implémentations hétérogènes, alors nous définissons cette entité qui va regrouper les intersections des différentes entités du système;
- **Laboratoire** : Un laboratoire représente ici une zone d'expérimentation, elle comporte plusieurs zones (**zone**) dans lesquelles on peut disposer plusieurs capteurs (**Capteur**), a une localisation (**adresse**) et a à sa charge un responsable (**Contact**);

- **Contact** : Cette entité existe car elle contient plusieurs informations pertinentes du responsable d'un ;
- **Data**: représente les données récupérées correspondant à un capteur en particulier.
- **AlerteModel**: représente le modèle d'un type d'alerte; elle a les principales attributs d'un modèle dans notre conception des choses: un **type** (le type d'alerte qui correspond à une comparaison précise par rapport à la valeur reçue et celle fixée comme valeur relative), une **valeur** et une **action** (l'action qui doit être effectuée lorsque l'alerte est signalée);
- **Alerte**: elle représente une donnée qui à soulever un type d'alerte défini.

## b. Modélisation de la couche Controller

Dans la couche Controller, nous allons définir principalement les couches qui permettent d'interagir avec la couche du Modèle ci-dessus. Elles permettent aussi de faire la persistance de ces données pour qu'elles soient stockées dans des ensembles et puissent être réutilisées:

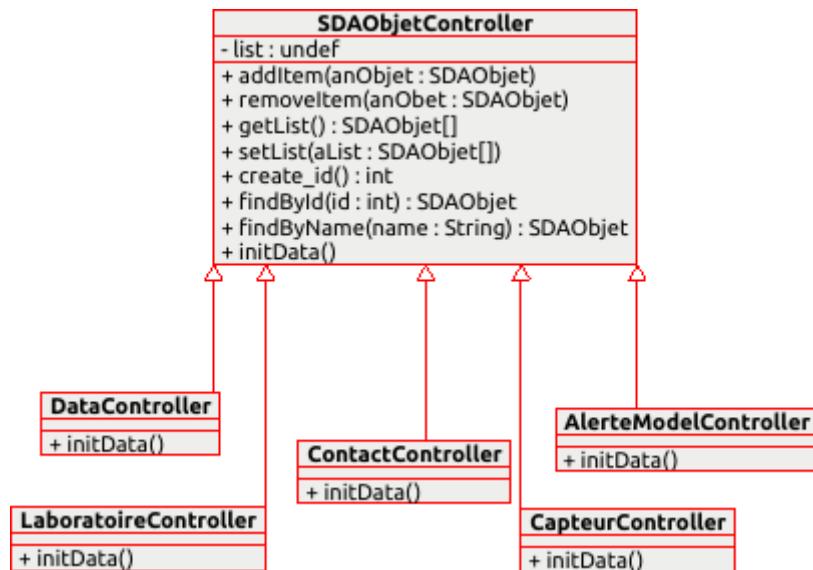


Figure 16 : Diagramme de classe de la couche Controller SDA

Nous allons pour mieux comprendre ce diagramme, expliquer l'existence des différentes entités:

- **D'objet** : On se rend compte que tous les entités de notre système possèdent des particularités communes qui ne demandent pas des implémentations hétérogènes, alors nous définissons cette entité qui va regrouper les intersections des différentes entités du système;
- **Laboratoire** : Un laboratoire représente ici une zone d'expérimentation, elle comporte plusieurs zones (**zone**) dans lesquelles on peut disposer plusieurs

capteurs (**Capteur**), a une localisation (**adresse**) et a à sa charge un responsable (**Contact**);

- **Contact** : Cette entité existe car elle contient plusieurs informations pertinentes du responsable d'un ;
- **Data**: représente les données récupérées correspondant à un capteur en particulier.
- **Alerte Model**: représente le modèle d'un type d'alerte; elle a les principales attributs d'un modèle dans notre conception des choses: un **type** (le type d'alerte qui correspond à une comparaison précise par rapport à la valeur reçue et celle fixée comme valeur relative), une **valeur** et une **action** (l'action qui doit être effectuée lorsque l'alerte est signalée);
- **Alerte**: elle représente une donnée qui à soulever un type d'alerte défini.

### c. Modélisation de la couche Vue

Nous avons décidé de travailler sur un modèle one page ; avoir une interface unique et ne dynamiser le contenu que dans certaines zones qui contiennent les informations sur la situation courante :

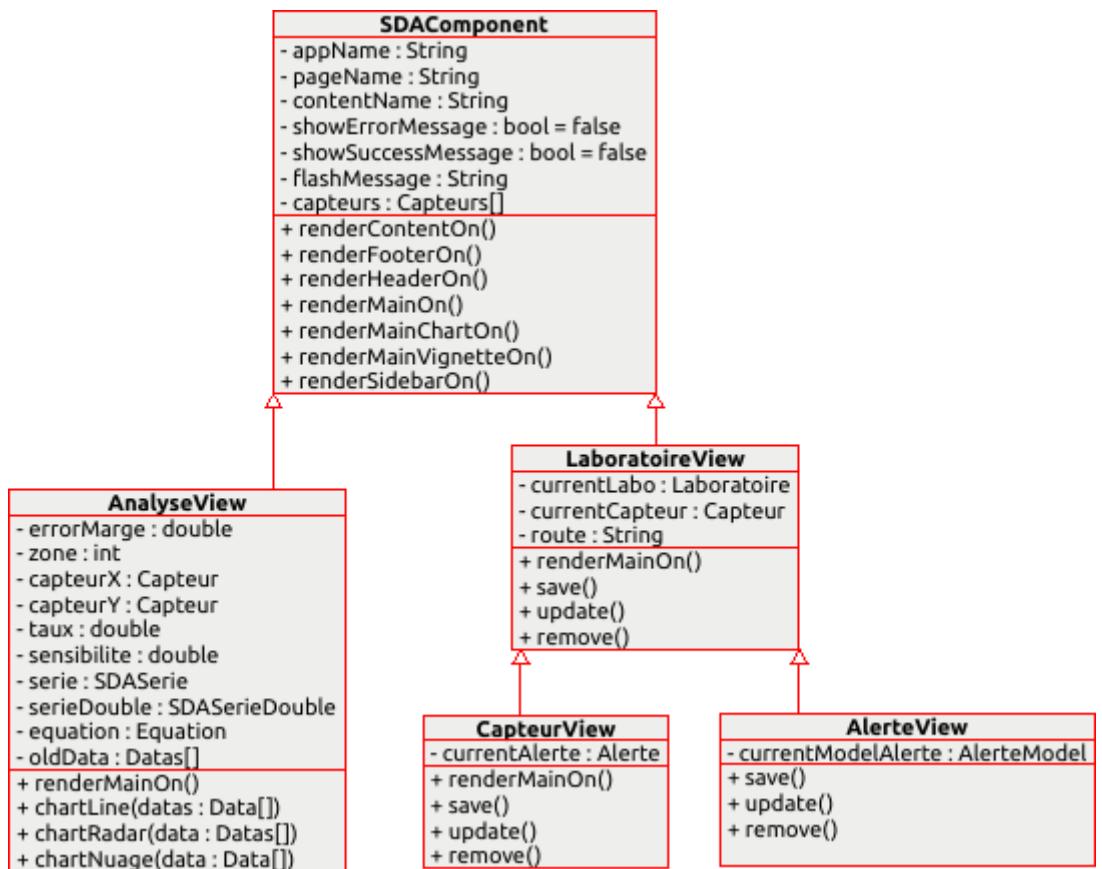


Figure 17 : Diagramme de classe de la couche vue SDA

Nous allons expliquer les deux illustrations de manière combinée ; sur le diagramme de classe nous avons les différentes vues et chaque vu remplissant certaines fonctionnalités :

- **SDAComponent** : cette vue représente la vue principale de notre application web. Elle définit les principales parties de l'interface à savoir :
  - o **L'entête** (la méthode est *RenderHeaderOn*), Cette partie a la barre haute du menu de notifications sur la plateforme, elle est statique ;
  - o **Le pied de page** (la méthode est *RenderFooterOn*). Cette partie a la barre inférieure de la plateforme, elle a des informations sur la date de modification de la plateforme, elle est statique ;
  - o **Le menu latéral gauche** (la méthode est *RenderRenderSideBarOn*). Cette partie le menu principale de la plateforme, elle est statique;
  - o **Le contenu principal** (la méthode est *RenderContentOn*) Cette partie a les informations générale sur les entités de la plateforme, elle est une partie dynamique;
  - o **Les vignettes** (la méthode est *RenderMainVignetteOn*). Cette partie a les informations générale sur les entités de la plateforme, elle a des informations dynamiques mais est statique sur toutes les pages de la plateforme;
  - o **Les Contenu principale contextuel** (la méthode est *RenderMainOn*). Elle est une partie dynamique, elle peut contenir un des diagrammes et des informations supplémentaires;
  - o **Les Graphe Principale Dynamique** (la méthode est *RenderMainChartOn*);
- **LaboratoireView** : elle permet de gérer les laboratoires et permettre les opérations CRUD sur ces dernières;

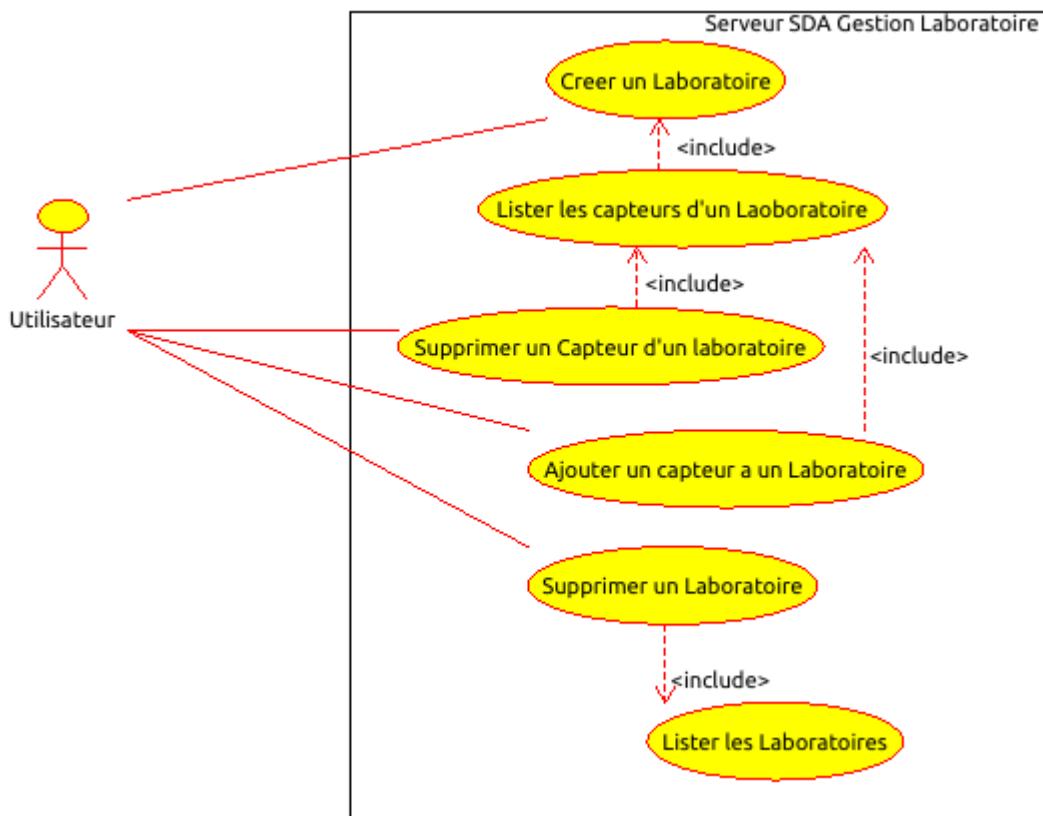


Figure 18 : Cas d'utilisation LaboratoireView SDA

Le diagramme de cas d'utilisation précédent illustre des opérations que l'on peut y effectuer et qui visent à modifier les informations concernant les Laboratoires.

- **CapteurView** : elle permet de gérer les capteurs et permettre les opérations CRUD sur ces dernières;

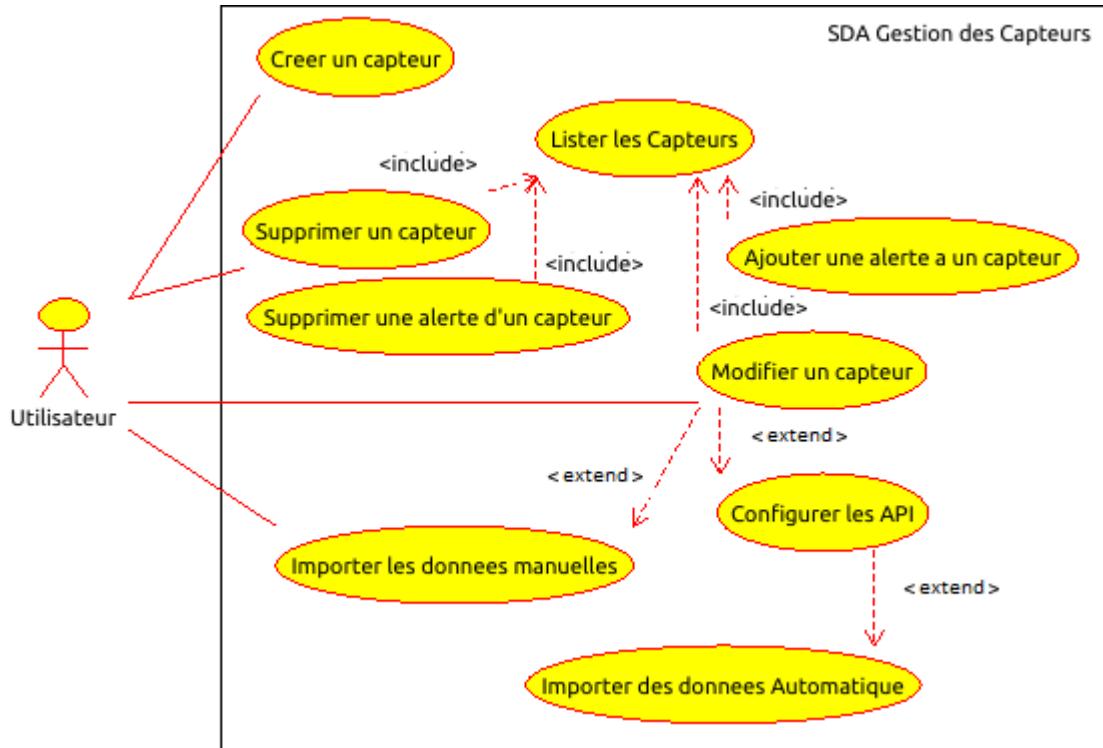


Figure 19 : Cas d'utilisation CapteurView SDA

Le diagramme de cas d'utilisation précédent illustre des opérations que l'on peut y effectuer et qui visent à modifier les informations concernant les Capteurs.

- **AnalyseView** : elle permet d'offrir une interface qui permet de faire une analyse d'échantillon et une modélisation mathématique. Nous allons expliquer chacun des modèles :
  - o **Analyse des échantillons** : Il faut ici prendre le contexte d'un capteur, puis analyser ces données pour pouvoir voir retirer les données non pertinentes et étudier les critères de validation de cette série.

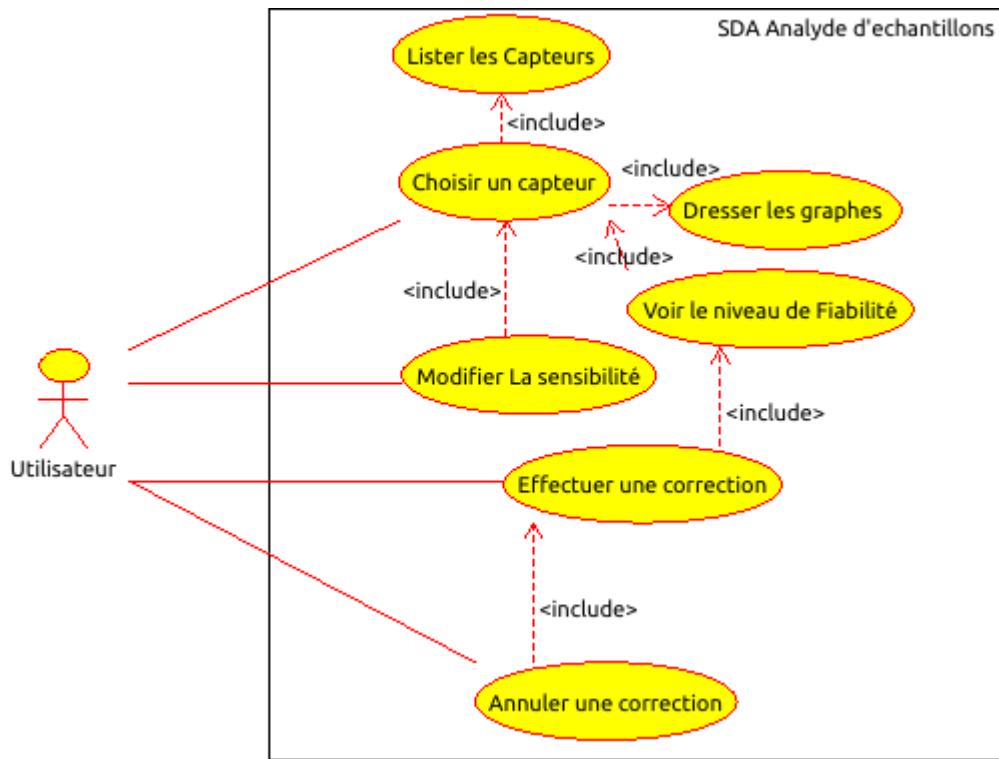


Figure 20 : Cas d'utilisation Analyse d'échantillon SDA

Ce diagramme de ce cas d'utilisation nous permet de voir le niveau de fiabilité des séries en fonction de la sensibilité définie et d'y faire une correction sur les données si cette série peut être validée.

- **Analyse des données combinatoire** : Il faut ici prendre le contexte de deux séries venant de deux sources. Le but étant de présenter des outils permettant de voir le niveau de corrélation, de dépendance entre deux séries.

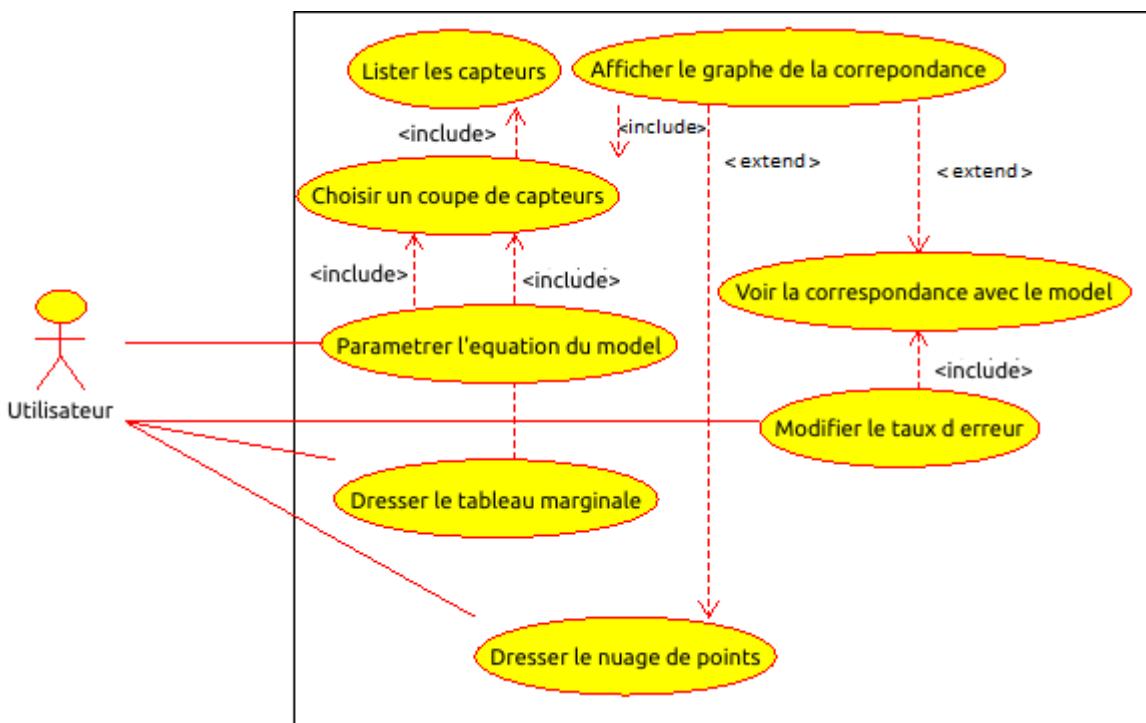


Figure 21 : Diagramme cas d'utilisation Analyse des données combinatoires

Le diagramme de cas d'utilisation présente les opérations que l'on peut effectuer sur cette interface. On choisit deux capteurs (source de données), puis de montrer plusieurs indicateurs et outils pour qui permettent d'étudier la correspondance entre les deux séries et de pouvoir définir un modèle.

## 5. Modélisation sur le Raspberry

Nous allons juste présenter ici quelques manipulations possibles sur les équipements Raspberry dans notre infrastructure. Et cela sera illustré par le diagramme de cas d'utilisation suivant.

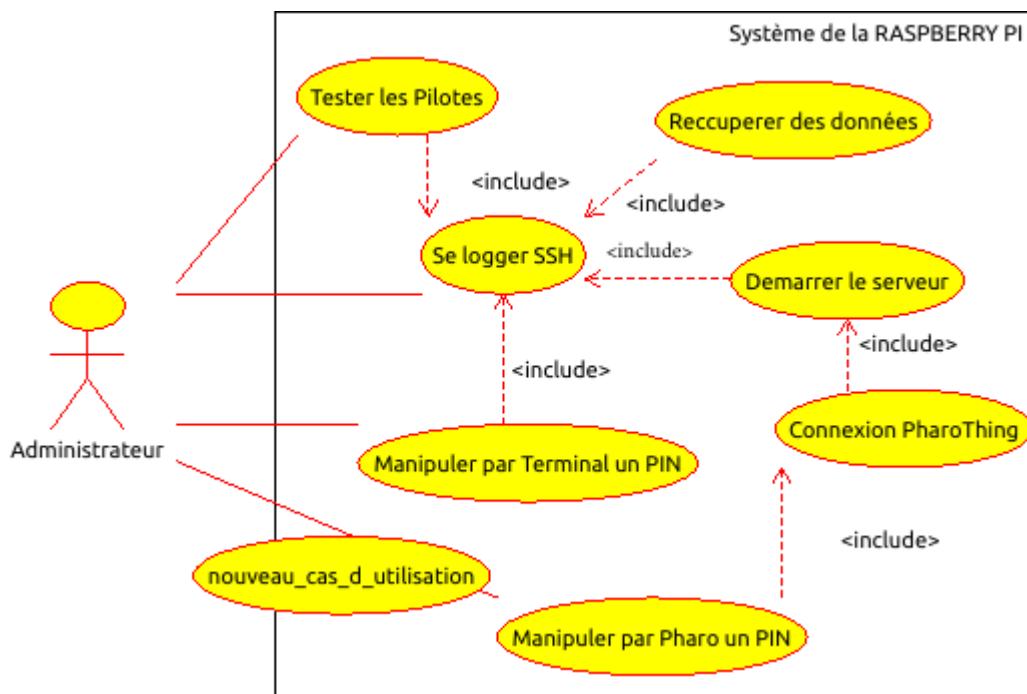


Figure 22 : Diagramme cas d'utilisation du Raspberry

Nous pouvons voir un ensemble d'opérations possibles sur le Raspberry, nous pouvons manipuler les Pins, lancer le serveur ou faire des importations de données ou de code; et pour cela il faut d'abord s'identifier par SSH.

## 6. Modélisation sur ThinkSpeak

Il s'agit-là d'un élément hors du système, mais vu sa forte implication dans ce projet pour expliquer brièvement son fonctionnement. Cela sera illustré par le diagramme de cas d'utilisation suivant.

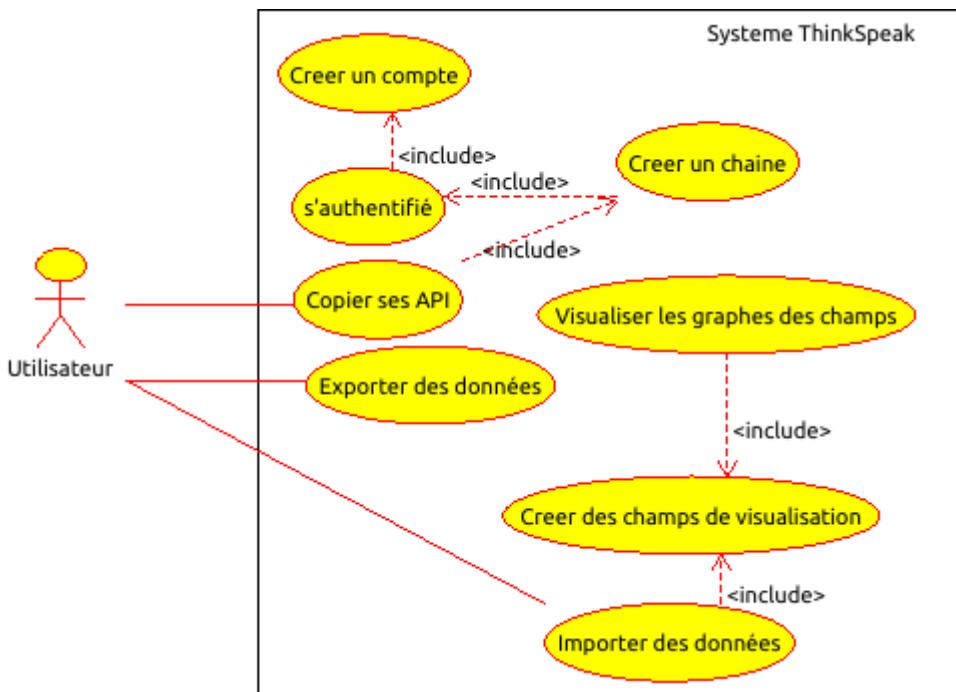


Figure 23 : Diagramme cas d'utilisation ThinkSpeak

Nous avons plus haut les opérations possibles ainsi que les relations entre elles; dans le cadre qui nous intéresse, les données sont stockées dans les différents champs de notre chaîne. Pour pouvoir y avoir accès, il faut s'authentifier, créer une chaîne et définir des champs ou seront stockées les données.

## V. IMPLEMENTATION

Une fois que la modélisation de l'infrastructure a été effectuée nous allons nous intéresser à la réalisation de cette dernière. Nous allons débuter par l'implémentation de la solution sur les outils électronique, puis nous allons nous intéresser au côté logiciel et ensuite on va regarder le coté fonctionnel et enfin on va voir le cout et le déploiement de la solution.

### 1. Implémentation du Réseau de capteurs

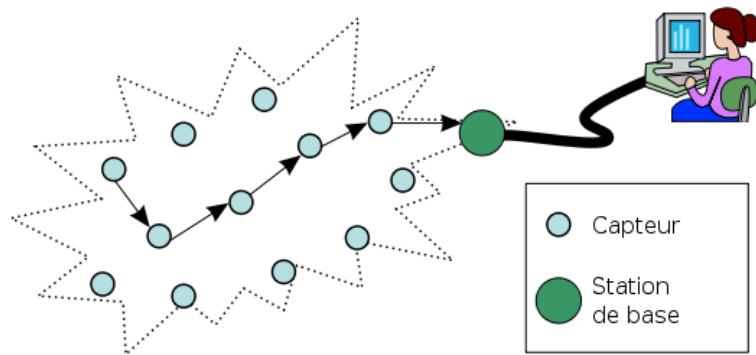


Figure 24 : Réseau de capteurs

Selon [11] « Un réseau de capteurs sans fil est un réseau d'un grand nombre de nœuds, qui sont des micro-capteurs capables de recueillir et de transmettre des données d'une manière autonome. »

Dans notre solution, pour étudier un cas de figure nous avons choisi travailler avec deux capteurs :

- Un capteur Humidité / Température ;
- Un capteur de lumière.

Nous allons étudier séparément les caractéristiques des différents capteurs avant de voir leur utilisation dans notre réseau :

#### a. Capteur d'humidité Température

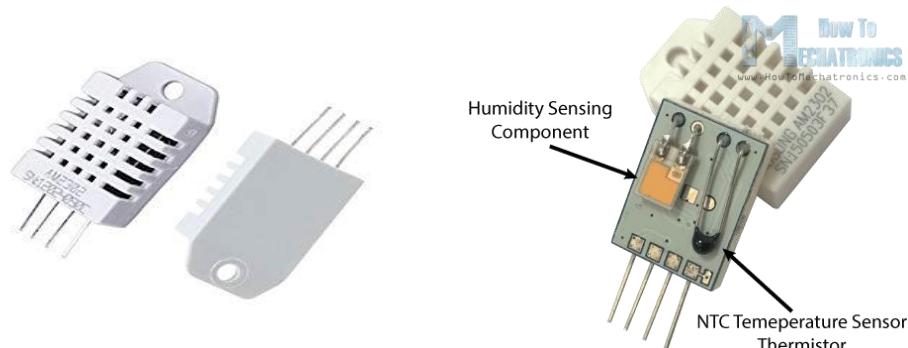


Figure 25 : Capteur d'Humidité température

Le capteur de température et d'humidité DHT22 (ou RHT03) communique avec un microcontrôleur via un port série. Le capteur est calibré et ne nécessite pas de composants supplémentaires pour pouvoir être utilisé. Ses caractéristiques sont les suivantes :

- Alimentation: 3,3 à 6 Vcc
- Consommation maxi: 1,5 mA
- Consommation au repos: 50 µA
- Plage de mesure:
  - o température: -40 à +80 °C
  - o humidité: 0 à 100 % RH
  - o Précision:

- température:  $\pm 0,5$  °C
- humidité:  $\pm 2$  % RH
- Dimensions: 25 x 15 x 9 mm

Ces informations sont disponibles sur le site d'achat en ligne [13] et permet d'avoir sa fiche technique :

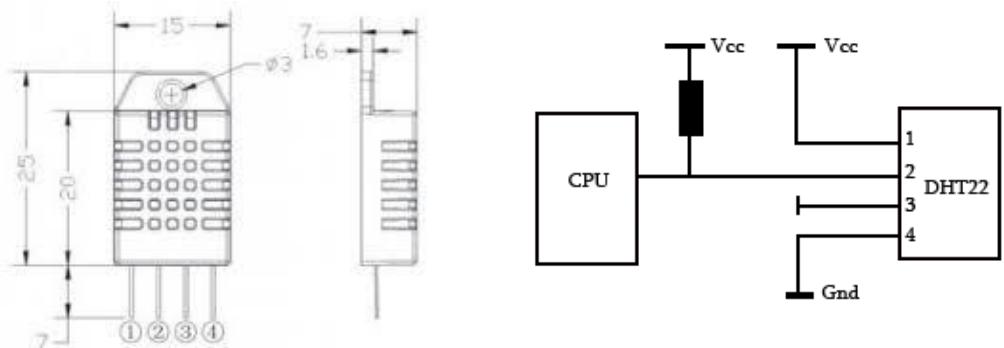


Figure 26 : Branchement d'un DHT22

Le premier pin est destiné à l'alimentation généralement elle peut varier de 3 à 6V ; Puis nous avons le second qui représente la sortie, c'est ce pin qui va nous renvoyer les données qui vont nous donner la température et l'humidité, il est lié à un résistor en parallèle pour ne pas endommager le périphérique de récupération. Ensuite nous avons le quatrième Pin qui est relié à la Terre (GROUND). Le troisième n'est pas utilisé pour notre usage.

Règles d'interprétation des données récupérées par le capteur sont définie selon le tableau ci-dessous :

Humidité								Température								Checksum			
A				B				C				D				E			
x	x	x	x	x	x	x	x	x	x	x	x	x	x	y	y	y	y	y	z

Tableau 55 : Format des données du DHT22

Le capteur lors qu'il renvoie les informations de récupération, elle les envoie sous forme d'une suite de 40 bits :

- **Humidité** : représente la suite les 16 premiers bits récupérés ; pour avoir la valeur de l'humidité en pourcentage on converti la série de « x » en décimal puis on divise par 10 ;
- **Température** : représente la suite les 16 second bits récupérés ; pour avoir la valeur de la température en Celsius, on converti la série de « y » en décimal puis on divise par 10 ;
- **Checksum (Somme de contrôle)** : représente la suite les 8 derniers bits récupérés ; pour vérifier si la suite des bits est valide on addition les séries binaires A, B, C et D. Si cette somme correspond les données sont validées sinon elles sont rejetées.

## Procédure de récupération des données :

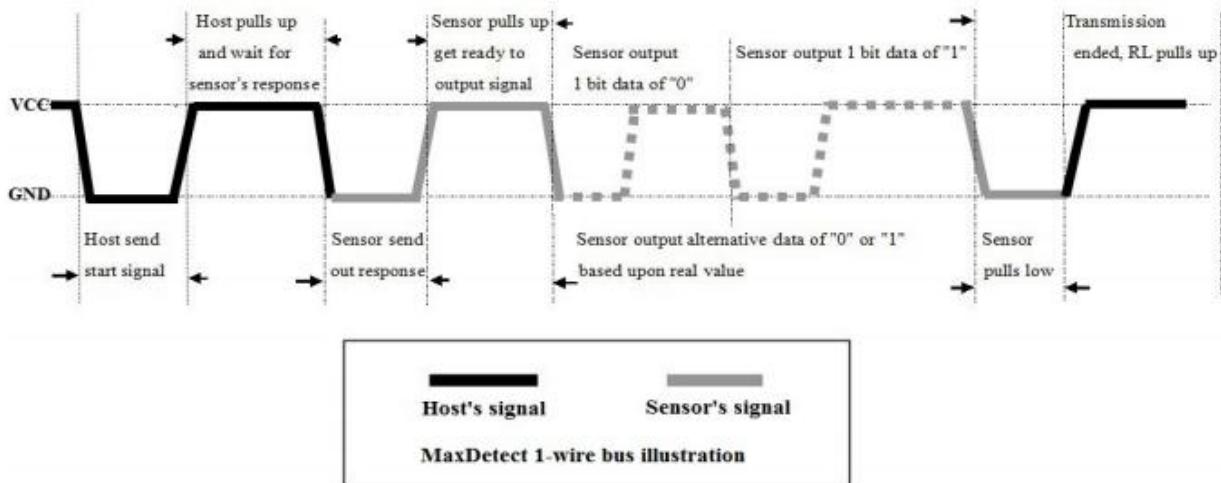


Figure 27 : Préparer la récupération

Lorsque le signal de démarrage de l'unité MCU est envoyé, DHT22 passe de l'état de veille à l'état de fonctionnement. Lorsque MCU termine l'envoi le signal de démarrage, DHT22 enverra un signal de réponse de données de 40 bits qui reflètent l'humidité relative et la température à MCU. Sans le signal de démarrage de MCU, DHT22 ne donnera pas de signal de réponse au MCU. Un signal de départ pour un les données de réponse de RHT03 qui reflètent l'humidité relative et la température. RHT03 passera en mode veille état lorsque la collecte de données est terminée si elle ne reçoit pas de nouveau le signal de démarrage du MCU.

Pour tirer court, la récupération des données venant de ce capteur se passe en un temps très réduit et l'intervalle entre deux bits étant de l'ordre d'une dizaine de nanoseconde. Il faut donc des outils de bas niveau pour manipuler à la fois ce composant physique mais aussi qui puisse manipuler le temps avec une telle précision.

### b. Capteur de luminosité

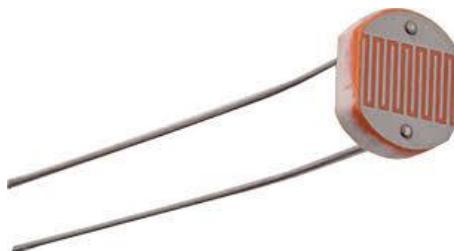


Figure 28 : Capteur de luminosité

Ce capteur se base donc sur la luminosité et l'unité correspondant à cette dite luminosité se nomme LUX. Voici quelques exemples de comparaison.

Exemples	Moyenne éclairement
Nuit de pleine lune	0.5 lux
Rue bien éclairée (nuit)	Entre 20 et 70 lux
Extérieur par ciel couvert	Entre 500 et 25 000 lux
Extérieur en plein soleil	Entre 50 000 et 100 000 lux

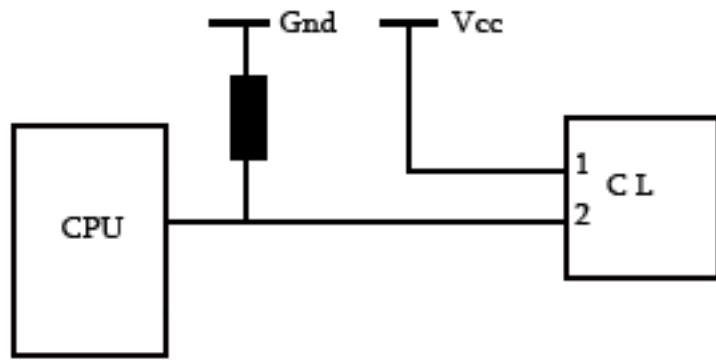


Figure 29 : Schéma électronique du capteur de luminosité

Le premier pin est généralement destiné à l'alimentation, elle peut varier de 3 à 5V ; Puis nous avons le second qui représente la sortie, c'est ce pin qui va nous renvoyer les données qui vont nous donner la valeur correspondant au niveau de luminosité, il est lié à un résistor en parallèle pour ne pas endommager le périphérique de récupération.

### c. Schéma électrique du réseau de nos capteurs

Une fois que nous avons vu séparément chaque capteur on peut voir comment associer ces deux pour qu'elles puissent fonctionner ensemble.

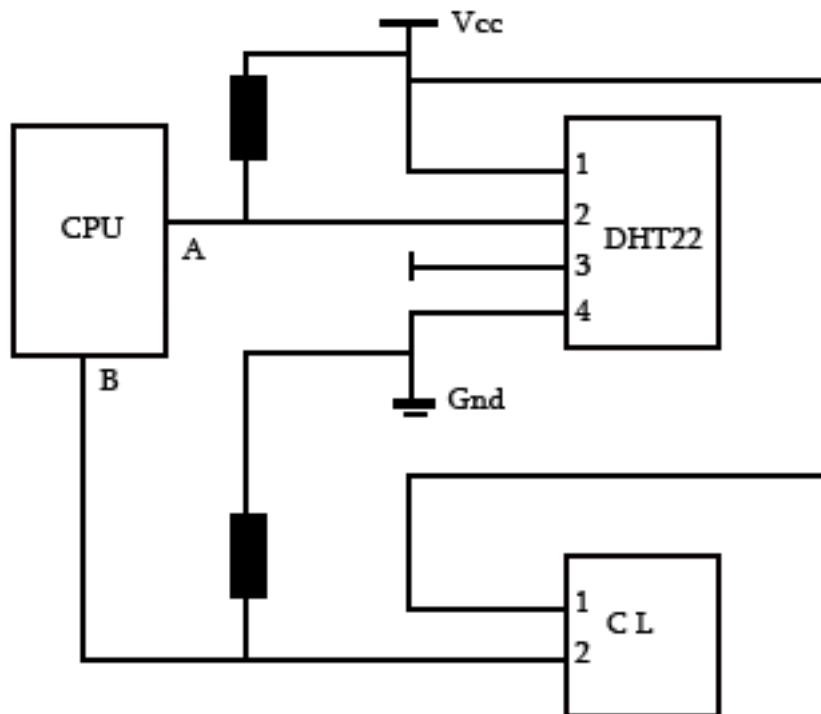


Figure 30 : Schéma électrique de notre réseau de capteurs

Nous n'allons pas revenir sur toutes les explications, on peut juste dire qu'on branche les deux équipements en parallèle pour qu'ils aient la même tension d'alimentation. Puis nous branchons ces fils à deux Pins (GPIO) de notre Raspberry.

## 2. Implémentation du Raspberry

Nous allons expliquer comment les branchements physiques vont s'effectuer puis nous allons voir comment la connexion à le Raspberry s'effectue et enfin nous verrons comment travailler avec le Raspberry.

### a. Connexion au réseau de capteurs

Suivant le schéma précédent, nous allons expliquer comment le Raspberry va intervenir pour alimenter et traiter les informations des différents capteurs.

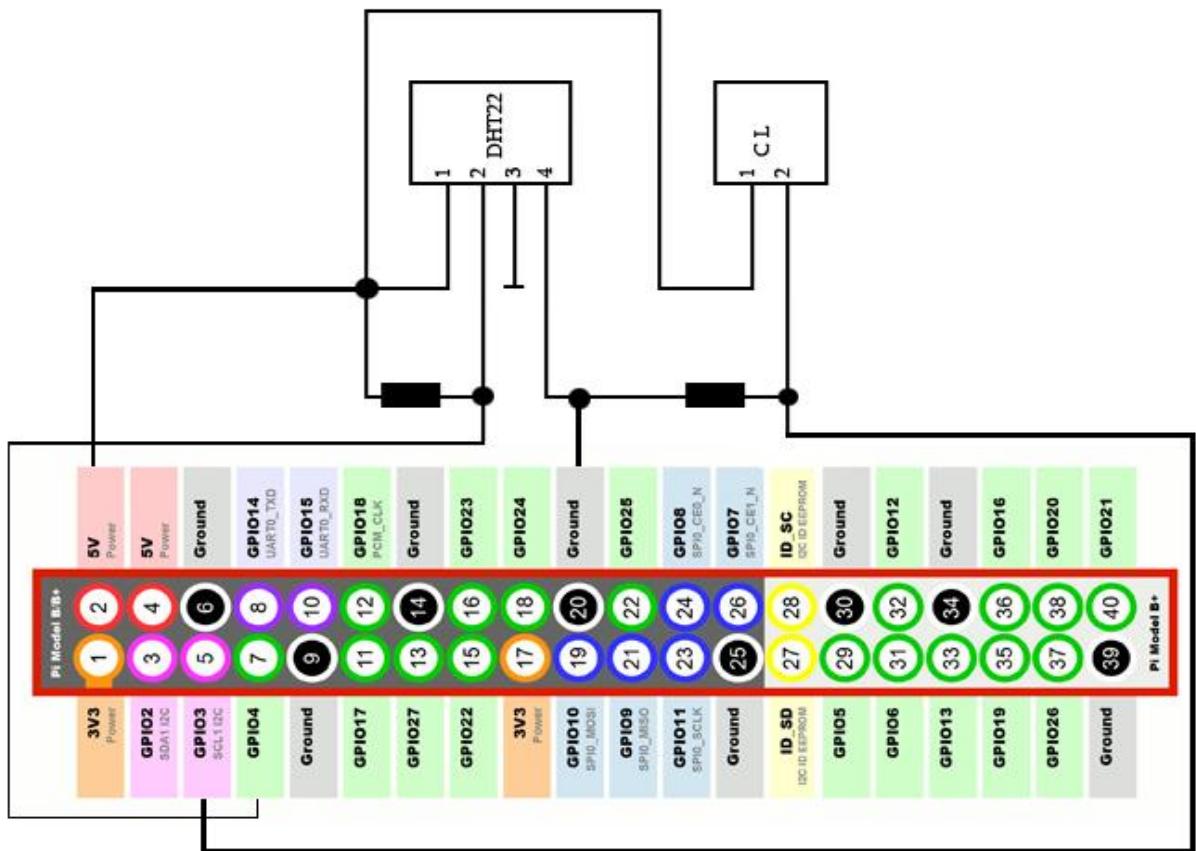


Figure 31 : Connexion Réseau de capteurs à le Raspberry

On branche l'alimentation du circuit en 5v, puis on branche la terre à un Pin GROUND du Raspberry. Pour la récupération des données on configure les Pins GPIO3 et GPIO4 en input puis on branche respectivement le capteur humidité/Température et le capteur de lumière au Pin GPIO3 et GPIO4.

### b. Connexion à la station de contrôle

Pour interconnecter la station de contrôle à le Raspberry on peut utiliser deux méthodes ; soit une connexion Wifi, soit une connexion Ethernet. Pour effectuer cette configuration il faut disposer soit :

- D'un clavier et un écran avec les câbles qui vont avec ;
- D'un ordinateur secondaire et d'un port carte, pour insérer le support mémoire du Raspberry et la modifier.

Nous allons prendre le cas où l'on dispose des périphériques nécessaires pour interagir directement avec le Raspberry. Les étapes sont donc les suivantes :

- S'identifier : dès que l'on a allumé le Raspberry on tombe sur l'identification. Les identifiants par défaut sont : **{password='raspberry', login : 'pi'}**
- Configurer le Raspberry pour quelle puisse se connecter à un Wifi existant :

On rappelle que cette fonctionnalité n'est ajoutée qu'à partir du modèle 3B. On tape la commande ci-dessous au terminale :

```
$ sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

Cette commande nous permet de modifier le fichier contenant les accès statiques vers un point d'accès wifi.

```
network= {
    ssid="SSID_POINT_ACES"
    psk="YOUR_NETWORK_PASSWORD_POINT_ACES"

    prot=RSN
    key_mgmt=WPA-PSK
    auth_alg=OPEN
}
```

- **Ssid** sera le nom de votre Wifi, vous remplacez **SSID\_POINT\_ACES** par ce nom ;
  - **Psk** représente le mot de passe de ce Wifi ;
  - **proto** pourrait être soit RSN(**WPA2**) ou WPA(**WPA1**) ;
  - **key\_mgmt** pourrait être soit **WPA-PSK** (très probablement) ou **WPA-EAP** (réseaux d'entreprise) ;
  - **pairwise** pourrait être soit CCMP(**WPA2**) ou TKIP(**WPA1**) ;
  - **auth\_alg** est le plus probablement **OPEN**, d'autres options sont **LEAP** et **SHARED**.
- Configurer les adresses statiques sur les interfaces réseau Wlan et l'Ethernet :

On rappelle que cette fonctionnalité n'est ajoutée qu'à partir du modèle 3B. On tape la commande ci-dessous au terminale :

```
$ sudo nano /etc/network/interfaces
```

Cette commande nous permet de modifier le fichier contenant les accès statiques vers un point d'accès wifi.

```
iface lo eth0 loopback
iface wlan0 inet static
    address 192.168.1.241
    netmask 255.255.255.0
    gateway 192.168.1.1

    allow-hotplug wlan0
    iface wlan0 inet static
        address 192.168.1.155
        netmask 255.255.255.0
        gateway 192.168.1.1
    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

Cette configuration nous permet de fixer les adresses sur les futures connexions sur les interfaces Wlan0 et Eth0 ; ainsi on n'aura pas à changer d'adresse à chaque connexion. Il ne reste plus qu'à redémarrer le Raspberry et la connecter au type de réseau que la machine de la station de contrôle utilise.

- Prendre le contrôle du Raspberry. Une fois que la station de contrôle est dans le même réseau que le Raspberry on se connecte par SSH.

```
$ sudo ssh pi@IP_ADRESSE
```

- **Pi** : représente l'utilisateur par défaut du Raspberry, son mot de passe est « raspberry » ;
- **IP\_Adresse** : est l'adresse de l'interface à travers laquelle on se connecte sur le Raspberry.

### c. Installation des outils

La plupart nécessite une connexion internet, c'est la raison pour laquelle le modèle B est conseillé. On va d'abord installer les bibliothèques indispensables :

- **Wiring Pi** : « **WiringPi** est une bibliothèque d'accès GPIO basée sur un **code PIN** et écrite en C pour les dispositifs SoC BCM2835, BCM2836 et BCM2837 utilisés dans tous les **Raspberry Pi...** Et est destiné à être utilisé par des programmeurs C / C ++ expérimentés. Ce n'est pas un outil d'apprentissage novice. » [13].

```
$ git clone git://git.drogon.net/wiringPi
$ cd ~/wiringPi
$ git pull origin
$ cd ~/wiringPi
$ ./build
```

Pour pallier à ce problème de connexion le télécharge à partir de Git puis on le copie dans le Raspberry et enfin on fait un build.

- **PharoThing** : « Plate-forme de programmation en direct pour les projets basés sur Pharo . Il comprend des outils de développement pour programmer, explorer et déboguer des cartes distantes (basé sur TelePharo ) qui utilise la bibliothèque de modélisation de carte qui simplifie la configuration de la carte Raspberry piloté par la bibliothèque WiringPi» [15]

```
$ wget http://files.pharo.org/vm/pharo-spur32/linux/armv6/latest.zip  
$ unzip lastest.zip  
$ cd lastest  
$ ./pharo --headless Server.image remotePharo --
```

On télécharge la dernière version de Pharo (compatible avec le processeur des Raspberry), puis on l'ouvre et on lance Pharo puis on ouvre un playground et on colle le code ci-dessous et on le lance :

```
Metacello new  
baseline: 'PharoThings';  
repository: 'github://pharo-iot/PharoThings/src';  
load: #(RemoteDevServer Raspberry).
```

*Code 1 : Installation de PharoThing Server*

Puis on fait un « save and quit » et on relance pharo maintenant sous forme de server avec un port (dans ce cas 40423) défini de connexion.

```
./pharo --headless Server.image remotePharo --startServerOnPort=40423
```

Maintenant que l'on a configuré le Raspberry comme server on installe des outils sur la partie client de la station de contrôle.

- On installe aussi Pharo puis on lance et on installe PharoThing côté Client en tapant le code ci-dessous dans le playground :

```
Metacello new  
baseline: 'PharoThings';  
repository: 'github://pharo-iot/PharoThings/src';  
load: #(RemoteDev).
```

*Code 2 : Installation de PharoThing Client*

- On connecte le client au serveur, considérons que l'adresse du serveur est « 192.168.1.242 » ; on se connecte donc de la façon qui suit :

```
remotePharo := TlpRemoteIDE connectTo: (TCPAddress ip: #[192 168 1 241]  
remoteBoard := remotePharo evaluate: [ RpiBoardBRev1 current].  
remoteBoard inspect port: 40423)
```

*Code 3 : Connexion à PharoThing Server*

On se connecte au serveur en notant l'adresse IP et le port utilisé puis on exécute ; on devrait avoir résultat comme celui-ci. Dessus on peut voir l'emplacement des différents PINs en fonction du modèle utilisé et de l'implémentation disponible. Et si les Pin son bien configuré on peut voir l'état des Pin directement et modifier la valeur de certains qui sont configuré en Input.

The screenshot shows a window titled "Inspector on a PotRemoteBoard (a RpiBoardBRev1 in #[169 254 0 2]:40423)". Below the title bar are tabs: P1, Devices, Raw, and Meta. The P1 tab is selected, displaying a table of pins:

P1	Devices	Raw	Meta				
Id	Value	Name	Pin#	Pin#	Name	Value	Id
		3.3v	1	2	5v		
0		SDA (I2C)	3	4	5v		
1		SCL (I2C)	5	6	Ground (0v)		
4		GPIO7	7	8	SerialPortTXD	14	
		Ground (0v)	9	10	SerialPortRXD	15	
17		GPIO0	11	12	GPIO1	18	
21		GPIO2	13	14	Ground (0v)		
22	in	GPIO3	15	16	GPIO4	out	23
		3.3v	17	18	GPIO5		24
10		MOSI (SPI)	19	20	Ground (0v)		
9		MISO (SPI)	21	22	GPIO6		25
11		SCLK (SPI)	23	24	CE (SPI)		8
		Ground (0v)	25	26	CE (SPI)		7

Below the table is a code editor containing the following Pharo code:

```

" a PotBoardConnector(P1): gpio0..gpio7 vars are bound to pins"
led := gpio4.
led beDigitalOutput.
led value: 1.
led value: 0.

button := gpio3.
button beDigitalInput. "button"
button enablePullDownResister.
button value.

buttonProcess := [ [100 milliseconds wait.
    led value: (button value=1) asBit
] repeat
1 fork.

```

Figure 32 : Interface de commande de PharoThing Server à partir du Client

#### d. Interaction avec les capteurs

On a vu plus haut les détails sur les capteurs que l'on a utilisés. Comme nous travaillons avec les bits, on peut remarquer que le capteur de luminosité sera purement binaire, c'est-à-dire 1 pour la présence d'une lumière et 0 sinon. Ce qui fait que c'est assez

évident de connaitre son état. Nous allons nous concentrer plus sur le capteur Humidité/Température.

Le capteur DHT22 renvoie les données à la nanoseconde près, donc nous avons décidé d'utiliser directement du code C pour la rapidité, pour son caractère bas niveau et plus important car la bibliothèque principale WiringPi est développée en C.

```
#include <wiringPi.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#define MAX_TIMINGS      85
#define DHT_PIN           4          /* GPIO-4 */
int data[5] = { 0, 0, 0, 0, 0 };
void read_dht_data()
{
    uint8_t laststate      = HIGH;
    uint8_t counter        = 0;
    uint8_t j = 0, i;
    data[0] = data[1] = data[2] = data[3] = data[4] = 0;
    /* pull pin down for 18 milliseconds */
    pinMode( DHT_PIN, OUTPUT );
    digitalWrite( DHT_PIN, LOW );
    delay( 18 );

    /* prepare to read the pin */
    pinMode( DHT_PIN, INPUT );

    /* detect change and read data */
    for( i = 0; i < MAX_TIMINGS; i++ )
    {
        counter = 0;
        while( digitalRead( DHT_PIN ) == laststate )
        {
            counter++;
            delayMicroseconds( 1 );
            if( counter == 255 )
                break;
        }
        laststate = digitalRead( DHT_PIN );

        if ( counter == 255 )
            break;

        /* ignore first 3 transitions */
        if ( (i >= 4) && (i % 2 == 0) )
        {
            /* shove each bit into the storage bytes */
            data[j / 8] <= 1;
            if ( counter > 16 )
                data[j / 8] |= 1;
            j++;
        }
    }
}
```

Code 4 : Récupération des 40bits du capteur DHT22

On suit la procédure suivante :

- On sort le capteur du mode veille en lui envoyant un signal, puis on attend 18 nano secondes et on configure le PIN en entrée ;
- On récupère au fur et à mesure tous les bits à intervalle d'une Milli Seconde ;
- A chaque série de 8 bits, on ne garde que la valeur décimale ainsi à la fin on se retrouve avec A, B, D, C et E telle qu'expliqué plus haut ;
- Pour valider les données on vérifie le checksum  $E = A+B+C+D$  ;
- Puis pour trouver l'Humidité on fait  $(AB)_{10}/10$  ;
- Puis pour trouver la température on fait  $(CD)_{10}/10$ .

Une que les données sont récupérées il ne reste qu'à les stocker ou les envoyer en ligne.

### 3. Configuration sur ThinkSpeak

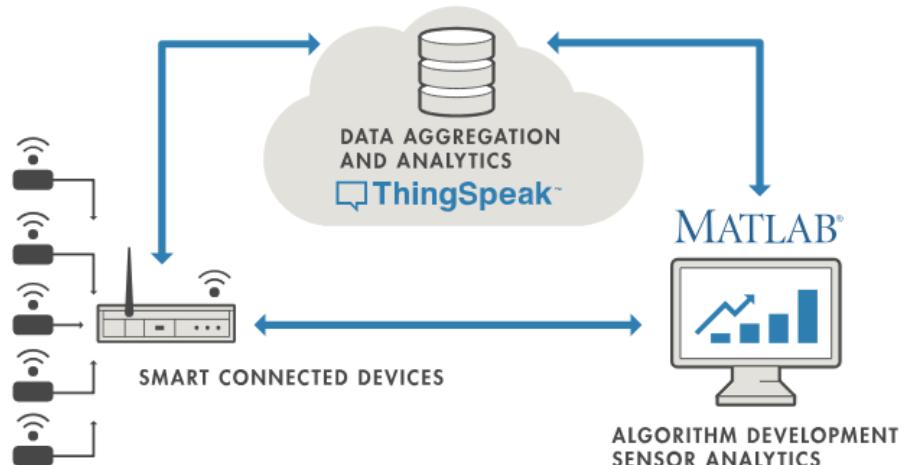


Figure 33 : Système de ThinkSpeak

La plateforme ThinkSpeak est une plateforme externe et nous offre des services de Cloud en termes de stockage en ligne. Nous allons suivre la procédure suivante :

- Créer un compte personnel sur la plateforme en utilisant ce lien : [https://thingspeak.com/users/sign\\_up](https://thingspeak.com/users/sign_up);
- Puis créer une chaîne de diffusion, puis copier les accès.

Pour les besoins du projet nous avons créé une chaîne publique

Nom	Description	Méthode	url ou clé
Accès en écriture	Permet d'envoyer des données en ligne	x	784TO4HJBKZBX9T8
Accès en lecture	Permet de lire ce qui se trouve en ligne lorsque la chaîne est privée		B6HWAE381748SJ4X

<b>Identifiant</b>	Permet d'identifier la chaîne lors d'un envoi de données		514579
<b>Api d'écriture</b>	On envoie la valeur 0 au champ <b>field1</b> de notre chaîne de diffusion	GET	<a href="https://api.thingspeak.com/update?api_key=784TO4HJBKZBX9T8&amp;field1=0">https://api.thingspeak.com/update ?api_key=784TO4HJBKZBX9T8 &amp;field1=0</a>
<b>Api de lecture Feed</b>	On récupère les informations sur la chaîne ainsi que les deux premières valeurs de chaque champ	GET	<a href="https://api.thingspeak.com/channels/514579/feeds.json?results=2">https://api.thingspeak.com/channels/ 514579/feeds.json?results=2</a>
<b>Api de lecture Field</b>	On récupère les deux premières données du champ <b>field1</b>	GET	<a href="https://api.thingspeak.com/channels/514579/fields/1.json?results=2">https://api.thingspeak.com/channels/ 514579/fields/1.json?results=2</a>

#### 4. Implémentation de l'application Mobile



Figure 34 : Logo Android Studio

Cet application a pour principale rôle le suivi des données en ligne et l'envoie des données manuelles sur le Cloud en ligne. Elle a été développée pour s'adapter aux API de ThinkSpeak et son nom est **SDA Mobile**.

Nous avons utilisé pour le développement l'environnement de développement Android Studio qui est l'éditeur officiel pour le développement des applications natives en Android. Nous allons nous servir du schéma architectural suivant pour expliquer comment son implémentation a été effectuée.

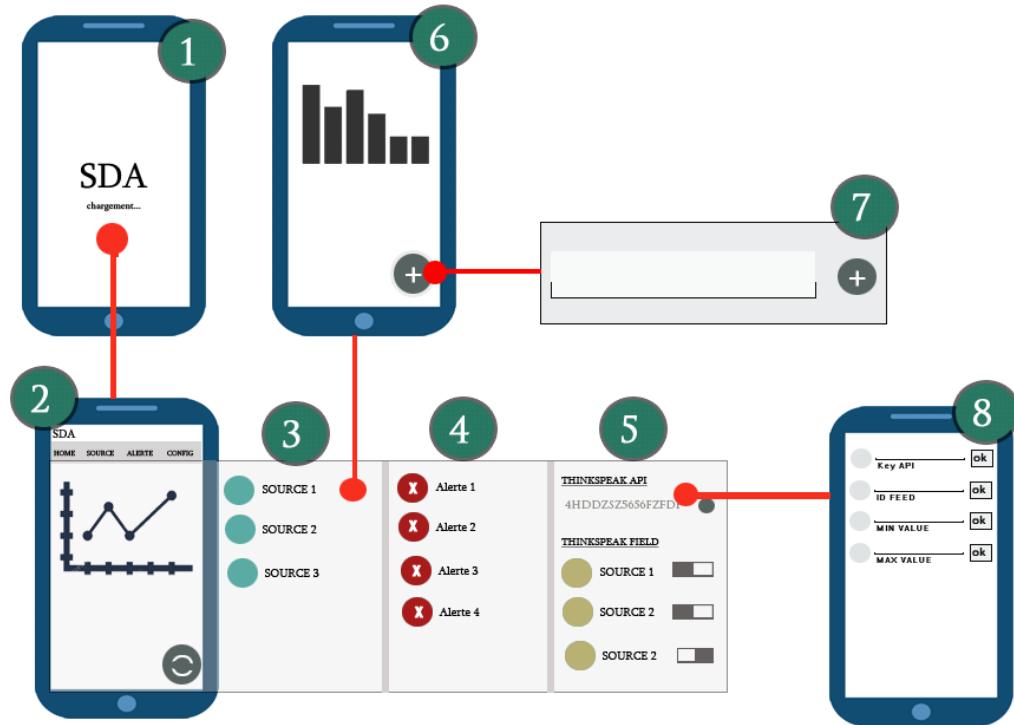


Figure 35 : Schéma de l'application Mobile

L'application a quatre activités (Activity est le métier de l'application et possède généralement une View au minimum) principales :

- **InitActivity** : cette activité correspond au **numéro 1**. C'est cette activité qui permet d'initialiser l'application à chaque démarrage et charge les données qui ont été modifiées et qui seront utilisées sur les autres activités. Elle utilise pour cela une connexion via le protocole http ; Une fois que les données ont été chargées elle appelle l'activité MainActivity ;

```
Intent intent = new Intent(this, MainActivity.class);
this.startActivity(intent);
```

Code 5 : Appel d'une Activité Android

L'appel des activités sur Android n'est pas impératif; on soumet une intention (un souhait) au système et si celui-ci peut fournir faire appel à l'activité, elle le fait.

- **MainActivity** : Cette activité est composée de plusieurs vues, générées par un **ViewPager** (faire coulisser les vue du doigt sous forme de page défilantes ou Slide) combinée aux **TabLayout**. Ainsi on a les sous métiers suivant :

- **La vue numéro 2 :** cette page utilise une bibliothèque qui permet de pouvoir faire appel à des **LineChart** et d'autres types de graphes. On fait appel à cette bibliothèque en ajoutant cette dépendance :

```
implementation 'com.github.PhilJay:MPAndroidChart:v3.0.3'
```

*Code 6 : Ajout des dépendances pour les Graphes Android*

- **La vue numéro 3 :** cette liste présente les différents champs de notre chaîne configurée. On utilise ici des **RecyclerView** pour produire ces listes. Et lorsque l'on choisit un champ on est dirigé vers l'activité **SourceDetailActivity** et cette fois on ajoute un paramètre lors de l'appel:

```
Intent intent = new Intent(this, SourceDetailActivity.class);
intent.putExtra("item_id", id);
this.startActivity(intent);
```

*Code 7 : Appel d'une activité avec paramètre*

- **La vue numéro 4 :** cette liste des alertes soulevées sur les données du champ actif ;
- **La vue numéro 5 :** Cette vue permet de choisir le champ principal, mais aussi permet de configurer les accès en ouvrant l'activité **ConfigActivity**.

## 5. Implémentation de la plateforme Web

Notre plateforme web a pour principal rôle d'effectuer le traitement et l'analyse des données. Elle est essentiellement développée en Pharo. Pour mieux expliquer l'implémentation nous n'allons pas nous intéresser aux différentes classes parmi la multitude que compte cette plateforme. Nous allons voir les dépendances de ce projet et leur installation, puis nous allons parler des différentes classes de la couche vue et voir quelques implémentations phares.

### a. Installation de SeaSide



*Figure 36 : Logo SeaSide*

SeaSide fournit un ensemble d'abstractions superposées sur HTTP et HTML qui vous permettent de créer des applications Web hautement interactives rapidement, réutilisables et maintenables. Il est basé sur Smalltalk, un langage éprouvé et robuste qui est implémenté par différents fournisseurs [16]. Elle comprend des outils comme :

- **Un Générateur de HTML** : SeaSide a une API riche pour générer du HTML qui vous permet d'abstraire ces modèles dans des méthodes pratiques plutôt que de coller la même séquence de balises dans des modèles à chaque fois.

```
Seaside
html div
    id:'identifiant';
    class:'les classes';
    with: 'Hello Word!'.

-----
Html
<div id="identifiant" class=="les classes" >
Hello Word!
</div>
```

Code 8 : Génération du code HTML SeaSide

On gère les balises du HTML ici, comme étant des méthodes d'un objet dont html est une instance. Ceci permet d'obtenir à la fin un code modulaire et d'y appliquer des notions orientées objet avancées pour optimiser le rendu.

- **Gestion des requêtes basée sur les Callbacks** : SeaSide ne complique plus avec l'usage des Url pour les liens et des **names** pour les champs des formulaires. On fait juste appel à une méthode métier et un lien est généré automatiquement.

```
Seaside
html anchor
    callback: [self fonction];
    with: 'Hello Word!'.

-----
Html
<a href="url générée" >
Hello Word!
</a>
```

Code 9 : SeaSide avec les appels Callback

- **Composants incorporés** : Dans cette vision on ne travaille pas sur tout le rendu de la vue à la fois ; on les travaille sous forme de composantes que l'on met ensemble pour générer le rendu final.

SeaSide comporte d'énormes points positifs que nous n'allons pas citer ; pour son installation on doit exécuter le code suivant dans son playground :

```

Metacello new
    configuration:'Seaside3';
    repository:
'http://www.smalltalkhub.com/mc/Seaside/MetacelloConfigurations/main';
    version: #stable;
load

```

*Code 10 : Installation de SeaSide Stable*

NB : Toutes les classes qui sont sur la couche vu de notre application d'une classe SeaSide **WPComponent** : il s'agit du composant principal ayant de nombreuses méthodes permettant de gérer l'aspect Vue.

### b. Installation de Bootstrap

Il s'agit d'une bibliothèque très utilisée dans le monde du développement, elle permet de développer de manière ergonomique et ce très facilement. Généralement il est intégré à SeaSide, mais dans le cas contraire il faut l'installer :

```

Metacello new
    baseline: 'PharoBootstrapProcess';
    repository: 'filetree://.';
load.

```

*Code 11 : Bootstrap Pharo*

Cette version de Bootstrap est adaptée à la vision de Pharo, car associée à Pharo elle permet de manipuler les balises et les facilités qu'offre Bootstrap sous forme de méthode.

### c. Installation de DataFrame

Il s'agit d'une bibliothèque Open source développée pour Pharo afin de faciliter le traitement de données statiques.

```

Metacello new
    baseline: 'DataFrame';
    repository: 'github://PolyMathOrg/DataFrame';
load.

```

*Code 12 : Installation de DataFrame*

#### a. Installation de HighChart

Il s'agit d'une bibliothèque Open source développée pour Pharo afin de faciliter l'intégration de nombreux outils graphiques, permettant une représentation statistique des données.

```
Metacello new  
  baseline: 'HighchartsSt';  
  repository: 'github://ba-st/HighchartsSt:master/repository';  
  load.
```

*Code 13 : Installation de HighChart*

Une fois que tous ces outils ont été installés, nous les avons utilisés pour pouvoir obtenir les résultats présentés dans le chapitre suivant.

# CHAPITRE IV : RESULTATS OBTENUS

Dans ce chapitre nous allons principalement présenter les captures d'écran des différents outils logiciels qui ont été développés. Nous allons démarrer par l'exécution du pilote des capteurs, puis nous allons présenter l'application mobile et ensuite nous allons présenter l'application Web.

## I. RESULTATS SUR L'APPLICATION MOBILE

Ici nous allons présenter les différentes interfaces et expliquer leurs rôles.

### 1. Initialisation de l'application et Accueil



Figure 37 : Initialisation de l'application

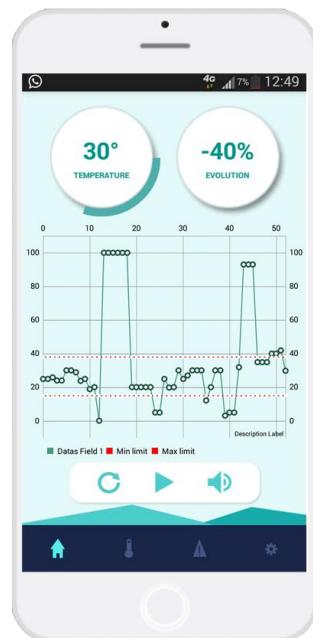


Figure 38 : Page d'accueil

Sur la figure de gauche nous avons un SplashScreen qui permet à l'utilisateur d'attendre que l'application charge les données correspondant aux API indiquées. En effet elle charge les données du champ principal configuré sur l'application, les différents champs et les informations supplémentaires sur la chaîne.

La seconde figure présente un graphe (LineChart) qui représente les données enregistrées sur ThinkSpeak du champ principal ; et en plus de cela, elle présente deux barres horizontales rouges. Ces lignes représentent les limites maximales et minimales que les données ne doivent pas dépasser. Si une donnée est au-dessus du maximum ou en dessous du minimum alors une alerte est signalée.

**NB** : cette application offre un avantage non négligeable par rapport aux graphes disponibles sur l'application. Elle permet de gérer le problème de données manquantes. Si on a N données et que la 3 ième est inexistante alors la plateforme ThinkSpeak va ignorer

les N-2 autres valeurs. Mais dans notre application on lui donne la valeur qui précède sinon la valeur 0.

Et le bouton vert permet de rafraîchir les données de l'application.

## 2. Liste des champs et des alertes

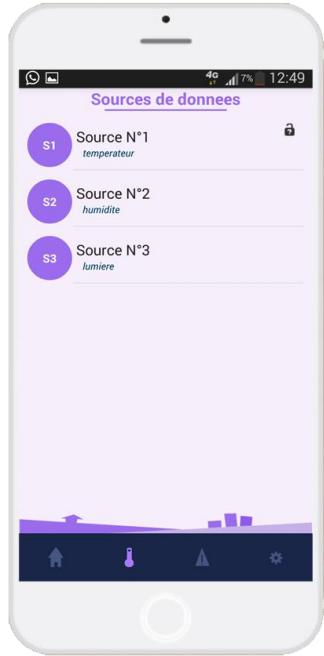


Figure 39 : Liste des champs de la chaîne



Figure 40 : Liste des Alertes Signalées

Sur la figure de gauche nous avons la liste complète des champs qui ont été créés en ligne sur notre chaîne. Le petit cadenas indique qu'il s'agit de la source de données qui est considérée comme principale parmi toutes les autres. Pour avoir les détails d'un champ il faut cliquer dessus.

Sur la figure de droite nous avons la liste complète des alertes signalées sur la source principale des données ; elle indique la valeur, le numéro de la source et une description sur l'alerte signalée.

## 3. Configuration de l'application



Figure 41 : Configuration de la source principale

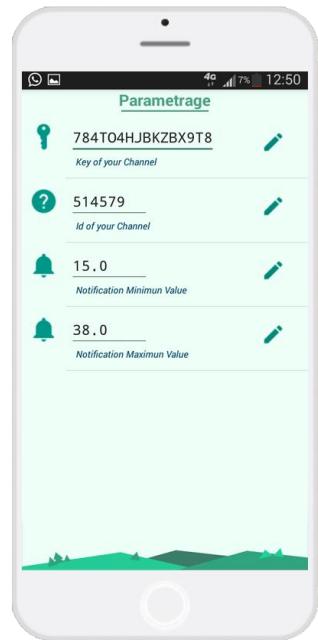


Figure 42 : Configuration des API et alertes

Sur la figure de gauche nous avons la liste complète des champs qui ont été créés en ligne sur notre chaîne. Il est question de choisir une des sources comme source principale et afficher son flux de données à la page d'accueil de l'application. Plus haut nous avons la clé de notre chaîne en ligne avec un bouton de configuration qui ouvre l'interface à gauche.

Sur la figure de droite nous configurons les clés qui donnent accès aux données de la chaîne en ligne ; et réglons les limites qui sont à l'origine des alertes.

## 1. Détail d'une source et push de données

Sur la figure de gauche donne un graphe (BarChart) représentant le flux de données de la source sélectionnée.

Sur la figure de droite nous pouvons directement ajouter des données à cette source. Elle sera horodatée directement au niveau du serveur.

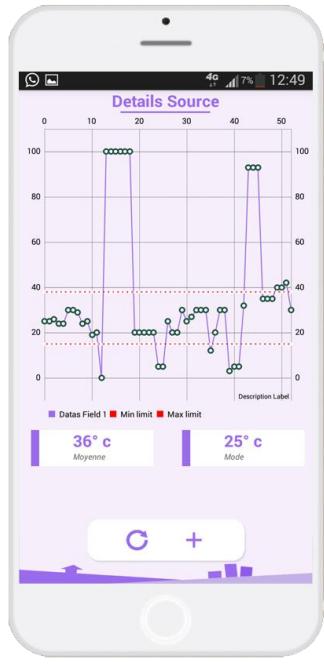


Figure 43 : Visualisation d'une source



Figure 44 : Envoie de données en ligne

## II. RESULTATS SUR L'APPLICATION WEB

Dans cette partie nous allons présenter les résultats que nous avons obtenus au niveau de notre application Pharo Web. Nous allons premièrement travailler sur l'interface Web, puis nous allons monter quelques équivalences pour l'administration de la plateforme directement depuis Pharo.

### 1. L'accueil de la plateforme web SDA

Sur l'interface ci-dessous, nous allons étudier les différentes parties :

- **La barre latérale verticale gauche (numéro 1)**: elle comporte le menu principal de notre plateforme Web. Conduisant vers les autres vues de l'application : Laboratoire, Capteur, Alerte et Analyse.
- **La vignette supérieure (numéro 2)**: elle permet de voir l'état général de la plateforme ; actuellement la plateforme compte sur son image locale : 2 laboratoires, 7 capteurs (ou sources de données), 87 données et 0 alerte signalée.
- **Les graphes (numéro 3)**: Elle permet d'avoir en une vue, une visualisation de l'ensemble des données des différentes sources. Avec chacune une couleur qui lui a été attribué à sa création. Elle a aussi un bouton à l'index droit haut, qui permet de choisir quel capteur précisément afficher, ou quels capteurs de quels laboratoires afficher.

- **Le tableau (numéro 4):** Elle permet d'avoir un tableau les capteurs qui sont représentés sur le graphe en indiquant la couleur comme une légende et d'autres informations utiles pour ces capteurs.

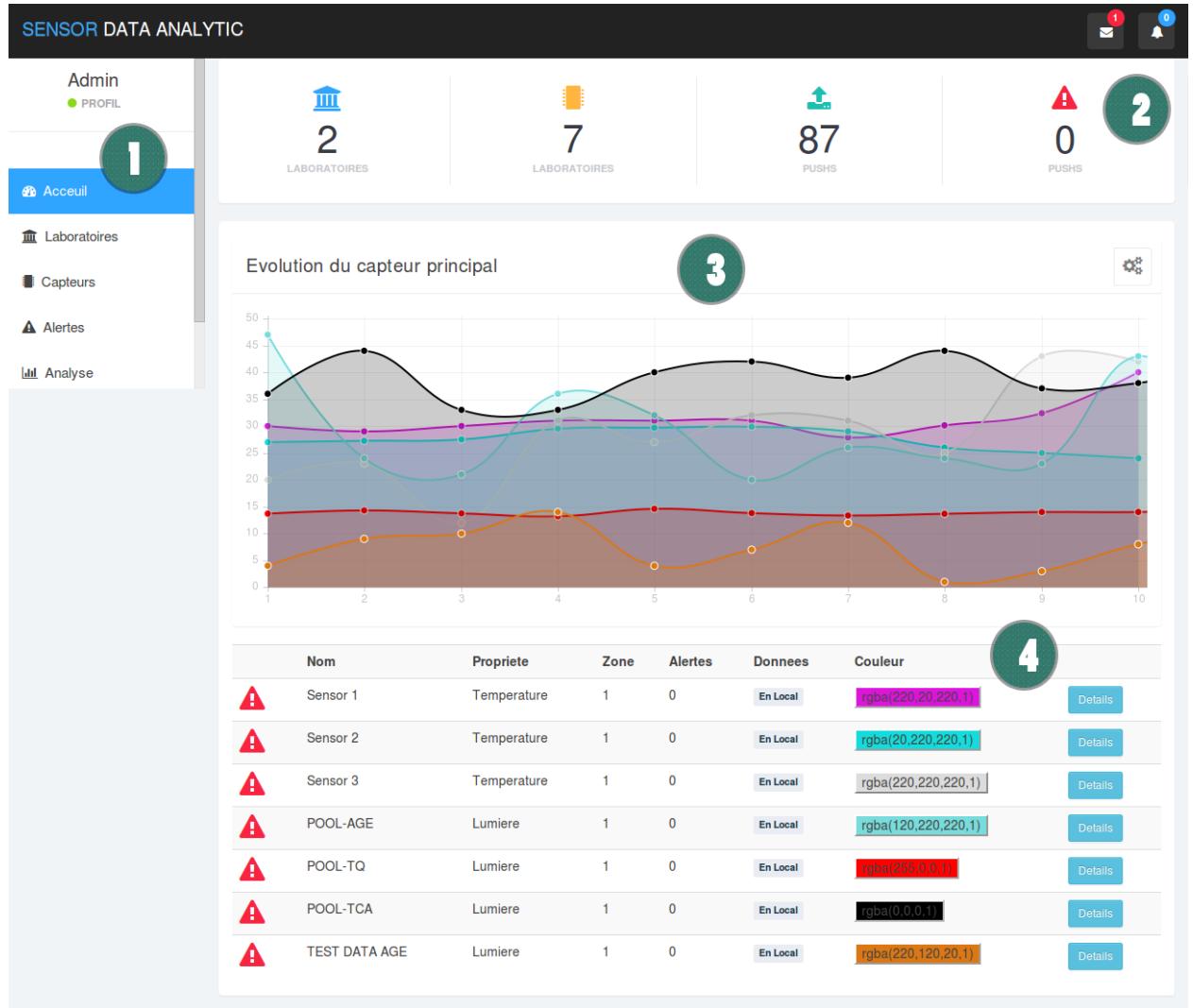


Figure 45 : Page d'accueil Web SDA

## 2. Information du propriétaire de l'image SDA

Cette page a un rôle principalement informatif. En fait tout le monde doit avoir son image pour son domaine d'observation ainsi vos données ne seront pas publiques et la configuration du profil utilisateur permet de connaître et de pouvoir contacter le propriétaire de l'image en question. Ainsi on recueille des informations comme le nom, l'email et la date de naissance.

Et si vous êtes plusieurs à la charge, et que chacun a le contrôle d'un laboratoire, vous pouvez enregistrer plusieurs, ainsi désigner qui est responsable dans tel ou tel laboratoire.

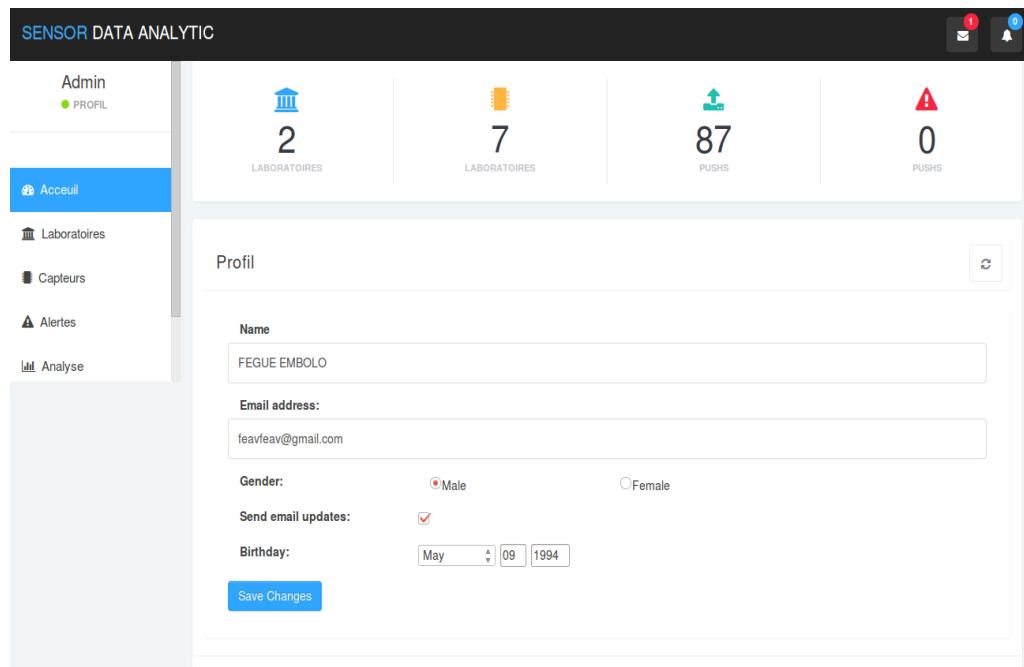


Figure 46 : Page Profil utilisateur

### 3. Le module laboratoire SDA

Dans ce module on peut effectuer plusieurs actions. Sa page contient deux onglets principaux : La liste des laboratoires et l'ajout d'un nouveau laboratoire:

#### a. Ajouter un laboratoire

Figure 47 : Ajout d'un laboratoire

Lorsqu'on appuie sur l'onglet « AJOUTER UN LABORATOIRE » on tombe sur cette page qui nous permet d'ajouter un nouveau laboratoire à notre image. On peut noter les éléments comme :

- **Le nom** : ce nom doit normalement être unique, mais dans le cas contraire ils auront toujours des identifiants différents ;
- **L'adresse** : elle permet de savoir à quel niveau le laboratoire d'expérimentation est situé ;
- **le nombre de zones** : dans les grands laboratoires on peut diviser le laboratoire en plusieurs zones. Et dans chacune des zones on peut disposer des capteurs, ceci facilite l'organisation du travail ;
- **Le responsable** : il s'agit du contact qui sera en charge du laboratoire.

## b. Lister les laboratoires

The screenshot shows the 'Sensor Data Analytic' application interface. The top navigation bar includes 'Admin' and 'PROFIL'. On the right, there are notification icons for email (1) and push notifications (0). The main dashboard features four summary cards: 'LABORATOIRES' (2), 'PUSHS' (87), and 'ALERTE' (0). The left sidebar has a navigation menu with 'Acceuil' (selected), 'Laboratoires' (selected), 'Capteurs', 'Alertes', and 'Analyse'. The main content area is titled 'Laboratoires' and contains two buttons: 'Liste des Laboratoires' (selected) and 'AJOUTER UN LABORATOIRE'. Below is a table listing two laboratories:

Nom	Adresse	Responsable	capteurs	Option
Laratoire 1	Yaounde 1 Rue 1071	feav	7	<button>Details</button> <button>Supprimer</button>
Laratoire 2	Yaounde 1 Rue 17643	armel	7	<button>Details</button> <button>Supprimer</button>

Figure 48 : Liste des laboratoires

Il s'agit d'une liste simple des différents laboratoires enregistrés dans l'image ; On peut voir leur nom l'endroit où ils se situent physiquement, le nom du responsable et le nombre de capteurs (ou source de données) qui a en son sein. On peut effectuer les opérations suivantes :

- **Supprimer un laboratoire** : ceci permet de supprimer définitivement un laboratoire, mais ses capteurs ne seront pas supprimés ;
- **Voir les détails du laboratoire** : Pour cela il suffit de cliquer sur le bouton « Détail ».

## c. Détails d'un laboratoire

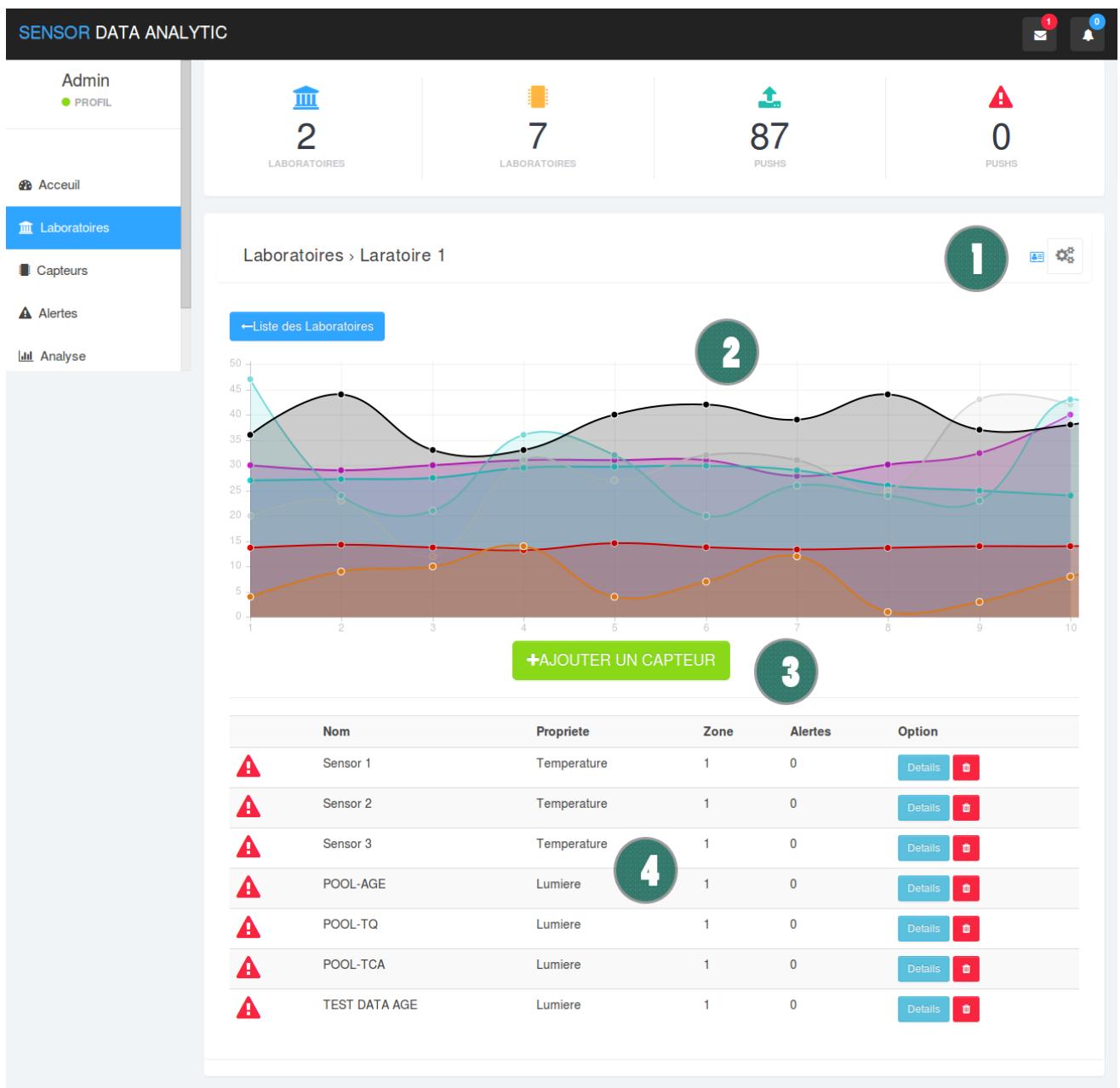


Figure 49 : Page détail d'un Laboratoire

Sur cette interface nous pouvons effectuer les actions suivantes :

- **Numéro 1** : cette partie présente deux boutons, un pour permettre de modifier les informations sur le laboratoire, et l'autre pour permettre modifier les informations qui s'affichent sur le graphe (voir seulement le graphe d'un capteur) ;
- **Numéro 2** : Permet de visualiser le flux de données des capteurs du laboratoire sélectionné ;
- **Numéro 3** : Ce bouton permet d'ajouter un capteur libre comme faisant partie du laboratoire sélectionné ;

- **Numéro 4 :** Permet de lister les capteurs qui font partie du laboratoire et affiche des informations utiles sur chaque capteur ; en plus de cela il donne la possibilité d'enlever un capteur de sa liste de capteurs.

#### d. Ajout d'un capteur à un laboratoire

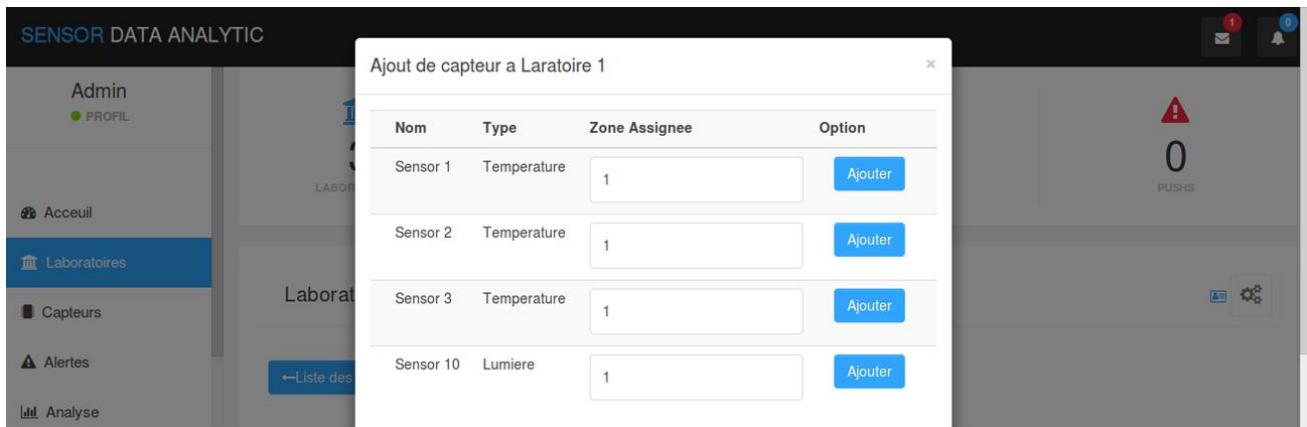


Figure 50 : Ajouter un capteur à un laboratoire

Cette figure présente juste comment ajouter un capteur ou source de données dans un laboratoire ; il est à noter qu'un capteur n'appartient qu'à un seul laboratoire, donc la liste qui s'affichera sera constitué des capteurs libres.

On choisit le capteur à ajouter puis on notifie la zone où il sera disposé.

## 4. Le module Capteur SDA

Dans ce module on peut effectuer plusieurs actions. Sa page contient deux onglets principaux : La liste des capteurs et l'ajout d'un nouveau capteur:

### a. Ajout d'un nouveau capteur

Pour ajouter un capteur il faut remplir un certain nombre d'informations :

- **Le nom du capteur :** il s'agit simplement du nom du capteur, il doit être unique ;
- **L'heure de démarrage du capteur :** c'est une information qui permet de savoir quand est ce que le capteur doit démarrer ;
- **L'heure d'extinction du capteur :** c'est l'heure à partir de laquelle on ne fait plus de prélèvements ;
- **L'intervalle :** il s'agit de l'intervalle estimé en heure du nombre d'heure entre chaque récupération de données ;
- **Les couleurs :** il s'agit des couleurs que le graphe du capteur aura pour se différencier des autres ;

- **La valeur minimale et maximale :** elle correspond aux valeurs minimales et maximales que les données doivent respecter. Dans le cas contraire une alerte est soulevée ;
- **Le type de capteur :** ici c'est le type de données, dans notre cas on a quatre types de données ; la température, l'humidité, la luminosité et les données manuelles ;
- **La clé de ThinkSpeak :** si il s'agit d'une source en ligne ;
- **L'identifiant du champ sur ThinkSpeak.**

**SENSOR DATA ANALYTIC**

Admin  
PROFIL

1 0

2 LABORATOIRES

3 LABORATOIRES

30 PUSHES

0 PUSHES

**Capteurs**

LISTE DES CAPTEURS AJOUTER UN CAPTEUR

**Nom du Capteur**  
Sensor 1

**Heure d activation du capteur**  
07:00

**Heure deactivation du capteur**  
17:00

**Nombre d heures entre chaque sonde**  
5

**Couleur primaire du capteur**  
"rgba(220,220,220,0.2)"

**Couleur secondaire du capteur**  
"rgba(220,220,220,1)"

**valeur minimale normale**  
80

**valeur maximale normale**  
80

**Nom du Capteur**  
humidite

**Inserez la clé de votre API ThinkSpeak pour synchroniser les données du web**  
B6HWAE381748SJ4X

**Inserez le champ de votre API ThinkSpeak correspondant aux données de ce capteur**  
0

Ajouter ce Laboratoire

Figure 51 : Ajouter un capteur

## b. Liste des capteurs

Nom	Actif	Free	Propriete	Signalisations	Alertes	Localisation	Option
Sensor 1	Oui	X	Temperature	0	0	En Local	<button>Details</button> <button>Supprimer</button>
Sensor 2	Oui	X	Temperature	0	0	En Local	<button>Details</button> <button>Supprimer</button>
Sensor 3	Oui	X	Temperature	0	0	En Local	<button>Details</button> <button>Supprimer</button>

Figure 52 : Liste des capteurs

Il s'agit d'une liste simple des différents capteurs enregistrés dans l'image ; On peut voir si le capteur est dans un bon état de fonctionnement ; **actif** pour savoir si ce capteur doit s'afficher sur le tableau de bord principal, le type de données que ça capte, le nombre d'alertes signalées et **Free** pour savoir si le capteur est déjà utilisé dans un laboratoire. On peut effectuer les opérations suivantes :

- **Supprimer un capteur** : ceci permet de supprimer définitivement un capteur;
- **Voir les détails du capteur**: il suffit de cliquer sur le bouton « Détail ».

## c. Détails d'un capteur

Dans la figure ci-dessous on peut voir plusieurs éléments à savoir :

- **Numéro 1** : il s'agit de certains petits rapport, sur l'évolution du capteur entre la dernière valeur et la nouvelle valeur ;
- **Numéro 2** : Représente le flux de données du capteur courant ;
- **Numéro 3** : Les informations de création du capteur sur les périodes d'allumage et sur les limites fixées;
- **Numéro 4** : On a un bouton qui permet d'ajouter un type précis d'alerte et plus bas la liste d'alertes qui ont été rajoutées.

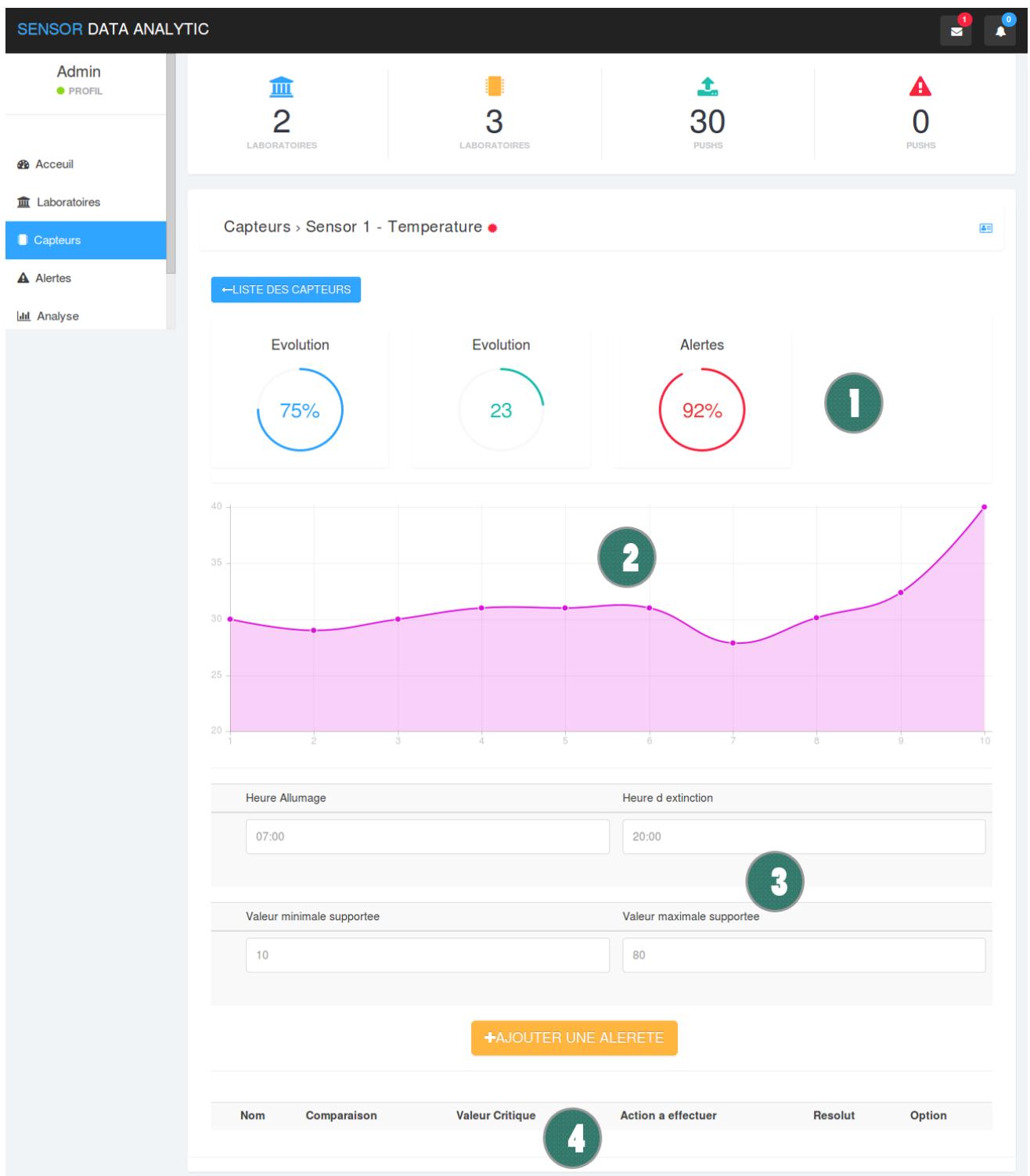


Figure 53 : Détail du capteur

## 5. Le module Alerte SDA

### a. Ajouter un modèle d'alerte

Les informations que l'on renseigne pour créer un modèle d'alerte sont :

- **La comparaison** : Il s'agit de l'opérateur de comparaison utilisé pour statuer sur une valeur ;
- **Valeur critique** : il s'agit de la valeur utilisée lors de la comparaison ;
- **Action à effectuer** : ici il s'agit d'une action que l'on doit effectuer lorsque cette alerte est signalée; on peut soit **Notifier le responsable**, soit **Éteindre le capteur**, soit **ne rien faire**.

The screenshot shows the 'SENSOR DATA ANALYTIC' application interface. The top navigation bar includes a user profile ('Admin PROFILO') and notification icons (1 message, 0 pushes). The left sidebar menu has items: 'Accueil', 'Laboratoires', 'Capteurs', 'Alertes' (which is highlighted in blue), and 'Analyse'. The main content area displays a dashboard with four summary cards: 'LABORATOIRES' (2), 'LABORATOIRES' (3), 'PUSHES' (30), and 'PUSHES' (0). Below the dashboard is a section titled 'Alertes' with sub-sections 'LISTE DES ALERTES' and 'AJOUTER UNE ALERTE'. The 'AJOUTER UNE ALERTE' section contains three input fields: 'Valeur critique' (with placeholder 'Une valeur admise pivo de l'alerte'), 'Action à effectuer au cas eventuel' (with placeholder 'NOTIFIER'), and 'Operateur utiliser pour evaluer la donnée sondée par le capteur' (with placeholder 'SUPERIORITE'). A blue button at the bottom right of this section says 'Ajouter une nouvelle Alerte'.

Figure 54 : Ajouter un modèle d'alerte

### b. Liste des modèles d'alerte

La liste des modèles d'alerte on peut distinguer directement :

- **Le nom** : il s'agit du nom de l'alerte ;
- **La comparaison** : Il s'agit de l'opérateur de comparaison utilisé pour statuer sur une valeur ;
- **Valeur critique** : il s'agit de la valeur utilisée lors de la comparaison ;
- **Action à effectuer** : ici il s'agit d'une action que l'on doit effectuer lorsque cette alerte est signalée ;

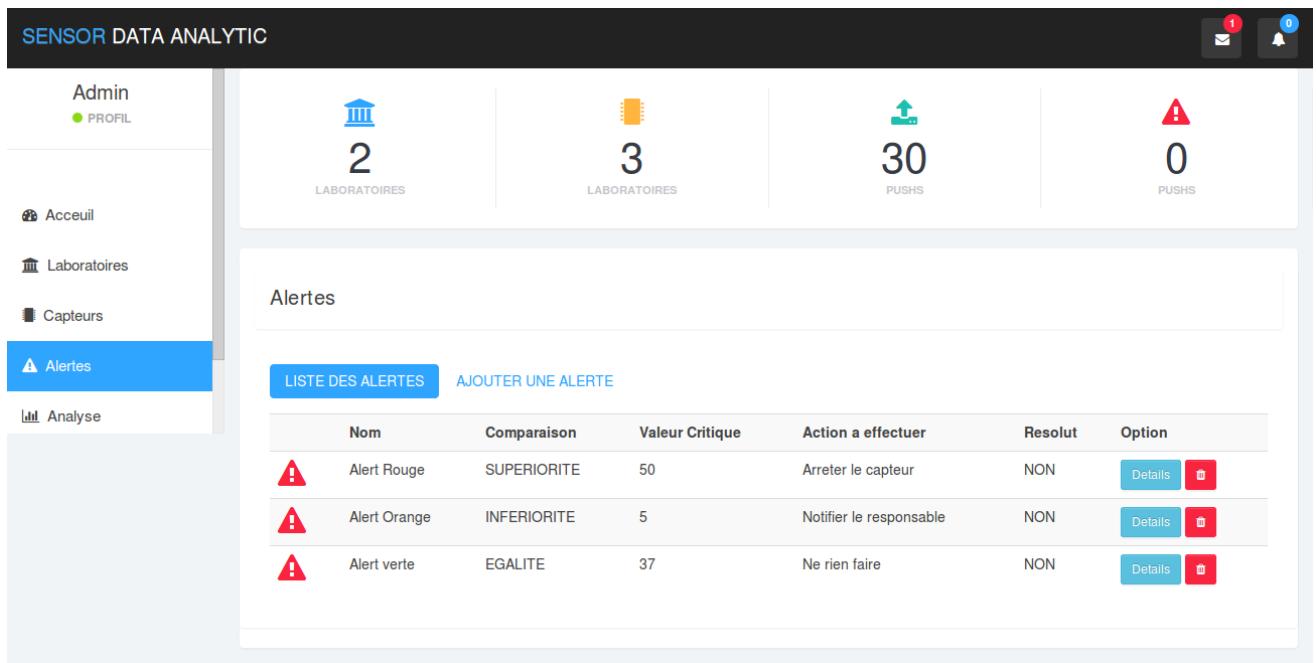


Figure 55 : Liste des modèles d'alerte

## 6. Le module Analyse SDA

On distingue deux onglets principaux, « VALIDATION DES ECHANTILLON » et « CORRELATION ».

### a. Validation des données

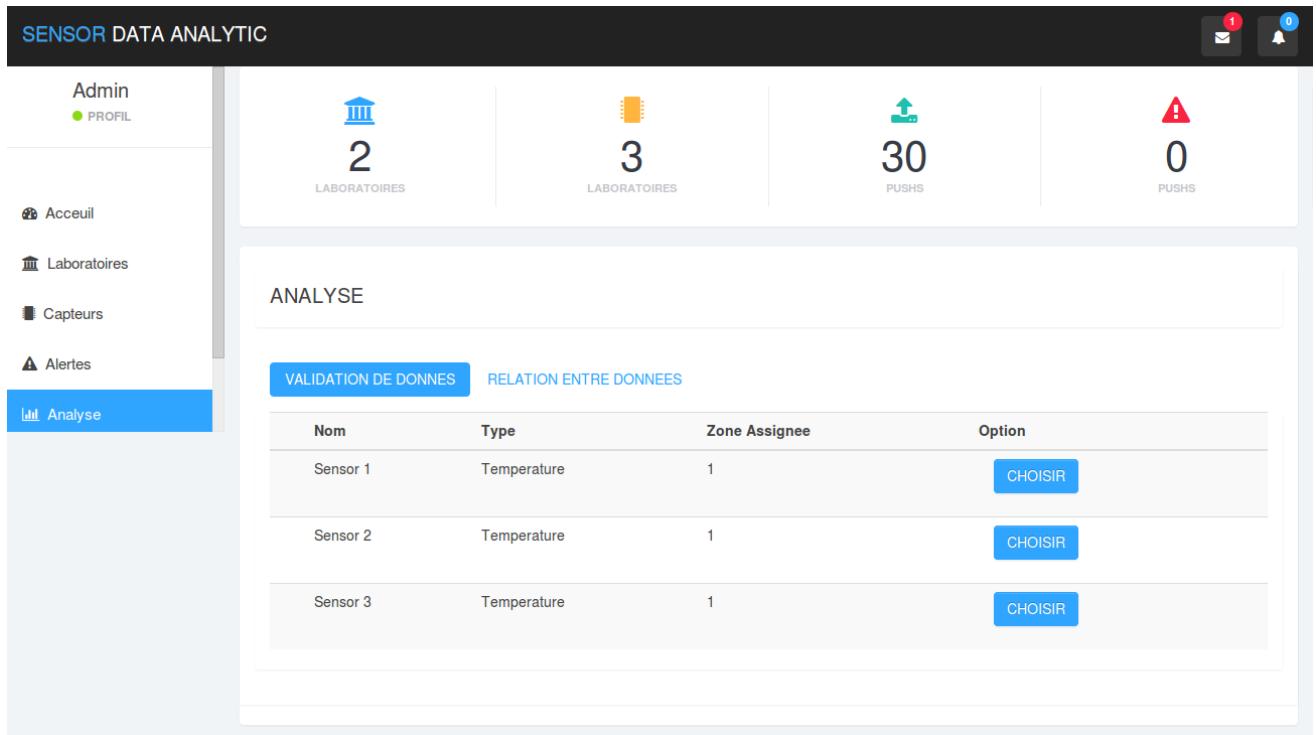


Figure 56 : Sélectionner un capteur pour une validation

Pour analyser les échantillons d'un capteur il faut sélectionner le capteur en question puis observer les résultats sur l'analyse.

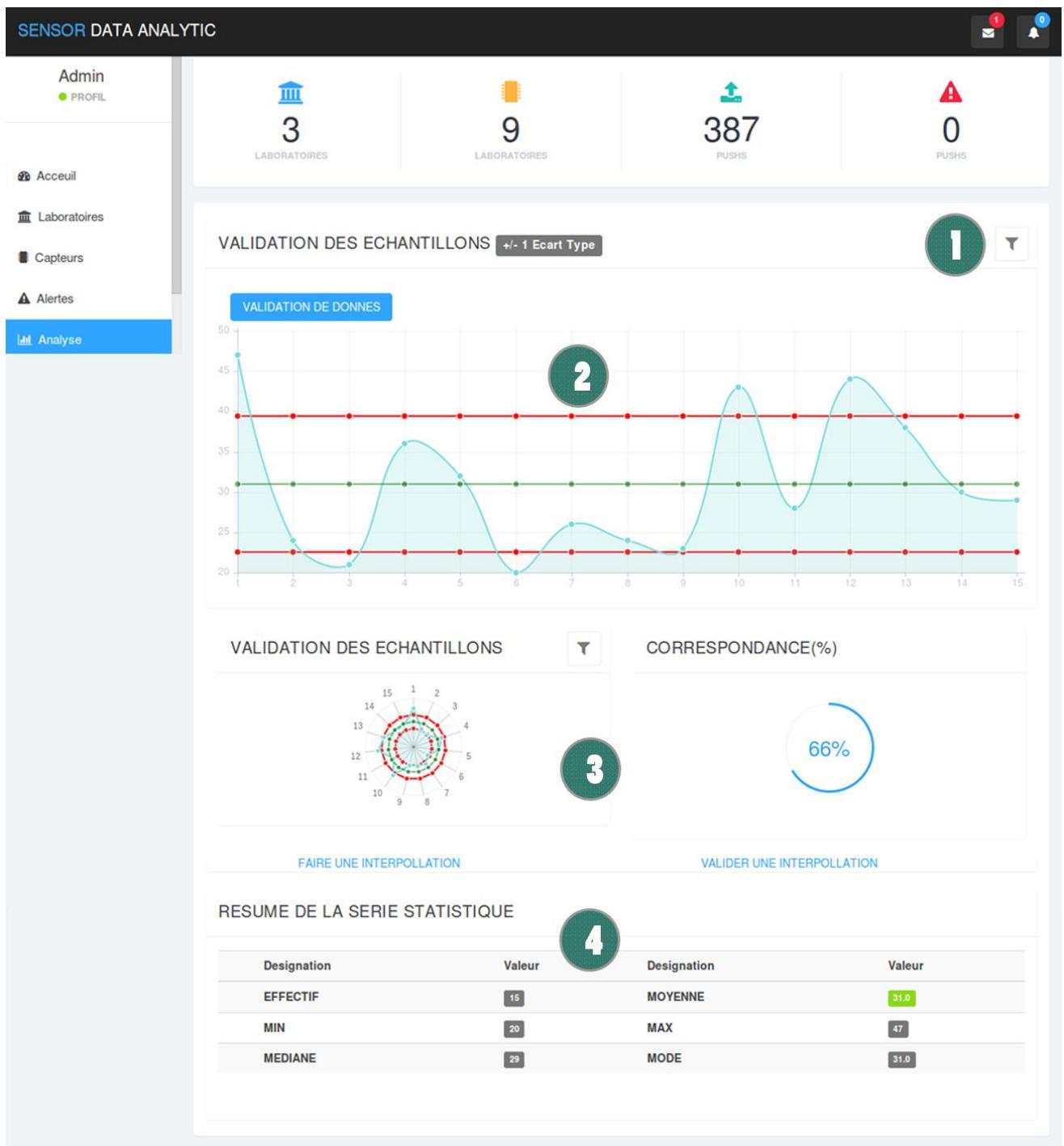


Figure 57 : Analyse des échantillons

Sur la figure on a plusieurs parties que l'on va expliquer :

- **Le numéro 1 :** il s'agit d'un filtre qui permet d'augmenter ou de réduire la sensibilité, en fonction des écarts type. On peut décider que les données du capteur seront comprises entre, **la moyenne +/- 1 écart Type, la moyenne +/- 2 écarts Type** en enfin **la moyenne +/- 3 écarts Type**. Ce filtre permet de modifier le graphe et de modifier tout le rendu de la page ;

- **Numéro 2** : Il s'agit d'une représentation personnalisée de **Jenning** avec en plus du graphe des données, la moyenne et les deux bornes relatives définies par la sensibilité choisie ;
- **Numéro 3** : Il s'agit d'un rapport par rapport au précédent graphe :
  - o **A gauche** : on peut voir un autre graphe qui permet de voir la précision des prélèvements des échantillons ;
  - o **A droite** : on montre le pourcentage de données qui sont dans le bon intervalle. Le but ici est d'augmenter ce pourcentage et d'éliminer les valeurs atypiques.
  - o **Plus bas** : on a deux liens, le premier qui permet de faire une interpolation linéaire sur les données qui sont atypiques (ceci va permettre de normaliser un peu plus le prélèvement, car les valeurs trop éloignées de la moyenne altèrent sa valeur). Et l'autre lien permet d'annuler la précédente interpolation qui permet de revenir à la version précédente des données au cas où l'interpolation ne satisfait pas à nos attentes.

Le but de cette page est de permettre la validation d'une série d'échantillon en utilisant les règles de WESTGARD et l'usage de l'interpolation. Si on prend la figure précédente qui a des valeurs un peu dispersées et 1/3 valeurs qui ne sont pas dans l'intervalle (25 – 40) ; On va obtenir la figure ci-dessous :

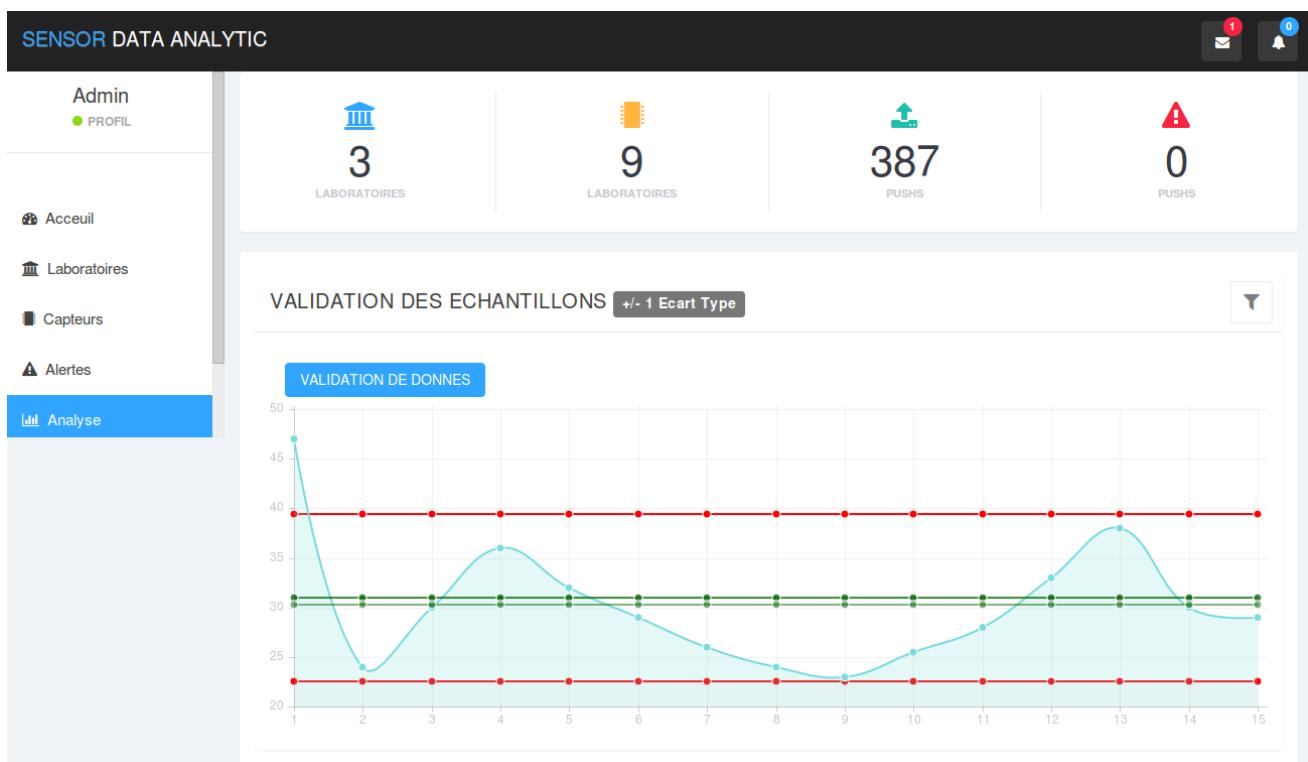


Figure 58 : Correction due à l'interpolation

On quitte de 1/3 à 1/15ième ce qui correspond à un énorme gain en termes de qualité.

## b. Combinaison de deux variables

Dans cette section il n'est nullement question de modifier les valeurs, on va sélectionner deux capteurs (ou source de données) puis on va faire un ensemble d'opérations et avoir accès à un certain nombre d'informations. Le but de ces opérations est de pouvoir ressortir un modèle mathématique permettant de trouver une relation entre les deux sources de données.

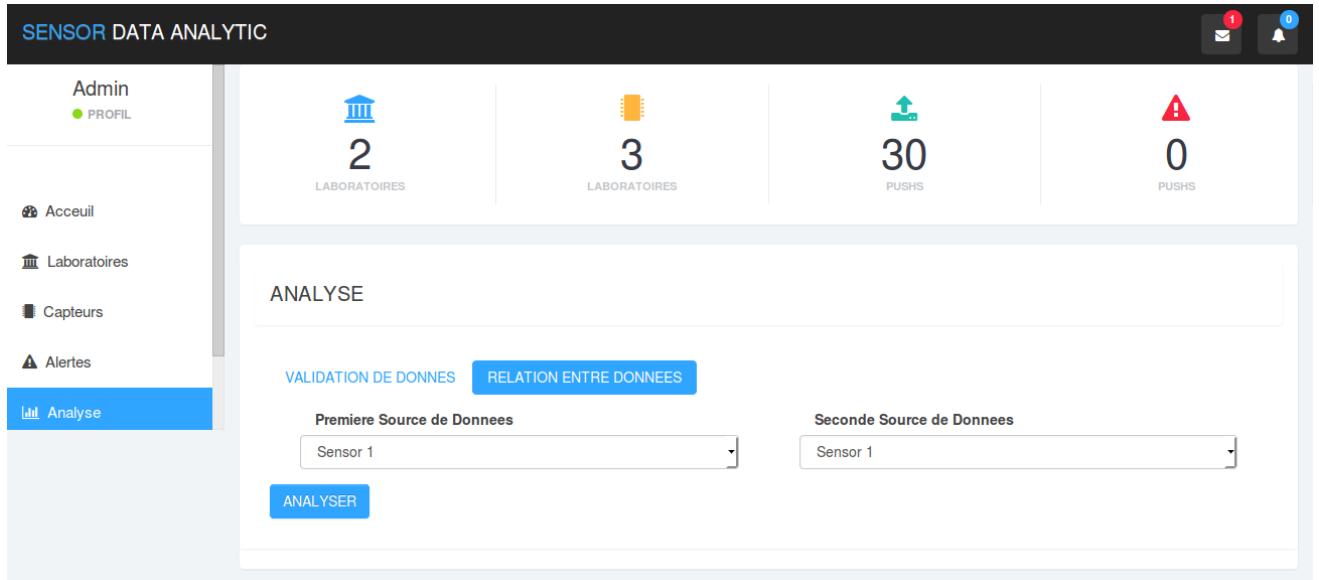


Figure 59 : Choix d'un couple de capteurs

Jean Herman GUAY disais ceci : « On n'accumule pas les variables pour les regarder côté à côté, mais pour les combiner » [17] ; il est donc question ici de prendre parmi nos sources de données, deux que l'on va croiser et analyser afin de quantifier la relation qui les lie.

Donc nous sélectionnons un premier capteur qui va servir de première source de données, puis un second qui va servir de seconde source. Une fois que l'on a sélectionné on clique sur le bouton analyse pour démarrer l'analyse.

Pour l'analyse nous allons prendre des sources simples avec une quinzaine de données (cas actuel sur la capture) et nous allons présenter les sections de la page d'analyse.

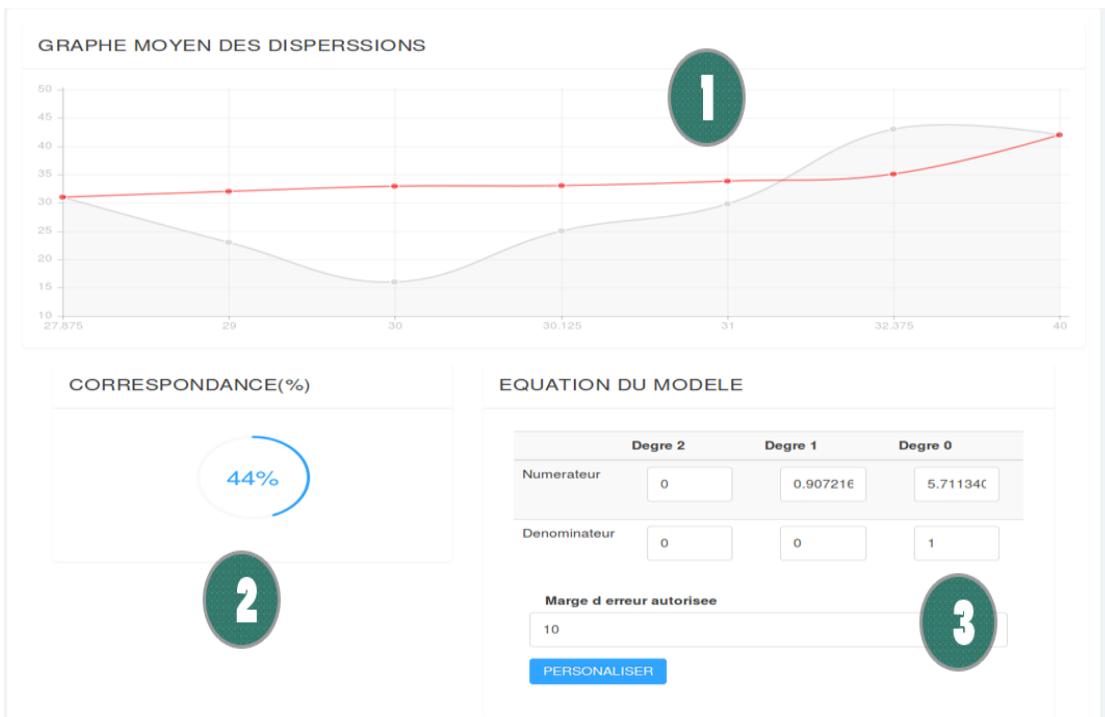


Figure 60 : Relation entre données – modèle

Sur la figure ci-dessus nous pouvons voir trois éléments principaux :

- **Numéro 1** : on remarque un graphe gris et une ligne rouge :
  - o Graphe gris : il s'agit de graphe moyen obtenu en trouvant la moyenne pondéré des concentrations des points à chaque indice. Nous avons préféré cette représentation car elle permet facilement de visualiser la correspondance avec le modèle mathématique ;
  - o La ligne rouge : il s'agit d'une représentation du modèle dont l'équation est donnée sur la section **numéro 3**. Le but est de rapprocher le plus possible la représentation graphique du modèle à celle du graphe ; et augmenter le plus possible le taux de correspondance sur la section **numéro 2**.
- **Numéro 2** : il s'agit du pourcentage entre le modèle et le graphe, le but est de trouver la combinaison qui maximise cette valeur (statistiquement parlant elle correspond au coefficient de corrélation en pourcentage) ;
- **Numéro 3** : on a plus haut une équation rationnelle qui permet de déterminer les paramètres du modèle ; ici on supporte les équations linaires, quadratique et rationnelles. Toute modification des paramètres a un impact direct sur le graphe et sur le pourcentage de correspondance. Et plus bas on définit une marge d'erreur de correspondance.

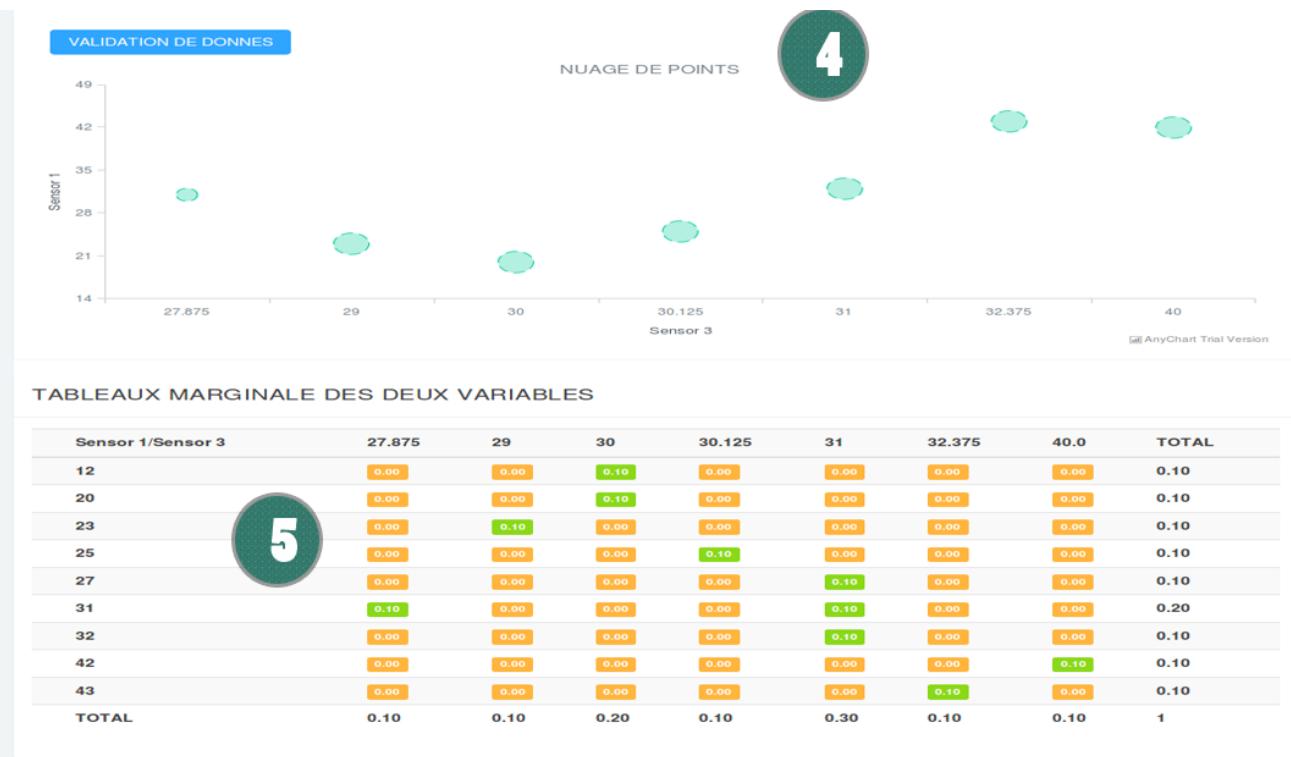


Figure 61 : Relation entre les données - Tableau marginale

Sur la figure ci-dessus nous pouvons voir trois éléments principaux :

- **Numéro 4 :** Cette section nous présente un nuage de points pondéré qui représente la situation ou un couple de données entre les deux source.
- **Numéro 5 :** il s'agit des calculs effectuer pour dresser un tableau de valeurs marginales entre les deux sources de données ;

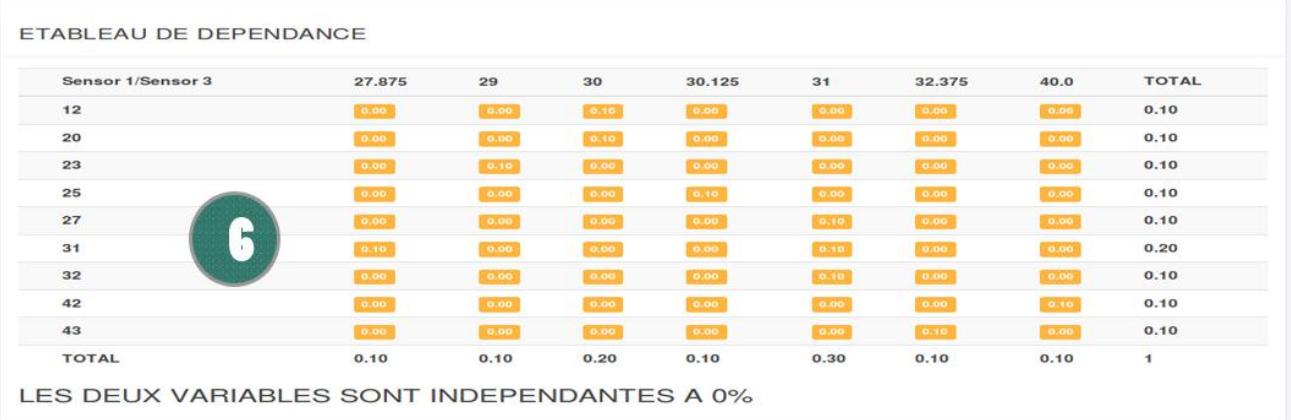


Figure 62 : Indépendance entre deux variables

Ici on effectue une série de calcul pour déterminer le taux d'indépendance entre les deux variables. Il est à noter que si deux variables sont corrélées dans les calculs, ceci ne suffit pas à conclure qu'elles le sont réellement il faut encore étudier le contexte du sujet.

## **CONCLUSION ET PERSPECTIVES**

En définitive il était question pour nous dans ce travail de concevoir et mettre en place une infrastructure SDA ; permettant grâce à des outils électroniques et une combinaison de logiciels, de faciliter et d'automatiser les processus de recherche ; allant des observations à la modélisation en passant par le suivi des données. Pour mener à bien notre travail, nous avons précisé le contexte de ce travail, ainsi que la problématique posée. Puis nous avons parlé des généralités, sur des différents acteurs et notions intervenant dans le système. Ensuite nous avons formalisé les besoins de cette application et présenter de manière détaillée la modélisation des différentes plateformes, ainsi que leurs implémentations. Pour finir, nous avons présenté les résultats suivis des commentaires de notre plateforme. Parvenu au terme de ce travail et vu les résultats obtenus, nous pouvons dire que les objectifs de ce projet ont été atteints. Par ailleurs, les perspectives envisagées sont orientées dans le sens de permettre à la plateforme de traiter simultanément plus de deux variables et pouvoir travailler sur des modèles d'au moins trois dimensions et d'intégrer une intelligence artificielle qui permettrait d'automatiser et optimiser la modélisation mathématique du problème.

# BIBLIOGRAPHIE

- [1] «wikipedia - Institut\_de\_recherche\_pour\_le\_développement,» 29 Juin 2018. [En ligne]. Available: [https://fr.wikipedia.org/wiki/Institut\\_de\\_recherche\\_pour\\_le\\_d%C3%A9veloppement](https://fr.wikipedia.org/wiki/Institut_de_recherche_pour_le_d%C3%A9veloppement). [Accès le 30 Juin 2018].
- [2] ird, «historique,» 27 juillet 2018. [En ligne]. Available: <https://www.ird.fr/l-ird/historique>. [Accès le 28 juillet 2018].
- [3] IRD, «Organigramme,» 28 Juillet 2018. [En ligne]. Available: <https://www.ird.fr/l-ird/l-organigramme>. [Accès le 28 Juillet 2018].
- [4] ird, «pole-science,» pp. <https://www.ird.fr/l-ird/l-organigramme/pole-science>.
- [5] IRD, «Les partenaires,» [En ligne]. Available: [http://www.ird.fr/les-partenariats/renforcement-des-capacites/des-programmes-specifiques/jeunes-equipes-associees-a-l-ird-jeai/jeai-en-cours-de-soutien-par-departement-scientifique/departement-sante-et-societes-sas/jeai-impala-cameroun-2018-2020/\(language](http://www.ird.fr/les-partenariats/renforcement-des-capacites/des-programmes-specifiques/jeunes-equipes-associees-a-l-ird-jeai/jeai-en-cours-de-soutien-par-departement-scientifique/departement-sante-et-societes-sas/jeai-impala-cameroun-2018-2020/(language).
- [6] G. Asch, Les capteurs en instrumentation industrielle, Dunod, 2016.
- [7] wikipedia, «raspberry,» 29 Juin 2018. [https://fr.wikipedia.org/wiki/Raspberry\\_Pi#Tableau\\_comparatif](https://fr.wikipedia.org/wiki/Raspberry_Pi#Tableau_comparatif).
- [8] N. NICODEM, Cours Electronique de Base Licence II - Loi des Mailles, 2014.
- [9] N. NICODEM, Cours Electronique de Base licence II, 2014.
- [10] elektronique, «resistance,» p. <http://www.elektronique.fr/cours/resistance/resistance.php>.
- [11] H. SENHAJI, LES RESEAUX DE CAPTEURS SANS FILS, Format Kindle.
- [12] «Go TRONIC,» pp. [https://www.gotronic.fr/art-capteur-de-t-et-d-humidite-dht22-20719.htm#complte\\_desc](https://www.gotronic.fr/art-capteur-de-t-et-d-humidite-dht22-20719.htm#complte_desc).
- [13] W. PI, «wiringpi-home,» 2018. [En ligne]. Available: <http://wiringpi.com/>.
- [14] «pharo-iot - PharoThings,» pp. <https://github.com/pharo-iot/PharoThings>.
- [15] seaside, «about,» p. <http://seaside.st/about>.
- [16] J. H. GUAY, Statistiques en sciences humaines avec R, BOECK.
- [17] D. Maniez, programmez un raspberry pi, DUNOD.

# INDEX

## A

Android, 52, 53, 81, 82

## B

BarChart, 89

Bootstrap, 85, 86

breadboard, 9

## C

capteurs, 9, 10, 11, 34, 35, 36, 37, 38, 39, 40, 42, 43, 44, 45, 46, 58, 62, 64, 66, 69, 72, 73, 74, 78

capteurs analogiques, 9

capteurs logiques, 10

capteurs numériques, 10

corrélation, 29, 30, 31, 68, 104

## E

écart-type, 25

## G

GPIO, 14, 15, 36, 37, 58, 73, 76

## I

IRD, 2, 3, 5, 6

## L

Levey-Jennings, 26, 44

LineChart, 83, 87

Loi d'Ohm, 18

Loi des mailles, 18

Loi des nœuds, 19

## M

médiane, 25

mode, 25, 71, 79

moyenne, 24, 25, 26, 29, 32, 35, 45, 101, 102, 104

## N

NUAGE de points, 29

## P

Pharo, 21, 22, 23, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 52, 53, 76, 77

PharoThing, 76

## S

SDA

Sensor Data Analytics, 1, 9, 33, 34, 35, 37, 38, 47, 48, 49, 58, 61, 62, 63, 65, 66, 67, 68

SeaSide, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 52, 83, 84, 85

## T

Tableau Marginale, 28

ThinkSpeak, 34, 35, 58, 59, 60, 80, 81

## V

variance, 25

## W

WESTGRAD, 26, 44

WiringPi, 52, 76, 78