GROUPE ESEO

Angers
Paris
Shanghai
Dijon

www.eseo.fr

Référence : CONC_EB1

Version : 3

Révision : 17

Équipe : Équipe B1

Responsable du document : Axel ROLLO

État du document : Fermé

Dossier de conception
ST Microelectronics

Notes : Le présent document est un document à but pédagogique. Il a été réalisé sous la direction de Jérôme Delatour, en collaboration avec des enseignants et des étudiants (Marwan Boughammoura, Sonasi Katoa, Sami Kouatli, Kriss Amzal, Romain Richard, Axel Rollo, Nivenn Lanos, Eva Beaubrun, Martin Bourseau, Jocelyn Girault et Alexis Gaonac'h) de l'option SE du groupe ESEO.
La société FORMATO et la société BT Company n'ont pas d'existence légale, il s'agit de sociétés fictives créées pour des besoins pédagogiques. Le personnage de César Bistrucla, un des consultants de FORMATO, est le pseudonyme de Jérôme Delatour.

Liste des évolutions et validations du document.

Date	Action	Auteur	Version	Révision
19 Octobre 2017	Création du livrable sous LateX	Axel ROLLO	0.1	1
9 & 10 Octobre 2017	Ajout des différentes parties réalisées	Axel ROLLO	1	5
16 Octobre 2017	Ajout des nouvelles parties complétées	Axel ROLLO	1.1	7
17 Octobre 2017	Ajout des MAE & de la nouvelle architecture candidate	Axel ROLLO	1.1	7

Table des matières

Chapitre 1

INTRODUCTION

1.1 Objet

Ce document de conception a pour objectif de rassembler tous les éléments utiles pour la conception logicielle et matérielle du projet « Ville intelligente – Qualité de l'air » réalisé par l'équipe de développement pour la PAVIC. Il présentera les éléments de conception déterminés suite à l'élaboration du dossier de spécification [SPEC_PFE_VI]. Ce dossier suit les recommandations de la norme IEEE 1016_2009 [IEEE- 1016_2009]. Les schéma et figures présentés dans ce document suivent la norme UML 2.4.1 [UML_2.4_2011].

1.2 Portée

Ce document décrit l'ensemble des éléments de conception se rapportant au Système à l'Etude (SaE).

Ce dossier de conception est destiné :

— à l'équipe de développement C et Android, pour permettre la réalisation des logiciels du projet « Ville intelligente – Qualité de l'air ».

— à l'équipe de développement matériel électronique.

— aux auditeurs étant donné que ce projet fera l'objet d'une évaluation finale.

— Aux testeurs, pour qu'ils puissent établir les tests à réaliser ainsi que récupérer les résultats de ces derniers.

1.3 Définitions, acronymes et abréviations

Les abréviations utilisées dans le présent document sont répertoriées et expliquées dans le tableau présenté ci-dessous. Les termes utiles pour interpréter correctement ce dossier de conception, sont définis dans le dictionnaire du domaine présent dans le dossier de spécification [SPEC_PFE_VI] et au chapitre 4 du présent document.

Acronymes, abréviations	**Définitions**
alt	Abréviation utilisée dans les diagrammes de séquences pour représenter les alternatives possibles.

BAL	Abréviation utilisée pour le terme "Boite aux Lettres"
CU	Cas d'Utilisation.
DS	Diagramme de Séquence.
Fiabilité	La fiabilité est l'aptitude d'un composant ou d'un système à fonctionner pendant un intervalle de temps.
GPIO (General Purpose Input/Output)	Désigne des ports d'entrée/sortie.
IHM (Interface Homma Machine)	Moyen permettant à l'administrateur et aux utilisateur d'interagir avec la valise.
LED (Light-Emitting Diode)	Une diode électroluminescente.
MC	Abréviation utilisée dans les diagrammes de séquence pour préciser que l'acteur appartient à MasterChef.
MM	Abréviation utilisée dans les diagrammes de séquence pour préciser que l'acteur appartient à MasterMind.
MP	Abréviation utilisée dans les diagrammes de séquence pour préciser que l'acteur appartient à MasterPower.
N.A	Non applicable.
OMG (Object Management Group)	Association à but non lucratif dont l'objectif est de standardiser et promouvoir le modèle objet.
PAQL (Plan d'Assurance Qualité Logiciel)	Dossier permettant d'assurer une qualité au projet de faciliter ce dernier.
PDU (Power Distribution Unit)	Ensemble power et carte STM32.
ProSE (Projet Système Embarqué)	Projet de l'option Système Embarqué par groupe de 10/11 étudiants.
ref	Abréviation utilisée dans les diagrammes de séquence pour indiquer une référence à un autre diagramme de séquence.
SaE (Système à l'Etude)	Il s'agit de l'ensemble des composants MaterChef, MasterMind et MasterPower.
SYTE	Nom du projet.
UI (User Interface)	Interface utilisateur.
UML (Unified Modeling Language)	Notation graphique normalisée, définie par l'OMG et utilisée en génie logiciel.
USB (Universal Serial Bus)	Norme décrivant un bus informatique en transmission série qui sert à connecter des périphériques informatiques à un système informatique.

Wi-Fi (Wireless Fidelity)	Protocole de communication sans fil régis par les normes du groupe IEEE-1016_2009.

TABLE 1.1 – Tableau des acronymes et abréviations

1.4 Références

Voici un tableau récapitulatif des documents utilisés pour le dossier de conception ainsi que les liens permettant d'accéder aux fichiers.

Référence	**Nom, auteur**
[IEEE-1016_2009]	IEEE, std 1016-2009 « Recommended Practice for Software Requirements Specifications », https ://standards.ieee.org/findstds/standard/1016-2009.html, 2009.
[SPEC_PFE_VI]	Dossier de spécification du projet « Ville intelligente – Qualité de l'air », référentiel documentaire projet PAVIC 2017/2018.
[UML_2.4_2011]	OMG, « Unified Modeling Language », version 2.4.1, http ://www.omg.org/spec/UML/, 2011.

TABLE 1.2 – Tableau des références

1.5 Vue d'ensemble

Ce document de conception est structuré en plusieurs parties :

— une première partie, qui concerne la conception générale du prototype. Cette partie présente l'architecture candidate et donne les grands principes de fonctionnement du projet « Ville Intelligente – Qualité de l'air ».

— une seconde partie présentera la conception détaillée. Cette partie présente les composantes du système en précisant cette fois-ci la gestion des entrées et des sorties, la gestion multitâche ainsi que la gestion de la persistance.

— un dictionnaire du domaine constitue la dernière partie du document.

Chapitre 2

Conception générale

2.1 Architecture candidate

Le diagramme d'objets suivant présente l'architecture candidate du système à l'étude. On y retrouve toutes les classes de la conception générale avec leurs méthodes.

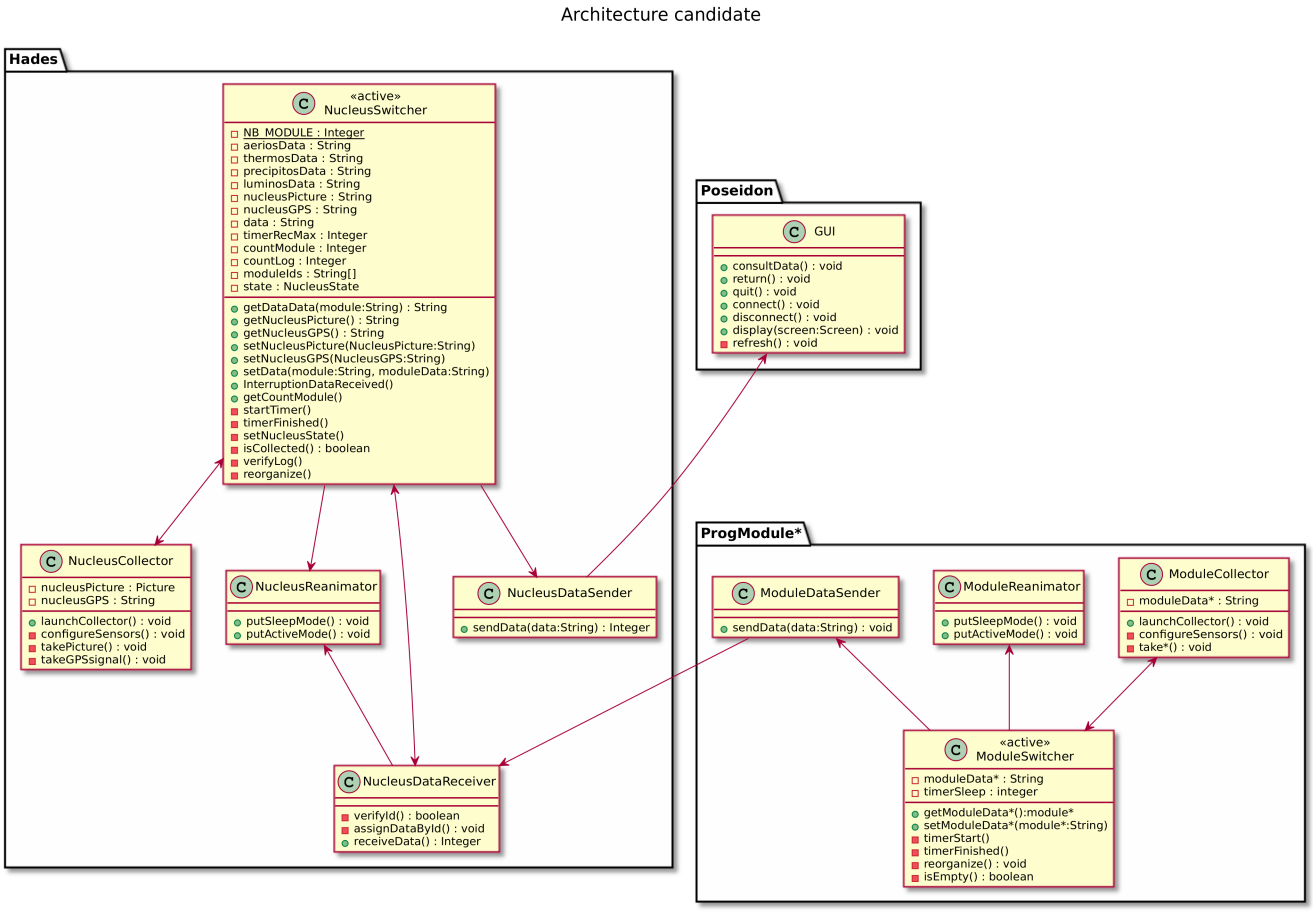

FIGURE 2.1 – Architecture candidate

Le diagramme ci-dessus représente l'architecture logique qui sera adoptée pour la réalisation du prototype de la station. Il s'agit de la conception générale ; l'hypothèse d'un système matériel à ressource infinie est pour l'instant posée.

2.1.1 Poseidon

Dans ce diagramme on retrouve « GUI » qui permet de contrôler les affichages de l'application « Poseidon » ainsi que les interactions utilisateurs. Son principal rôle est de collecter et de mettre à jour les champs de l'écran « Screen_Data ».

2.1.2 Nucleus

Le composant « NucleusDataRecever » a la responsabilité de recevoir des données sous forme de trame. Il extrait l'information de la trame de réception et envoi les données triées vers le « NucleusSwitcher ». Il peut recevoir les quatre différentes trames venant des quatre modules.

Le composant « NucleusSwitcher » a un rôle d'aiguilleur, il redirige les différentes données vers les différentes plages mémoires, afin d'organiser le contenu des données reçues. De plus il met en forme une nouvelle trame comprenant les données des quatre modules. Il l'envoi ensuite vers « NucleusDataSender ».

Le composant « NucleusCollector » a le rôle de collecter les informations des capteurs interne à Nucléus, c'est-à-dire l'image du capteur photographique et la donnée GPS. Il envoi ensuite directement les informations à « NucleusSwitcher » pour stocker les données.

Le composant « NucleusReanimator » permet de contrôler les différents modes de fonctionnement de la puce ESP8266-01 (« Sleep mode » ou « Active mode »).

Le composant « NucleusDataSender » a le rôle d'envoyer la trame mise en forme précédemment vers un Broker avec l'aide du protocole MQTT.

2.1.3 Modules

Pour le prototype réalisé, nous utilisons quatre modules avec des fonctionnalités manifestement similaires. Pour cette raison, nous avons créé un modèle pouvant être applicable à n'importe quel de nos quatre modules. Par la suite, l'ajout éventuel d'un nouveau module ne posera pas de soucis. Il faut tout de même qu'il respecte les normes de conception et de spécification du projet « Ville Intelligente – Qualité de l'air ».

Le composant « ModuleReanimator » permet de contrôler les différents modes de fonctionnement de la puce du NRF24LE1 (« Sleep mode » ou « Active mode »).

Le composant « ModuleDataSender » permet d'envoyer la trame représentant la concaténation des différents relevés des mesures des capteurs.

Le composant « ModuleSwitcher » permet d'aiguiller les différentes informations aux différentes plages mémoires disponibles sur la puce. Il a aussi un rôle de timer. Tous les temps « T_DR » il réveille la puce, lance une nouvelle acquisition de donnée puis la rendors.

Le composant « ModuleCollector » a pour rôle de récupérer tous les temps « T_DR » les différentes données des capteurs et de les envoyer au « ModuleSwitcher ».

2.2 Grands principes de fonctionnement

Cette partie présentera le fonctionnement du prototype. Les diagrammes de séquence présentés sont : le diagramme de séquence principal et rafraîchir les données.

2.2.1 Principal

Ce diagramme de séquence correspond au scénario nominal du CU « Informer périodiquement les utilisateurs afin d'anticiper les risques de réactions allergiques. » du dossier de spécification. L'utilisateur effectue une séquence d'enchaînement d'actions pour observer le principe de fonctionnement du système.

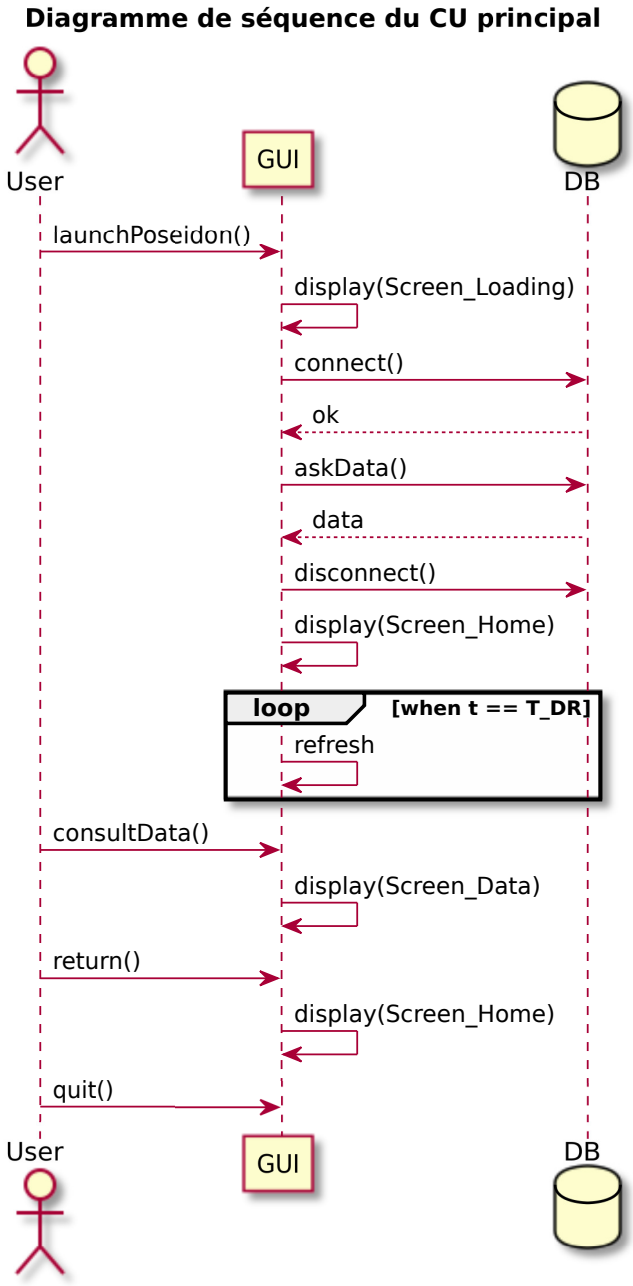

FIGURE 2.2 – Diagramme de séquence : Principal

2.2.2 Rafraîchir données

Ce diagramme représente le scénario du CU « Rafraîchir les données » du dossier de spécification. Poseidon demande à se connecter à la base de donnée pour récupérer les données les plus à jour.

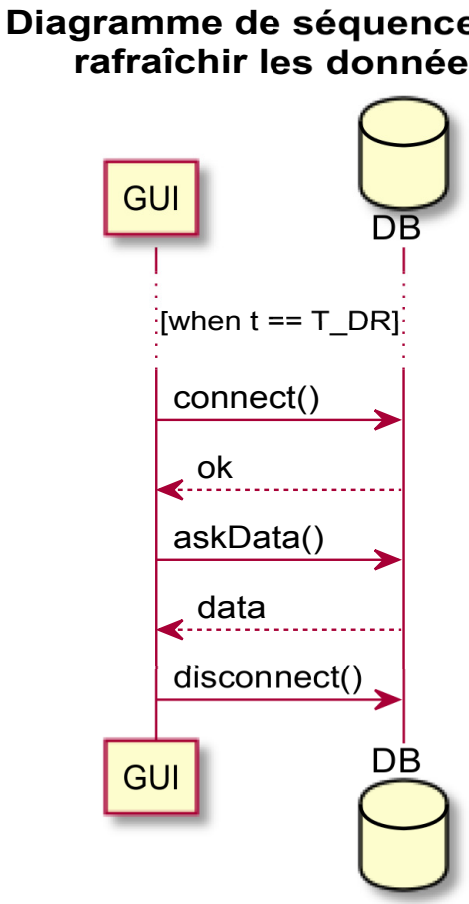

FIGURE 2.3 – Diagramme de séquence : Rafraîchir données

2.3 Description des composants

2.3.1 Description des types manipulés entre composants

— aeriosData, thermosData, precipitosData, luminosData, nucleusPicture, nucleusGPS, moduleData* : Chaîne de caractère pour y stocker les données des différents capteurs à un instant t.

— data : Chaîne de caractère qui correspond à l'assemblage des différentes données de capteurs.

— timerRecMax : Entier non signé qui correspond à un temps d'échantillonnage maximal.

— timerSleep : Entier non signé qui correspond au temps pour lequel le module doit se mettre à l'état "Sleep Mode".

2.3.2 Description des unités manipulés pour chaque type

— ppm : La "partie par million", est la faction valant un millionième. On l'utilise pour exprimer une faction massique (cf. mg/m^3, 1 mg/m^3 = 0.001 ppm).

— mg/m^3 : Le "milligramme par mètre cube" est une unité de mesure généralement utilisé pour mesurer la concentration de polluant dans l'air.

— °C : Le "degré celcius" est unité pour exprimer une différence de température.

— % : Le pourcentage est un nombre représenté comme un nombre relatif de l'ordre de 10^{-2}. Il est utilisé pour représenter une ordre de grande allant de 0 à 100.

— Pa : Le pascal est une unité de mesure de pression.

— Lx : Le lux est une unité de mesure de l'éclairement lumineux.

— m/s : Le "mètre par seconde", mesure une vitesse de propagation.

— m : Le mètre mesure une distance.

Chapitre 3

Conception détaillée

La conception détaillée définit l'architecture logique du système, c'est à dire :

— répartir le système sur l'architecture matérielle.

— gérer les entrées/sorties et les IHMs.

— élaborer les protocoles de communication.

— définir le parallélisme et l'initialisation.

3.1 Architecture physique

Cette partie présente l'architecture physique et logiciel détaillé du projet "Ville intelligente - Qualité de l'air". Elle reprend la conception générale auquel nous ajoutons certaines des éléments pour que l'architecture soit portée sur des contraintes réelles.

L'architecture physique est séparée en plusieurs parties, qui comprend, la partie Nucelus et la partie des Modules. Nous présentons également une autre partie qui présente la gestion des entrées/sorties, la communication et enfin une explication sur la gestion de la concurrence multitâche.

Voici un rappel de l'architecture matériel. L'architecture physique du projet repose sur la configuration de ce schéma :

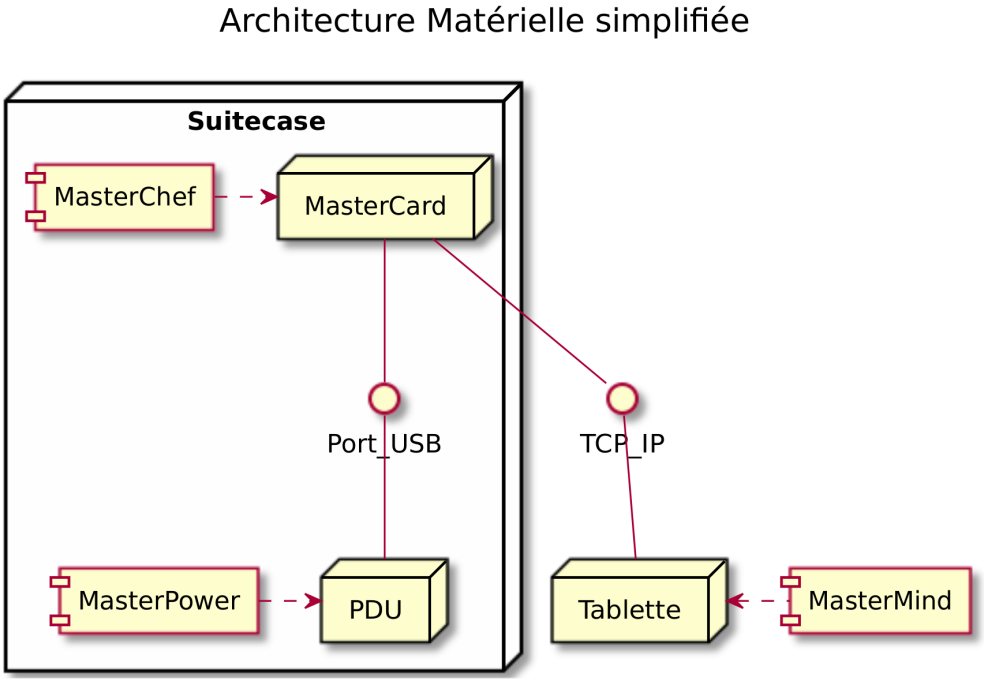

FIGURE 3.1 – Schéma de l'architecture matérielle simplifiée

3.1.1 Schéma Nucleus

Le diagramme ci-dessous représente l'architecture logiciel Hades de Nucleus dans son intégralité.

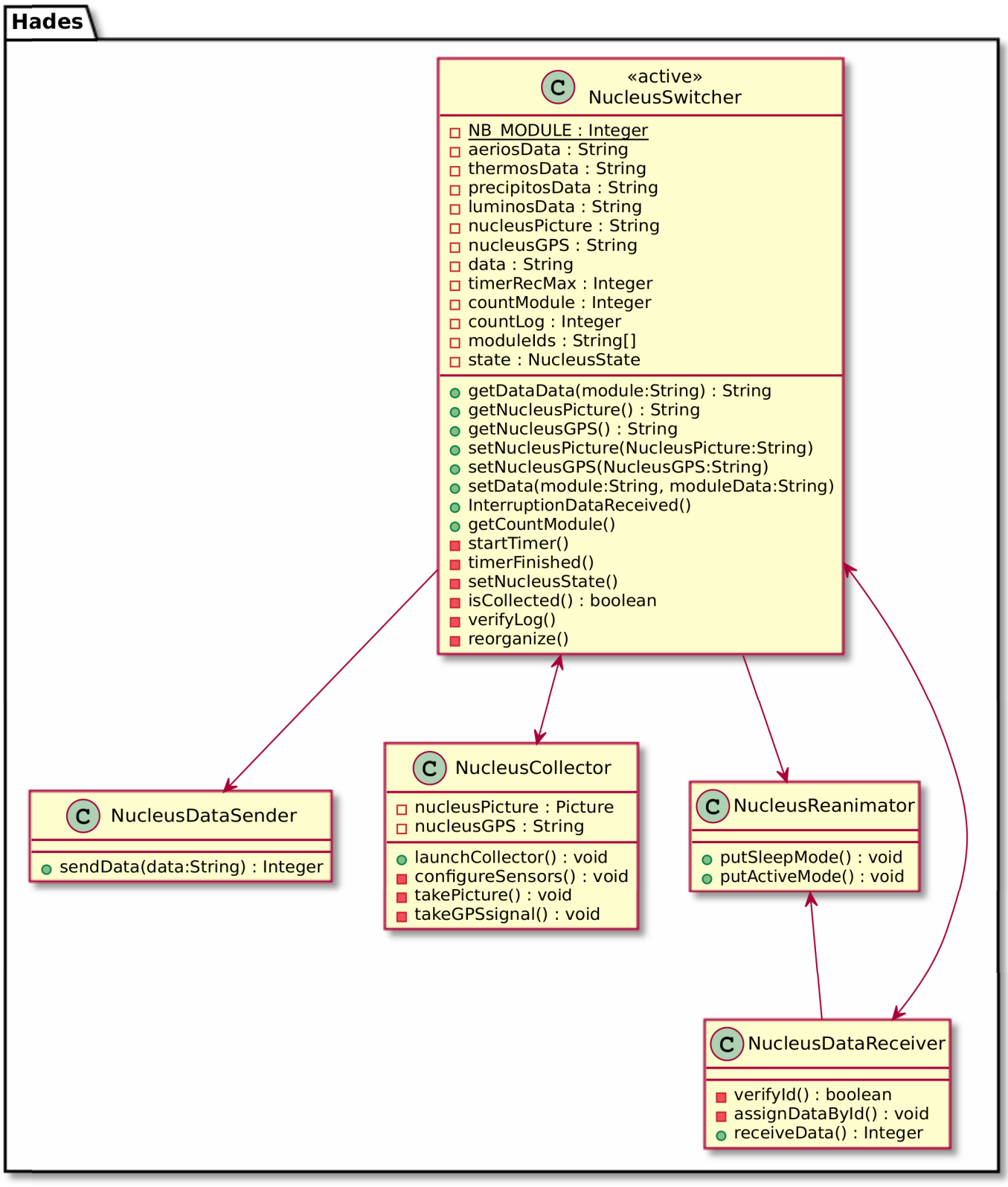

FIGURE 3.2 – Schéma de MasterChef

3.2 Description des classes

3.2.1 Description des classes de Nucleus

3.2.1.1 La Classe NucleusSwitcher «Singleton»

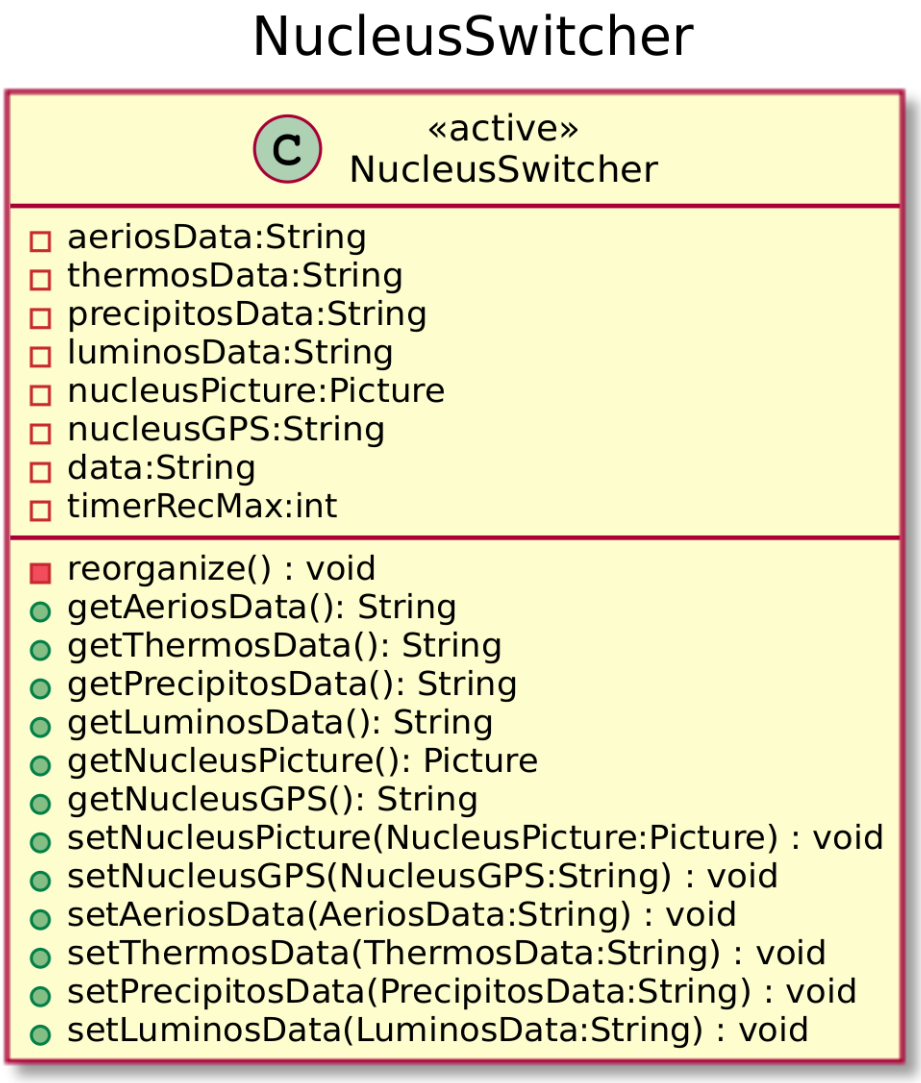

FIGURE 3.3 – Description de la classe NucleusSwitcher.

3.2.1.1.1 Philosophie de conception

La classe NucleusSwitcher est la classe du Nucleus responsable de la machine à états de la Nucleus. NucleusSwitcher gère les différentes actions effectués par la Nucleus.

Cette classe se distingue sous la forme d'un Singleton.

Reponsabilités :

— Initialiser les classes de la Nucleus.

— Gèrer la machine à états de la Nucleus en appelant les méthodes des autres classes.

Collaborateurs :

— NucleusDataReceiver

— NucleusCollector

— NucleusDataSender

— NucleusReanimator

3.2.1.1.2 Description structurelle

3.2.1.1.2.1 Attributs

— aerosData : String : les informations provenant de aeros.

— thermosData : String : les informations provenant de thermos.

— precipitosData : String : : les informations provenant de precipitos.

— luminosData : String : les informations provenant de luminos.

— nucleusPicture : VOIR : la photo prise par NucleusCollector.

— nucleusGPS : String : la localisation GPS prise pas NucleusCollector.

— TIMER_REC_MAX : int : le temps maximum que la Nucleus attend entre la première reception et la reception suivante avant de se remettre en sommeil.

— dataSend : String : l'ensemble des informations des différents modules une fois mises en forme.

3.2.1.1.2.2 Services offerts

— INIT ?

— DESTROY ?

— reorganize() : mets en forme l'ensemble des informations reçus pour pouvoir les envoyer ensuite.

— Accesseurs et mutateurs des attributs module*Data, nucleusPicture et nucleusGPS.

3.2.1.1.3 Gestion du multitâche

Un thread tourne en arrière plan sur la Nucleus et est en attente d'une interruption de réception de données en provenance d'un des modules.

3.2.1.1.4 Gestion des entrées/sorties

NucleusSwitcher possède deux entrées physiques pour effectuer des relevés sur la caméra et sur le GPS. La caméra envoie une donnée sous forme de String via une liaison série SPI. Le GPS envoie une donnée en Sting via une liaison série UART.

3.2.1.1.5 Gestion comportementale

La réception est échantillonnée toutes les 15 minutes, le thread principal surveille et bascule le microprocesseur du "Sleep Mode" au "Active Mode" lorsqu'une émission de donnée en provenance d'un module est détecté.

L'important à comprend est que, pendant une durée de 15 minutes, la Nucleus reçoit les données de tous les modules. Le cas échéant, la Nucléus comprendra que le module absent est soit déconnecté, soit hors service. A la fin de ce temps, le logiciel Hades met en forme la donnée pour la transmettre sur le serveur MQTT.

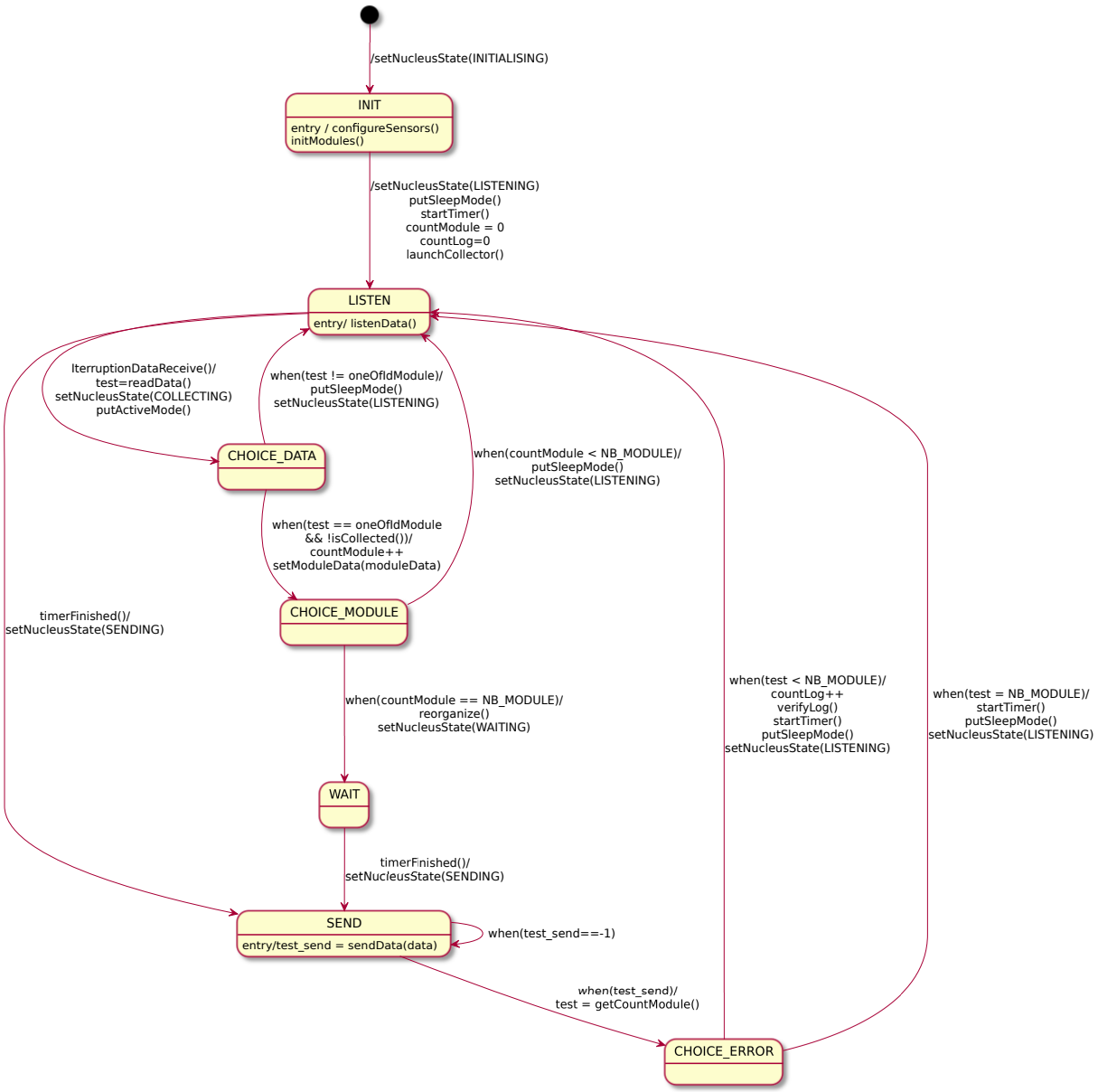

FIGURE 3.4 – Description de la machine à état du switcher de Nucleus.

3.2.1.2 La Classe NucleusCollector «Singleton»

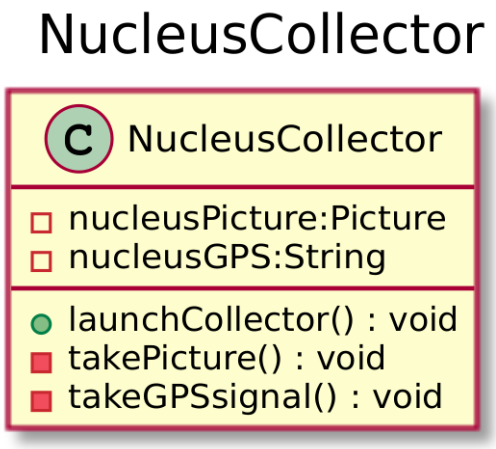

FIGURE 3.5 – Description de la classe NucleusCollector.

3.2.1.2.1 Philosophie de conception

La classe NucleusCollector est la classe du Nucleus responsable de la prise de la photo et de la localisation GPS de la Nucleus.

Cette classe se distingue sous la forme d'un Singleton.

Reponsabilités :

— Prendre une photo.

— Localiser la Nucleus.

Collaborateur :

— NucleusSwitcher.

3.2.1.2.2 Description structurelle

3.2.1.2.2.1 Attributs

— nucleusPicture : A VOIR : Photo prise par la caméra.

— nucleusGPS : String : Localisation GPS.

— A VOIR

3.2.1.2.2.2 Services offerts

— INIT ?

— DESTROY ?

— launchCollector() : Demande de lancer la collecte d'information.

— takePicture() : Prends la photo.

— takeGPS() : Prends la localisation GPS.

— Accesseurs et mutateurs des attributs.

3.2.1.2.3 Gestion du multitâche

A VOIR

3.2.1.2.4 Gestion des entrées/sorties

A VOIR

3.2.1.2.5 Gestion comportementale

A VOIR

3.2.1.3 La Classe NucleusReanimator «Singleton»

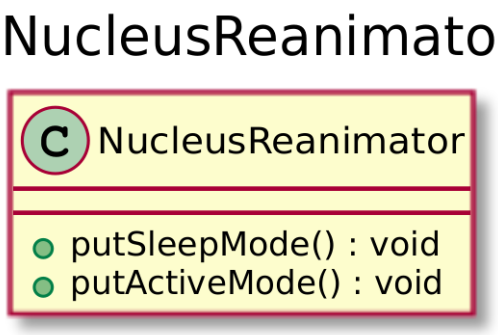

FIGURE 3.6 – Description de la classe NucleusReanimator.

3.2.1.3.1 Philosophie de conception

La classe NucleusReanimator est la classe du Nucleus responsable de la mise en sommeil du système et de son reveil.

Cette classe se distingue sous la forme d'un Singleton.

Reponsabilités :

— Endormir la Nucleus.

— Reveiller la Nucleus.

Collaborateurs :

— NucleusSwitcher.

— NucleusDataReceiver.

3.2.1.3.2 Description structurelle

3.2.1.3.2.1 Attributs

— A VOIR

3.2.1.3.2.2 Services offerts

— INIT ?

— DESTROY ?

— putSleepMode() : Mets la Nucleus en mode sleep.

— putActiveMode() : Mets la Nucleus en mode actif.

3.2.1.3.3 Gestion du multitâche

A VOIR

3.2.1.3.4 Gestion des entrées/sorties

A VOIR

3.2.1.3.5 Gestion comportementale

A VOIR

3.2.1.4 La Classe NucleusDataReceiver «Singleton»

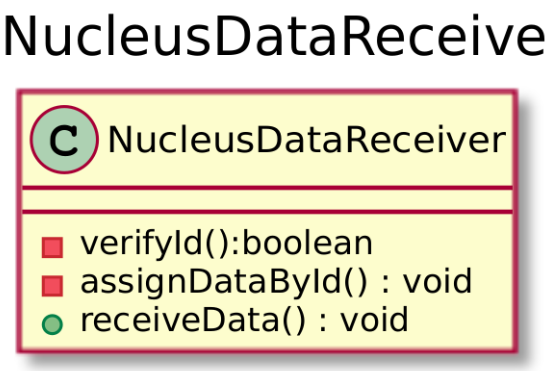

FIGURE 3.7 – Description de la classe NucleusDataReceiver.

3.2.1.4.1 Philosophie de conception

La classe NucleusDataSender est la classe du Nucleus responsable de la reception des données provenant des noeuds de capteurs.

Cette classe se distingue sous la forme d'un Singleton.

Reponsabilités :

— Recevoir les données envoyées par les noeuds de capteurs.

— Vérifier si le message reçu est correct.

— Trouver de quel interlocuteur provient le message.

— Enregistrer les données reçus au bon endroit en mémoire.

— Reveiller le Nucleus si on est en mode sleep.

Collaborateurs :

— NucleusSwitcher.

— NucleusReanimator.

3.2.1.4.2 Description structurelle

3.2.1.4.2.1 Attributs

— A VOIR

3.2.1.4.2.2 Services offerts

— INIT ?

— DESTROY ?

— receiveData() : String : Récupère les données envoyées par le noeuds de capteurs.

— verifyId(data : String) : boolean : Vérifie le message reçu pour savoir s'il est valide et trouver qui à envoyer le message.

— assignDataById(data : String) : Enregistre les données reçus au bon endroit en mémoire.

3.2.1.4.3 Gestion du multitâche

A VOIR

3.2.1.4.4 Gestion des entrées/sorties

A VOIR

3.2.1.4.5 Gestion comportementale

A VOIR PEUT-ETRE BESOIN D'UNE MAE POUR GERER LE REVEIL

3.2.1.5 La Classe NucleusDataSender «Singleton»

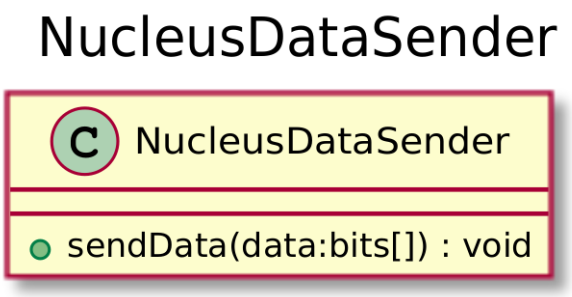

FIGURE 3.8 – Description de la classe NucleusDataSender.

3.2.1.5.1 Philosophie de conception

La classe NucleusDataSender est la classe du Nucleus responsable de la communication entre le Nucleus et le Broker MQQT. Son rôle est d'envoyer les données récoltées par protocole MQTT.

Cette classe se distingue sous la forme d'un Singleton.

Responsabilité :

— Envoyer les données au Broker MQTT.

Collaborateur :

— NucleusSwitcher.

3.2.1.5.2 Description structurelle

3.2.1.5.2.1 Attributs

— A VOIR

3.2.1.5.2.2 Services offerts

— INIT ?

— DESTROY ?

— sendData(data : String) : Envoie l'ensemble des données relevées au broker MQTT.

3.2.1.5.3 Gestion du multitâche

A VOIR

3.2.1.5.4 Gestion des entrées/sorties

A VOIR

3.2.1.5.5 Gestion comportementale

A VOIR

3.2.2 Schéma Modules

Le diagramme ci-dessous représente l'architecture logiciel générique des logiciels "Athena", "Apollon", "Hermes" et "Zeus" des différents modules.

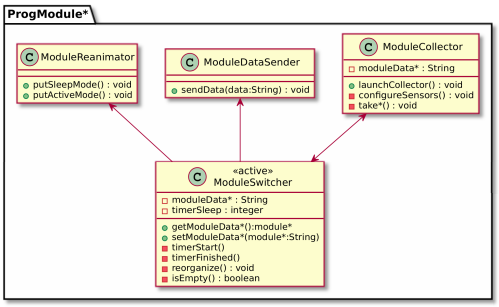

FIGURE 3.9 – Architecture logicielle générique des modules

3.2.3 Description des classes des Modules

3.2.3.1 La Classe ModuleSwitcher «Singleton»

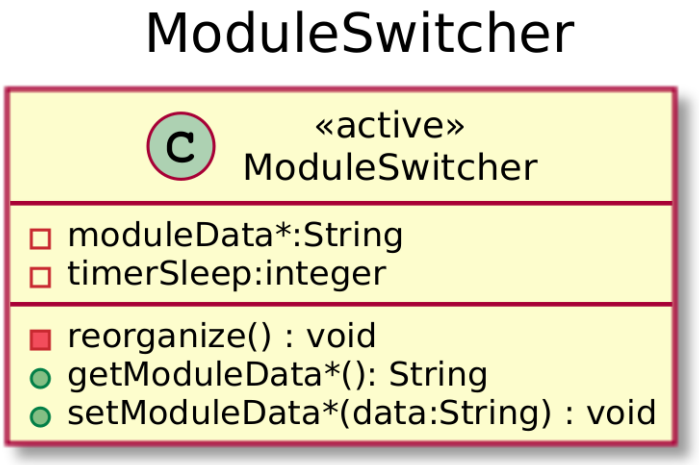

FIGURE 3.10 – Description de la classe ModuleSwitcher.

3.2.3.1.1 Philosophie de conception

La classe ModuleSwitcher est la classe du noeuds de capteurs responsable de la machine à états des noeuds de capteurs. ModuleSwitcher gère les différentes actions effectués par les modules.

Cette classe se distingue sous la forme d'un Singleton. Elle se décline en 4 formes en fonctions du noeud de capteurs dans lequel elle opère : ThermosSwitcher dans Athéna, LuminosSwitcher dans Apollon, AerosSwitcher dans Hermes et PrecipitosSwitcher dans Zeus.

Cette classe se distingue sous la forme d'un Singleton.

Reponsabilités :

— Initialiser les classes du module.
— Gèrer la machine à états du module en appelant les méthodes des autres classes.

Collaborateurs :

— ModuleDataReceiver
— ModuleCollector
— ModuleDataSender
— ModuleReanimator

3.2.3.1.2 Description structurelle

3.2.3.1.2.1 Attributs

— dataSend : String : les informations recupérés par le module une fois mises en forme.
— TIMER_SLEEP : int : le temps de sommeil du module.
— A VOIR

3.2.3.1.2.2 Services offerts

— INIT ?
— DESTROY ?
— reorganize() : mets en forme l'ensemble des informations reçus pour pouvoir les envoyer ensuite.
— Accesseurs et mutateurs des attributs module*Data, nucleusPicture et nucleusGPS.

3.2.3.1.3 Gestion du multitâche

A VOIR

3.2.3.1.4 Gestion des entrées/sorties

A VOIR

3.2.3.1.5 Gestion comportementale

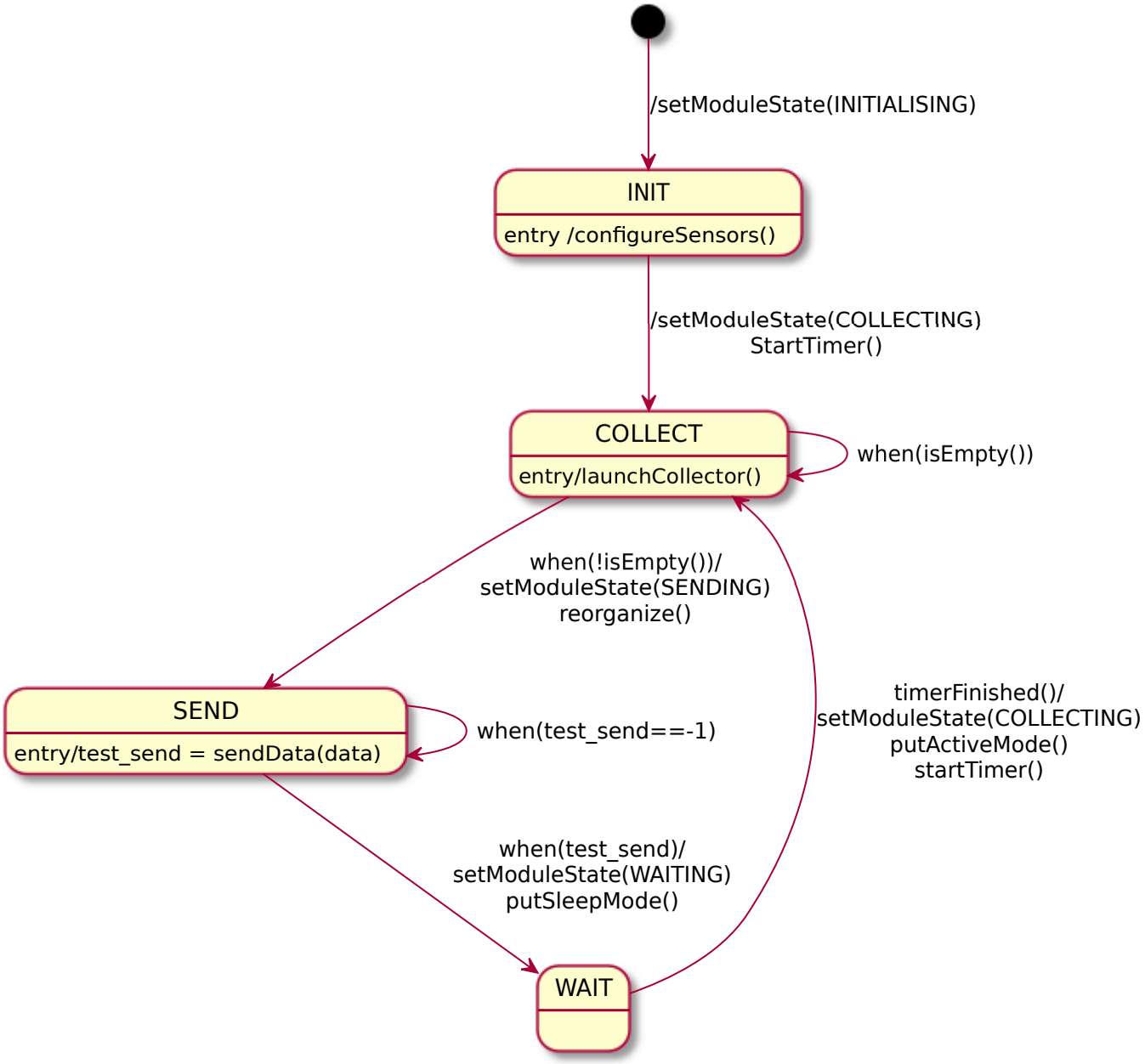

FIGURE 3.11 – Description de la machine à état d'un module.

3.2.3.2 La Classe ModuleCollector «Singleton»

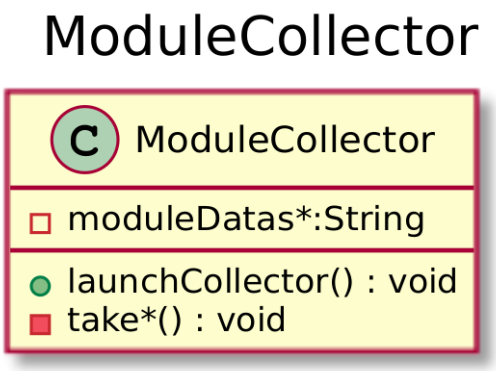

FIGURE 3.12 – Description de la classe ModuleCollector.

3.2.3.2.1 Philosophie de conception

La classe ModuleCollector est la classe du noeud de capteurs responsable de la collecte des données environnementales.

Cette classe se distingue sous la forme d'un Singleton. Elle se décline en 4 formes en fonctions du noeud de capteurs dans lequel elle opère : ThermosCollector dans Athéna, LuminosCollector dans Apollon, AerosCollector dans Hermes et PrecipitosCollector dans Zeus.

Reponsabilités :

Pour ThermosCollector :

— Relever la température.

— Relever l'humidité.

— Relever la pression.

Pour LuminosCollector :

— Relever la luminosité.

— Relever l'UV.

Pour AerosCollector :

— Relever le taux de dioxyde de carbone.

— Relever le taux de monoxyde de carbone.

— Relever le taux de d'ozone.

— Relever le taux de dioxyde d'azote.

— Relever le taux de d'ammoniac.

Pour PrecipitosCollector :

— Relever la précipitation.

— Relever la vitesse du vent.

— Relever la direction du vent.

— Relever les particules.

Collaborateur :

— ModuleSwitcher associé.

3.2.3.2.2 Description structurelle

3.2.3.2.2.1 Attributs Pour ThermosCollector :

— thermosTemperature : float : la température
— thermosHumidity : float : l'humidité.
— thermosPressure : float : la pression.

Pour LuminosCollector :

— luminosBrightness : float : la luminosité.
— luminosUV : float : l'UV.

Pour AerosCollector :

— aerosCO2 : float : le taux de dioxyde de carbone.
— aerosCO : float : le taux de monoxyde de carbone.
— aerosO3 : float : le taux de d'ozone.
— aerosNO2 : float : le taux de dioxyde d'azote.
— aerosNH3 : float : le taux de d'ammoniac.

Pour PrecipitosCollector :

— precipitosPrecipitations : float : la précipitation.
— precipitosWindSpeed : float : la vitesse du vent.
— precipitosWindDirection : Direction : la direction du vent.
— precipitosParticules : float : le taux de particules fines dans l'air.

3.2.3.2.2.2 Services offerts Global :

— INIT ?
— DESTROY ?
— Accesseurs et mutateurs des attributs.

Pour ThermosCollector :

— takeTemperature : float : Releve la température
— takeHumidity : float : Releve l'humidité.
— takePressure : float : Releve la pression.

Pour LuminosCollector :

— takeBrightness : float : Releve la luminosité.
— takesUV : float : Releve l'UV.

Pour AerosCollector :

— takeCO2 : float : Releve le taux de dioxyde de carbone.
— takeCO : float : Releve le taux de monoxyde de carbone.
— takeO3 : float : Releve le taux de d'ozone.
— takeNO2 : float : Releve le taux de dioxyde d'azote.
— takeNH3 : float : Releve le taux de d'ammoniac.

Pour PrecipitosCollector :

— takePrecipitations : float : Releve la précipitation.
— takeWindSpeed : float : Releve la vitesse du vent.
— takeWindDirection : Direction : Releve la direction du vent.
— takeParticules : float : Relève le taux de particules fines dans l'air.

3.2.3.2.3 Gestion du multitâche

A VOIR

3.2.3.2.4 Gestion des entrées/sorties

A VOIR

3.2.3.2.5 Gestion comportementale

A VOIR

3.2.3.3 La Classe ModuleReanimator «Singleton»

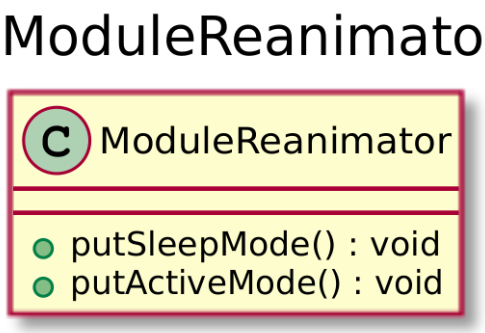

FIGURE 3.13 – Description de la classe ModuleReanimator.

3.2.3.3.1 Philosophie de conception

La classe ModuleReanimator est la classe du noeud de capteurs responsable de la mise en sommeil du système et de son reveil.

Cette classe se distingue sous la forme d'un Singleton. Elle se décline en 4 formes en fonctions du noeud de capteurs dans lequel elle opère : ThermosReanimator dans Athéna, LuminosReanimator dans Apollon, AerosReanimator dans Hermes et PrecipitosReanimator dans Zeus.

Reponsabilités :

— Endormir la module.

— Reveiller la module.

Collaborateurs :

— ModuleSwitcher associé.

— ModuleDataReceiver associé.

3.2.3.3.2 Description structurelle

3.2.3.3.2.1 Attributs

— A VOIR

3.2.3.3.2.2 Services offerts

— INIT ?

— DESTROY ?

— putSleepMode() : Mets le module en mode sleep.

— putActiveMode() : Mets le module en mode actif.

3.2.3.3.3 Gestion du multitâche

A VOIR

3.2.3.3.4 Gestion des entrées/sorties

A VOIR

3.2.3.3.5 Gestion comportementale

A VOIR

3.2.3.4 La Classe ModuleDataSender «Singleton»

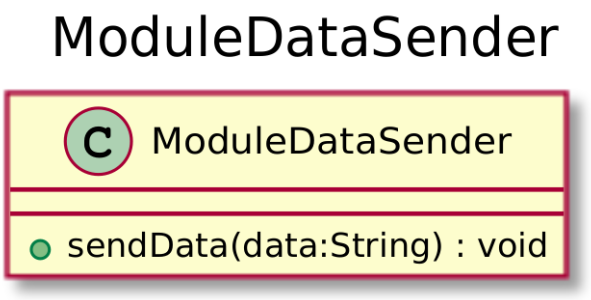

FIGURE 3.14 – Description de la classe ModuleDataSender.

3.2.3.4.1 Philosophie de conception

La classe ModuleDataSender est la classe des noeuds de capteurs responsable de la communication entre le noeud de capteur et la Nucleus. Son rôle est d'envoyer les données récoltées par ShockBurst.

Cette classe se distingue sous la forme d'un Singleton. Elle se décline en 4 formes en fonctions du noeud de capteurs dans lequel elle opère : ThermosDataSender dans Athéna, LuminosDataSender dans Apollon, AerosDataSender dans Hermes et PrecipitosDataSender dans Zeus.

Responsabilité :

— Envoyer les données à la Nucleus.

Collaborateur :

— ModuleSwitcher associé.

3.2.3.4.2 Description structurelle

3.2.3.4.2.1 Attributs

— A VOIR

3.2.3.4.2.2 Services offerts

— INIT ?

— DESTROY ?

— sendData(data : String) : Envoie l'ensemble des données relevées à la Nucleus.

3.2.3.4.3 Gestion du multitâche

A VOIR

3.2.3.4.4 Gestion des entrées/sorties

A VOIR

3.2.3.4.5 Gestion comportementale

A VOIR

3.3 Décomposition de GUI

3.3.1 Schéma MVC de la classe GUI

Dans cette partie, on représente la classe GUI avec le modèle MVC. Ce dernier respecte un modèle destiné aux interfaces graphique, qui permet notamment d'obtenir une architecture divisée en trois packages différents. Le package modèle contient les données à afficher, le package vue contient toues les vues de l'application et le package contrôleur contient la logique concernant les actions de l'utilisateur.

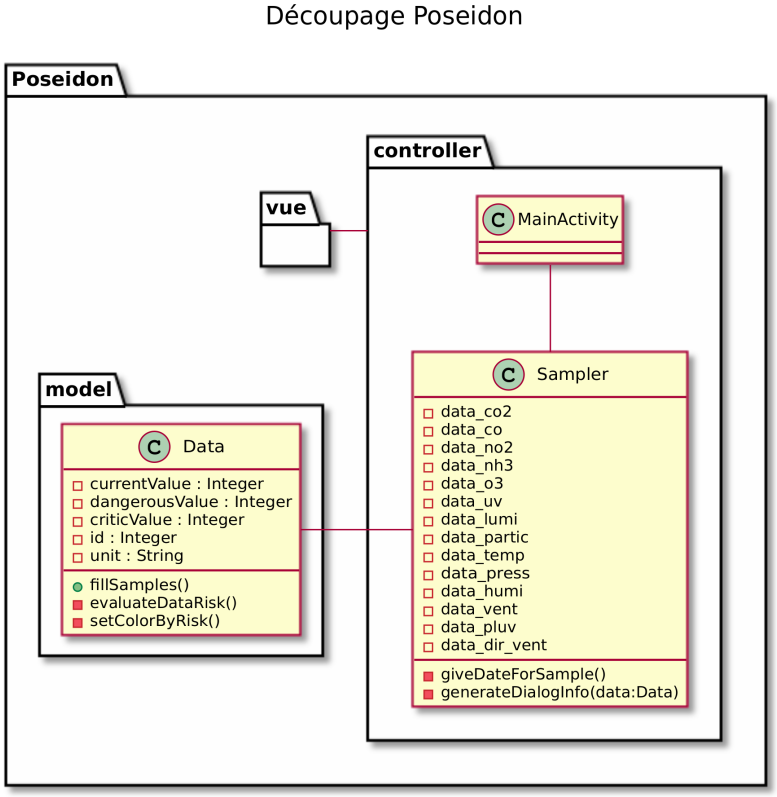

FIGURE 3.15 – Description du modèle MVC de la classe GUI

— Le package "view" représente les différents layout en "*.xml" de l'application.

— Le package "models" représente les objets métier de notre projet.

— Le package "controller" représente les activités et ses différentes classes de gestion des activités.

3.3.2 Description des classes de GUI

3.4 Protocole de communication

Les échanges entre MasterMind et MasterChef se feront par l'intermédiaire d'un réseau local créé par MasterChef et nommé MasterTest. MasterChef se comportera également comme un socket serveur auquel le socket de MasterMind viendra se connecter.

Par la suite les échanges entre les 2 parties se feront par le protocole JSON.

3.4.1 Formalisation du protocole

Chaque composant du JSON est encodée d'une manière spécifique. Ci-dessous sont définis les l'encodage des composants de façon formelle.

Chaque échange via JSON contiendra un objet "type" qui aura pour valeur un entier. Cet entier nous permettra d'adopter un Pattern Command.

Le nom des scenarios a pour format : la chaine de caractère « SCENARIO_ » suivi de la date et heure au format « YYYY_MM_DD_hh_mm_ss ».
Exemple : "SCENARIO_2017_02_22_10_05_35".

Le nom des DUT a pour format : la chaine de caractère « DUT_ » suivi du numéro du DUT. Exemple : "DUT_2".

Le nom des images correspond au nom de l'image lors de son ajout sur Mass_Storage. Exemple : "st-example-image-minimal-st-opensdk-sti-lts".

Le nom des images correspond au nom du script lors de son ajout sur Mass_Storage. Exemple : "smoke".

Les résultats de scenarios lors de l'envoi de la liste des résultats de MasterChef à MasterMind possèdent un boolean "archive". Si archive est positif, le resultat est archivé, sinon il ne l'est pas.

Lorsque l'utilisateur décide d'envoyer la demande de lancement ou d'arrêt du scenario. L'action du scenario peut alors avoir 2 valeurs : « PLAY » ou « STOP ».

Lors de l'envoie de l'état d'un scenario de MasterChef à MasterMind. L'état peut avoir 5 valeurs : « IDLE », « WAITING_FLASH »,« FLASHING », « TESTING » ou « FINISHED ».

Lors d'un retour de MasterChef à MasterMind sur un ordre envoyé, l'objet JSON possèdent un boolean "success". Si l'ordre est effectué sans erreur alors success est positif. Si MasterChef a une erreur alors success est negatif. Les ordres sont l'ajout, la suppression, jouer ou arrêter un scenario ainsi que l'archivage et la suppression d'un résultat.

Lors d'un erreur d'execution de scenario. L'objet JSON du message envoyé par MasterChef à MasterMind possède un string "operation" pouvant avoir 2 valeurs : "FLASH" ou "TEST". La valeur de operation signifie une erreur de flashage ou d'execution des tests.

Dans les prochains exemples de JSON, les données sont factices et destinées simplement à définir le protocole de communication.

3.4.2 Envoie des données depuis MasterMind

3.4.2.1 Demander la liste des DUTS

ProxyDataKeeper de MasterMind demande à DataKeeper de MasterChef la liste des DUTS.

```
{
  "type": 0
}
```

3.4.2.2 Demander l'archivage d'un résultat de scénario

ProxyDataKeeper de MasterMind demande à DataKeeper de MasterChef de sauvegarder une archive.

```
{
  "type": 1,
  "scenario": "SCENARIO_2017_05_26_14_23_32"
}
```

3.4.2.3 Demander la suppression d'un scénario

ProxyDataKeeper de MasterMind demande à DataKeeper de MasterChef de supprimer un scénario.

```
{
  "type": 2,
  "scenario": "SCENARIO_2017_05_26_14_23_32"
}
```

3.4.2.4 Stopper ou jouer un scénario

ProxyPlayer de MasterMind demande à Player de MasterChef de stopper ou jouer un scénario.

```
{
  "type": 3,
  "action": "STOP"
  "scenario": "SCENARIO_2017_05_26_14_23_32"
}
```

3.4.2.5 Ajouter un scenario

ProxyDataKeeper de MasterMind demande à DataKeeper de MasterChef d'ajouter un scénario.


```
{
  "type": 4,
  "name": "SCENARIO_2017_02_05_14_32_13",
  "dut": "DUT_2",
  "image": "IMG_EXEMPLE4",
  "scripts": ["SCRIPT_EXEMPLE3", "SCRIPT_EXEMPLE9", "
    SCRIPT_EXEMPLE2"]
}
```

3.4.2.6 Demander l'arrêt de MasterChef.

ProxyStarter de MasterMind demande à Starter de MasterChef d'arrêter le système MasterChef.

```
{
  "type": 5
}
```

3.4.2.7 Demander à MasterChef les informations de démarrage.

ProxyDataKeeper de MasterMind demande à DataKeeper de MasterChef la liste de configuration, la scenariolist et la liste des résultats.

```
{
  "type": 6
}
```

3.4.2.8 Demander la suppression d'un résultat de scénario

ProxyDataKeeper de MasterMind demande à DataKeeper de MasterChef de supprimer un résultat de scénario.

```
{
  "type": 7,
  "scenario": "SCENARIO_2017_05_26_14_23_32"
}
```


3.4.3 Envoie des données depuis MasterChef

3.4.3.1 Envoyer la liste de configuration

ProxyTestCampaign de MasterChef demande à TestCampaign de MasterMind de mettre à jour la liste des duts, scripts et images.

```
{
  "type": 0,
  "scripts": ["SCRIPT_EXEMPLE2", "SCRIPT_EXEMPLE5", "
    SCRIPT_EXEMPLE10", "SCRIPT_EXEMPLE15"],
  "images": ["IMG_EXEMPLE2", "IMG_EXEMPLE23", "IMG_EXEMPLE5", "
    IMG_EXEMPLE12"],
  "duts": ["DUT_EXEMPLE1", "DUT_EXEMPLE_2"]
}
```

3.4.3.2 Envoyer la scenariolist

ProxyTestCampaign de MasterChef demande à TestCampaign de MasterMind de mettre à jour la scenariolist.

```
{
  "type": 1,
  "playlist": [{
    "name": "SCENARIO_2017_02_05_14_32_13",
    "state": "FLASHING",
    "dut": "DUT_2",
    "image": "IMG_EXEMPLE45",
    "scripts": ["SCRIPT_EXEMPLE3", "SCRIPT_EXEMPLE4", "
      SCRIPT_EXEMPLE10"]
  }, {
    "name": "SCENARIO_2017_05_35_14_31_13",
    "state": "WAITING",
    "dut": "DUT_2",
    "image": "IMG_EXEMPLE2",
    "scripts": ["SCRIPT_EXEMPLE1", "SCRIPT_EXEMPLE2"]
  }],
  "standbylist": [{
    "name": "SCENARIO_2017_02_15_14_52_13",
    "dut": "DUT_2",
    "image": "IMG_EXEMPLE45",
    "scripts": ["SCRIPT_EXEMPLE3", "SCRIPT_EXEMPLE4", "
      SCRIPT_EXEMPLE10"]
  }]
}
```


3.4.3.3 Envoyer la liste des résultats

ProxyTestCampaign de MasterChef demande à TestCampaign de MasterMind de mettre à jour la liste des résultats de l'ensemble des scenarios finis de la session courrante.

```
{
  "type": 2,
  "results": [{
    "name": "SCENARIO_2017_05_35_14_31_13",
    "dut": "DUT_2",
    "image": "IMG_EXEMPLE2",
    "archive": false,
    "scripts": [{
      "name": "SCRIPT_EXEMPLE1",
      "results": [{
        "name": "TEST_EXEMPLE1",
        "result": "PASS"
      }, {
        "name": "TEXT_EXEMPLE2",
        "result": "FAIL"
      }, {
        "name": "TEXT_EXEMPLE3",
        "result": "FAIL"
      }],
      "log": "Lorem ipsum dolor sit amet, consectetur adipiscing
         elit, sed do eiusmod tempor incididunt ut labore et
         dolore magna aliqua."
    }]
  }, {
    "name": "SCENARIO_2017_05_12_14_31_13",
    "dut": "DUT_1",
    "image": "IMG_EXEMPLE2",
    "archive": true,
    "scripts": [{
      "name": "SCRIPT_EXEMPLE1",
      "results": [{
        "name": "TEST_EXEMPLE1",
        "result": "PASS"
      }, {
        "name": "TEXT_EXEMPLE2",
        "result": "FAIL"
      }, {
        "name": "TEXT_EXEMPLE3",
        "result": "FAIL"
      }],
      "log": ""
    }]

  }]
}
```


3.4.3.4 Envoyer la liste des DUTS

ProxyTestCampaign de MasterChef demande à TestCampaign de MasterMind de mettre à jour la liste des duts.

```
{
  "type": 3,
  "duts": ["DUT_EXEMPLE1", "DUT_EXEMPLE_2"]
}
```

3.4.3.5 Envoyer les résultats d'un scénario

ProxyTestCampaign de MasterChef demande à TestCampaign de MasterMind de mettre à jour les résultats d'un scenario.

```
{
  "type": 4,
  "scenario": "scenario_2017_02_05_14_32_13",
  "results": [{
    "script": "SCRIPT_EXEMPLE3",
    "results": [{
      "name": "PING",
      "result": "PASS"
    }, {
      "name": "CPU",
      "result": "FAIL"
    }, {
      "name": "ETH",
      "result": "FAIL"
    }],
    "log": "Lorem ipsum dolor sit amet, consectetur adipiscing
       elit, sed do eiusmod tempor incididunt ut labore et dolore
        magna aliqua."
  }, {
    "script": "SCRIPT_EXEMPLE4",
    "results": [{
      "name": "TRY",
      "result": "PASS"
    }, {
      "name": "CATCH",
      "result": "PASS"
    }, {
      "name": "SWITCH",
      "result": "PASS"
    }],
    "log": ""
  }]
}
```


3.4.3.6 Envoyer une erreur d'espace mémoire

ProxyGUI de MasterChef demande à GUI de MasterMind d'afficher un dialog d'erreur d'espace mémoire.

```
{
  "type": 5
}
```

3.4.3.7 Envoyer un retour sur l'ajout d'un scénario

ProxyGUI de MasterChef demande à GUI de MasterMind d'afficher un dialog de confirmation ou d'echec de l'ajout d'un scénario.

```
{
  "type": 6,
  "scenario": "SCENARIO_2017_05_02_23_22_23",
  "success": false
}
```

3.4.3.8 Envoyer un retour sur l'archivage d'un test

ProxyGUI de MasterChef demande à GUI de MasterMind d'afficher un dialog de confirmation ou d'echec de l'archivage d'un test.

```
{
  "type": 7,
  "scenario": "SCENARIO_2017_05_02_23_22_23",
  "success": true
}
```

3.4.3.9 Envoyer un retour sur la suppression d'un scénario

ProxyGUI de MasterChef demande à GUI de MasterMind d'afficher un dialog de confirmation ou d'echec de suppression d'un scénario ou résultat de scénario.

```
{
  "type": 8,
  "scenario": "SCENARIO_2017_05_02_23_22_23",
  "success": true
}
```

3.4.3.10 Envoyer l'état d'un scénario

ProxyGUI de MasterChef demande à GUI de MasterMind d'afficher un dialog de confirmation de suppression d'un scénario ou résultat de scénario.

```
{
  "type": 9,
  "scenario": "SCENARIO_2017_05_02_23_22_23",
  "state": "FLASHING"
}
```


3.4.3.11 Envoyer une confirmation de la demande d'arrêt

ProxyGUI de MasterChef demande à GUI de MasterMind d'afficher un dialog de confirmation d'extinction de MasterChef.

```
{
  "type": 10
}
```

3.4.3.12 Envoyer un retour sur le lancement ou l'arrêt d'un scénario

ProxyGUI de MasterChef demande à GUI de MasterMind d'afficher un dialog de confirmation ou d'echec de lancement/arrêt d'un scénario ou résultat de scénario.

```
{
  "type": 11,
  "scenario": "SCENARIO_2017_05_02_23_22_23",
  "action": "PLAY",
  "success": true
}
```

3.4.3.13 Envoyer un retour sur la suppression du résultat d'un scenario

ProxyGUI de MasterChef demande à GUI de MasterMind d'afficher un dialog de confirmation ou d'echec de suppression d'un résultat scénario ou résultat de scénario.

```
{
  "type": 12,
  "scenario": "SCENARIO_2017_05_02_23_22_23",
  "success": true
}
```

3.4.3.14 Envoyer une erreur d'execution

ProxyGUI de MasterChef demande à GUI de MasterMind d'afficher un dialog d'erreur d'execution.

```
{
  "type": 13,
  "scenario": "SCENARIO_2017_05_02_23_22_23",
  "operation": "FLASH"
}
```


Chapitre 4

Dictionnaire du Domaine

Ce dictionnaire du domaine s'ajoute au dictionnaire du domaine de la spécification, qui défini davantage de termes.
Activer un DUT : MasterPower alimente et allume le DUT nécessaire.

Activité : Élément Android représentant un écran. Une activité contient tout les éléments de l'écran comme un bouton, un message, etc. Elle représente donc ce que l'utilisateur peut faire durant son exécution.

Adresse IP : C'est un numéro d'identification propre à chaque appareil connecté à un réseau informatique. On utilise le protocole IPv4.

Adresse MAC : Adresse unique constituée de 6 octets variant de 0 à 255 bits pour toutes les cartes électronique. Cette adresse est fixée par le constructeur afin de d'identifier de façon unique une carte sur un réseau local. C'est avec cette adresse que MasterPower va identifier les DUT branchés.

Buffer de lecture : Mémoire tampon dite provisoire utilisée durant une communication entre MasterMind et MasterChef.

Connexion serial : Transmission de données dans laquelle les éléments d'information se succèdent.

Côté client : MasterMind correspond au client de la communication entre MasterMind/MasterChef.

Côté server : MasterChef correspond au server de la communication entre MasterMind/MasterChef.

Démarrer correctement MasterMind : StarterMM allume MasterMind et affiche l'écran « Home_Settings ».

Données de configuration pour la connexion : Correspond à la chaînes de caractère représentant l'IP et le Port d'une connexion.

DUT branchés : Les cartes sont alimentés et branchés.

Fragment : Élément Android représentant des sections d'écrans. Il permet de scinder une

activité en plusieurs fragments ayant chacun sa propre vue. Il permet une meilleur adaptation des différentes tailles d'écran par exemple.

Handler : Processus associé à une activité dans le but de travailler avec l'UIThread.

Informer du déroulement du flashage : Correspond à l'affichage d'une popUp contenant une barre de progression pour prévenir l'utilisateur de l'état d'avance du flashage .

JSON : (JavaScript Object Notation) Est un format de données textuelles, il permet de représenter de l'information de façon structurée.

Kermit : Programme de communication à multiple usage utilisé sous Linux.

MasterChef est prêt à fonctionner : MasterChef est allumé et le WiFi est généré.

Packages : Correspond au regroupement de plusieurs classes pour organiser et établir une hiérarchie dans développement C et Android.

Paramètres de connexion par défaut : Paramètres définis au préalable par les développeurs C permettant une connexion à la valise presque automatique. L'utilisateur n'a pas besoin de rentrer l'adresse IP et le port à chaque perte de connexion. Il a juste à valider ces paramètres.

Pattern command : est un patron de conception de type comportemental qui encapsule la notion d'invocation. Il permet de séparer complètement le code initiateur de l'action, du code de l'action elle-même.
Playlist : Correspond à la liste des scénarios allant être jouées les uns après les autres.

Port : Numéro correspondant à la couche de transport du modèle OSI, la notion de port logiciel permet, sur un ordinateur donné, de distinguer différents interlocuteurs.

Proxy : Classe de substitution permettant de simuler l'appel des méthodes d'une classe d'un autre système.

Script : Un script est un programme en langage interprété.

Script kermit : Fichier contenant les commandes utiles à l'utilisation du kermit sous Linux.

Scruter un DUT : MasterPower alimente et allume les DUT pour obtenir la liste des DUT branchés.

Shell "Unix" : C'est un interpréteur de commandes destiné aux systèmes d'exploitation Unix.

Socket : modèle permettant la communication entre deux éléments (ici MasterMind et MasterChef) à travers un réseau TCP/IP. Il permet l'envoi et la réception de données entre les deux éléments : un server et un client.

StandByList : Correspond à la liste des scénarios créés mais qui ne sont pas en attente d'être joués.

Stopper un scénario : Il est possible de stopper un scénario en cours dans la playList. Ce

scénario est alors placé dans la StandbyList et réinitialisé. Autrement dit si l'utilisateur veut relancer ce scénario, il se refera entièrement du début.

Thread : Fil d'exécution dans un programme.

Timeout : Définition d'un temps pour lequel une exception est levée.

UART : (Universal Asynchronous Receiver Transmitter) C'est un émetteur-récepteur asynchrone universel.

UIThread : Fil d'exécution qui donne la priorité pour l'affichage graphique d'une application.

Table des figures

Liste des tableaux