



Le génie pour l'industrie

RAPPORT TECHNIQUE
PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
DANS LE CADRE DU COURS LOG795/GTI795
PROJET DE FIN D'ÉTUDES EN GÉNIE LOGICIEL ET DES TI

PREUVE DE CONCEPT D'IOT AVEC THINGWORX

Échange de données IOT dans l'industrie 4.0

MARC-ANTOINE VÉZINA - VEZM11129004
FRANÇOIS MARCHAND - MARF17049209
RENA LAKISS - LAKR07609408
ÉMILE ROUILLARD - ROUE10079403
ALEXANDRE TRÉPANIÉ - TREA26029307

DÉPARTEMENT DE GÉNIE LOGICIEL ET DES TI

Professeur-superviseur

ALAIN APRIL

MONTREAL, AVRIL 2019
HIVER 2019

REMERCIEMENTS

L'équipe de projet tient à remercier l'École de Technologie Supérieure pour les enseignements précieux et les opportunités de projet de fin d'études offert aux étudiants.

L'équipe tient également à remercier Réjean Ouellet et Cédric Melançon de chez Matricis, sans qui le projet n'aurait pas eu lieu. Leur accueil, leur savoir et leur support tout au long du projet ont contribué à la réalisation de celui-ci.

L'équipe tient tout autant à remercier monsieur Alain April, sans qui ce projet n'aurait pas eu lieu, pour ses précieux contacts dans l'industrie et son encadrement lors du projet.

Finalement, l'équipe tient à remercier François Blackburn-Grenon, étudiant au doctorat en génie industriel à l'École de technologie supérieure, pour son support, ses suivis, et la gestion du projet.

PREUVE DE CONCEPT D'IOT AVEC THINGWORX

Échange de données IOT dans l'industrie 4.0

MARC-ANTOINE VÉZINA - VEZM11129004

FRANÇOIS MARCHAND - MARF17049209

RENA LAKISS - LAKR07609408

ÉMILE ROUILLARD - ROUE10079403

ALEXANDRE TRÉPANIÉ - TREA26029307

RÉSUMÉ

Dans le cadre du cours LOG795/GTI795, l'équipe composée de cinq étudiants a eu le mandat de mettre en œuvre une preuve de concept d'IoT à l'aide du progiciel ThingWorx lors du projet de fin d'études (Pfe). Avec l'aide du client Matricis et de son équipe de développement, l'équipe de Pfe a implémenté trois sous-systèmes différents : 1) une application en réalité augmentée qui permet d'afficher aux utilisateurs les données relatives à l'état d'un objet en temps réel en superposant une image sur l'écran de l'appareil; 2) une application de bureau servant à simuler un objet connecté, et finalement; 3) la mise en place des éléments nécessaires sur la plateforme ThingWorx (de l'entreprise PTC) qui sert d'intersection entre tous ces sous-systèmes et qui contient les interfaces utilisateurs de cette preuve de concept.

Ce projet consiste principalement à récupérer des données englobant le cycle de vie complet d'un turboréacteur, permettant d'améliorer sa maintenance prédictive et ainsi de prédire sa durée de vie utile. L'injection des données peut s'effectuer de manière manuelle, aléatoire ou par la lecture d'un fichier de données. L'application de bureau, contenant le simulateur, peut également être lancée dans plusieurs instances afin de simuler un volume plus important de trafic provenant de plusieurs objets connectés. À partir du module de réalité augmentée, il est possible de visualiser l'état actuel de l'objet obtenu par les données en scannant un marqueur apposé sur celui-ci, et ce, à l'aide d'un appareil mobile doté d'une caméra. Il est aussi possible de manipuler l'état de l'objet en changeant les paramètres disponibles dans le simulateur. Finalement, l'ensemble des données et l'état actuel de l'objet peuvent être visualisés et supervisés à l'aide de la plateforme ThingWorx qui est disponible aux utilisateurs facilitant ainsi la maintenance préventive du turboréacteur. Malgré quelques embûches surmontées au cours de la période de conception, cette preuve de concept, a permis l'expérimentation de diverses technologies IoT.

TABLE DES MATIÈRES

LISTE DES TABLEAUX	5
LISTE DES FIGURES	6
LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES	8
LISTE DES SYMBOLES ET UNITÉS DE MESURE	9
CHAPITRE 1 - INTRODUCTION	10
CHAPITRE 2 - OBJECTIFS	11
2.1 Présentation du promoteur	11
2.2 Besoins du promoteur	11
2.3 Architecture existante	11
2.4 Mandat du projet	12
2.5 Composition de l'équipe	12
2.6 Livrables	14
2.6.1 Simulateur	14
2.6.2 Réalité augmentée	15
2.6.3 Apprentissage Prédicatif	15
2.6.4 Modélisation dans ThingWorx	15
2.6.5 Interfaces et Alertes	16
2.6.6 Rapport de synthèse (présent rapport)	16
2.6.7 Présentations de projet	16
2.7 Risques	16
CHAPITRE 3 - MÉTHODOLOGIE DE TRAVAIL	18
3.1 Gestion de projet	18
3.3 Suivi de projet	18
3.4 Outils et technologies utilisées	19
3.4.1 ThingWorx	19
3.4.2 Unity + Vuforia	20
3.4.3 ThingWorx Edge C SDK	20
3.3.4 Boost	20
3.3.5 Qt	21
3.3.6 CMake	21
3.3.7 Plugiciel ThingWorx pour Eclipse et le SDK d'extension	21
CHAPITRE 4 - CONCEPTION	22
4.1 Architecture globale	22
4.2 Simulateur	24
4.2.1 Module config	25
4.2.2 Module ui	25
4.2.3 Module simulator	27

4.2.4	Module connector	28
4.2.5	Module utils	28
4.3	Réalité augmentée	29
4.3.1	Modèle	29
4.3.2	Services	30
4.3.3	Configurations	30
4.3.4	Logique de rafraîchissement des données	31
4.3.5	Logique d'affichage	33
4.4	Modélisation dans ThingWorx	35
4.5	Interfaces et Alertes	36
CHAPITRE 5 - INTÉGRATION ET RÉSULTATS		37
5.1	Architecture de référence	37
5.2	Simulateur	38
5.3	Réalité augmentée	44
5.4	Modélisation dans ThingWorx	45
5.5	Interfaces et Alertes	47
CHAPITRE 6 - PROBLÈMES RENCONTRÉS		55
6.1	Manque de connaissance global sur ThingWorx	55
6.2	Licences ThingWorx et Serveur Analytics	55
6.3	Configuration et Compatibilité	56
6.4	Contraintes de temps	56
6.5	Disponibilité des équipements	57
CHAPITRE 7 - AMÉLIORATIONS ET TRAVAUX FUTURS		58
7.1	Apprentissage Prédicatif	58
7.2	Améliorations au simulateur	59
7.3	Réalité augmentée	59
7.4	Interfaces et alertes	60
CHAPITRE 8 - CONCLUSION		61
RÉFÉRENCES		62
BIBLIOGRAPHIE		63
ANNEXE 1 - Diagrammes		64

LISTE DES TABLEAUX

<u>Tableau</u>	<u>Titre</u>	<u>Page</u>
Tableau 1	Liste des abréviations, sigles et acronymes	9
Tableau 2	Liste des symboles et unités de mesure	10
Tableau 2.1	Composition de l'équipe de projet	13
Tableau 2.2	Risques à la réalisation du projet	17

LISTE DES FIGURES

<u>Figure</u>	<u>Titre</u>	<u>Page</u>
Figure 2.1	Représentation des rôles de l'équipe	15
Figure 4.1	Architecture à haut niveau de la solution	24
Figure 4.2	Diagramme de classe du Simulateur	25
Figure 4.3.1	Diagramme de séquence de la configuration de l'application	32
Figure 4.3.2	Diagramme de séquence de la détection d'une cible (partie 1)	32
Figure 4.3.4	Diagramme de séquence de la détection d'une cible (partie 2)	34
Figure 4.3.5	Premier mock up avec un menu épinglé au bas de l'écran	35
Figure 4.4.1	Découpage de la turbine en composants	36
Figure 4.4.2	Conception de l'engin pour l'intégration dans ThingWorx.	36
Figure 5.1	Architecture de référence des composants	38
Figure 5.2.1	Vue principale du simulateur	39
Figure 5.2.2	Vue de l'onglet Server du simulateur	40
Figure 5.2.3	Vue de l'onglet Thing du simulateur	41
Figure 5.2.4	Vue de l'onglet Simulation du simulateur	41
Figure 5.2.5	Vue Simulation du simulateur en mode manuel	42
Figure 5.2.6	Vue Simulation du simulateur en mode Random	43
Figure 5.2.7	Vue Simulation du simulateur en mode File	43
Figure 5.2.8	Vue du graphique avec le simulateur en mode aléatoire	44
Figure 5.3	Vue de l'interface du module de réalité augmentée	45
Figure 5.4	Résultat de la structure sur ThingWorx	47

Figure 5.5	Interface de l’outil de mashups de ThingWorx	48
Figure 5.5.1	Mashup Master	49
Figure 5.5.2	Configuration du menu de navigation	49
Figure 5.5.3	Mashup tableau de bord	50
Figure 5.5.4	Mashup assets	51
Figure 5.5.5	Éditeur de texte pour services	52
Figure 5.5.6	Menu des services pour Mashup	52
Figure 5.5.7	Menu des liens d’un élément	53
Figure 5.5.8	Éditeur de texte pour styles	53
Figure 5.5.9	Import de fichier css	54

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

Acronymes	Définitions
API	Interface de Programmation Applicative (<i>Application Programming Interface</i>)
CSS	Feuille de style en cascade (<i>Cascading Style Sheets</i>)
ÉTS	École de technologie supérieure
HTTP	Protocole de Transfert Hypertexte (<i>Hypertext Transfer Protocol</i>)
IDE	Environnement de développement intégré (<i>Integrated development environment</i>)
IoT	Internet des objets (<i>Internet of Things</i>)
IIoT	Internet des objets industriel (<i>Industrial Internet of Things</i>)
MVP	Produit minimum viable (<i>Minimal viable product</i>)
RA / AR	Réalité augmentée (<i>Augmented Reality</i>)
ReST	Transfert d'État Représentationnel (<i>Representational State Transfer</i>)
SDK	Kit de développement logiciel (<i>Software Development Kit</i>)
SMS	Messagerie texte (<i>Short Message Service</i>)
URI	Identifiant de ressource uniforme (<i>Uniform Resource Identifier</i>)
WYSIWYG	<i>What You See Is What You Get</i>

Tableau 1 : Liste des abréviations, sigles et acronymes

LISTE DES SYMBOLES ET UNITÉS DE MESURE

Symboles / Unités	Définitions
°r	Degré Rankine, échelle de mesure de température ¹
rpm	Tour par minute (« <i>revolution per minutes</i> »)
psia	Livre-force par pouce carré (« <i>pound-force per square inch</i> »)
lbm/s	Livres par secondes (« <i>pound mass per second</i> »)
pps	Pulsions par secondes (« <i>pulse per second</i> »)

Tableau 2 : Liste des symboles et unités de mesure

¹ Source: Fr.wikipedia.org. (2019). *Échelle Rankine*. [En ligne] Disponible sur: https://fr.wikipedia.org/wiki/Échelle_Rankine [Accédé le 18 janvier. 2019].

CHAPITRE 1 - INTRODUCTION

L'Internet des Objets (IoT) est le concept d'objets connectés à l'internet. Il comprend entre autres, mais ne se limite pas à: des ordinateurs et des téléphones mobiles. Avec la popularité récente de la domotique ces dernières années, communément appelée "maisons intelligentes", plusieurs nouveaux appareils sont maintenant connectés. Par exemple, les systèmes de chauffage et d'éclairage. Avec la croissance actuelle de ces objets connectables, il est estimé que 80 milliards d'appareils seront connectés en 2025. L'attrait de l'internet des objets est de collecter des données de ces différents objets et à l'aide de cette information créer une intelligence afin de mieux automatiser et interagir avec ceux-ci.

Dans ce contexte, l'entreprise Matricis a proposé à l'équipe un projet exploratoire visant à utiliser les principes de l'IoT dans un contexte industriel. On parle d'industrie 4.0 lorsqu'il est question des lignes de productions automatisées et intelligentes. Par contre, l'intellectualisation des opérations, dont la maintenance des appareils de la chaîne de montage et des engins servant au transport, peut certainement devenir un sujet important à traiter. Cette prochaine étape d'informatisation dans cette industrie consiste à virtualiser les différents appareils de production, entre autres afin de mieux analyser le travail fait et pouvoir prévenir certains problèmes dans la chaîne de montage.

Ce rapport présente, dans un premier temps, les objectifs du projet, son contexte d'affaires ainsi que les problématiques et les enjeux. Dans cette section seront également présentés les différents livrables, les membres de l'équipe et une évaluation des différents risques liés au mandat de projet. Ensuite, une présentation de la méthodologie de travail et des technologies utilisées est effectuée. Cette section est suivie d'une description des décisions de conception des différents modules. Les résultats de l'intégration des modules sont ensuite présentés avant de conclure par une courte description des problèmes rencontrés et surmontés par l'équipe lors du projet et, finalement, une courte discussion des travaux futurs possibles afin de bonifier ce prototype expérimental développé à l'avenir.

CHAPITRE 2 - OBJECTIFS

Ce chapitre porte sur la description de la problématique d'affaires et du prototype expérimental conçu et développé au cours de ce projet. La présentation du contexte passera tout d'abord par une description du promoteur, ses besoins, ainsi que les éléments mis en place pour démarrer le projet. Une brève présentation de l'équipe de travail suit, accompagnée d'une description à haut niveau des différents livrables. Pour terminer, les risques identifiés en début de projet sont énumérés.

2.1 Présentation du promoteur

Matricis est une compagnie œuvrant dans l'intégration des données d'entreprise, l'internet des objets, l'intelligence d'affaires, l'analyse de données et l'efficacité opérationnelle. Depuis plus de 20 ans, Matricis aide ses clients à tirer pleine valeur de leurs données en offrant des solutions d'intégration et des produits connectés, pouvant être hébergés sur le nuage ou directement chez le client.

2.2 Besoins du promoteur

Les besoins de Matricis relèvent autant de l'exploration technologique de l'IoT que l'objectif d'avoir d'un produit de démonstration des possibilités de cette technologie émergente. La compagnie veut mettre de l'avant la plateforme IoT ThingWorx dans ses solutions d'intégration avec ses clients industriels. Pour ce faire, l'entreprise aimerait posséder un prototype complet et intégré afin d'effectuer des présentations plus pratiques du fonctionnement de l'IoT. De ce fait, l'entreprise a besoin d'aide pour rassembler et faire une preuve de concept impliquant les éléments matériels et logiciels de ce prototype. Les éléments mécaniques et électroniques du prototype furent d'ailleurs développés en parallèle par une autre équipe d'étudiants de l'école de technologie supérieure en génie des systèmes.

2.3 Architecture existante

En termes d'architecture logicielle, au tout début du projet, les composants existants étaient quasiment nuls. Quelques croquis d'interfaces avaient été réalisés préalablement concernant des tableaux de bord possibles et différentes fenêtres de l'interface graphique souhaitées du

simulateur IoT recherché. De plus, un graphique de vue d'ensemble de l'interconnexion des modules avait été produit (voir **figure 4.1**). Une version en développement d'un « wrapper » en langage de programmation C++ avait également été fourni afin d'être utilisé conjointement avec le simulateur dans le but de rendre accessibles certaines fonctionnalités. Des outils de gestion de projet et de contrôle de version avaient également été mis en place sur le service infonuagique Azure de Microsoft. Ainsi, l'équipe de développement avait accès au Wiki du projet, aux répertoires de contrôle de versions pour les différents modules, ainsi qu'au tableau de Kanban de projet sur lequel plusieurs tâches avaient été préalablement créées/listées.

2.4 Mandat du projet

Lors d'une précédente démo de la compagnie PTC, tenue dans les bureaux de Matricis, un projet intégré représentant une valise dans laquelle se trouvait un petit moteur doté de plusieurs senseurs avait été présenté afin de démontrer les capacités du produit ThingWorx. La présentation consistait à scanner un marqueur sur le moteur afin d'afficher les données en temps réel. Le mandat du projet actuel consiste à développer une solution similaire en utilisant les outils d'analyse avancés de ThingWorx afin de faire de la maintenance prédictive. Ainsi, le projet comporte le développement de plusieurs éléments logiciels qui devront communiquer entre eux afin de partager de l'information relative à l'objet ciblé par le système. Une description plus détaillée des différents composants est disponible à la section 2.6 du rapport.

2.5 Composition de l'équipe

L'équipe d'étudiants a été formée préalablement au choix du projet et avant de contacter le professeur Alain April afin d'obtenir des contacts pour un projet pratique en l'industrie. Un projet d'IOT visant à créer une interface de données a alors été proposé à l'équipe.

Lors de la première rencontre de lancement du projet, une brève présentation des parties prenantes et des éléments et livrables du projet fut effectuée. Lors de cette rencontre, les différents composants du projet furent également attribués aux membres de l'équipe selon leurs expériences et habiletés et la méthodologie quant à la gestion de projet fut décidée. Le tableau récapitulatif suivant définit le rôle attribué à chacun des membres de l'équipe de développement.

Membre	Poste	Responsabilités
François Beaubien	Promoteur	<ul style="list-style-type: none"> - Dédier des ressources au projet.
Réjean Ouellet	Propriétaire du projet	<ul style="list-style-type: none"> - Émettre les requis fonctionnels - Clarifier les exigences - Mettre les outils et infrastructures nécessaires à la disposition de l'équipe - Assurance qualité - Intégration, acceptation
Cédric Melançon	Architecte de solution	<ul style="list-style-type: none"> - Supporter l'équipe de développement - Effectuer les révisions de code et approuver les changements - Assurance qualité - Intégration, acceptation
François Blackburn-Grenon	Gestionnaire de projet	<ul style="list-style-type: none"> - Assurer le suivi des tâches et de l'avancement global du projet
Alain April	Responsable de la partie LOG du PFE	<ul style="list-style-type: none"> - Superviser les activités de développement logiciel - Supporter l'équipe - Évaluation du projet
François Marchand	Développeur C++	<ul style="list-style-type: none"> - Développer le simulateur et y effectuer les ajouts fonctionnels. - Assurance qualité - revue de code/design
Rena Lakiss	Développeur C++	<ul style="list-style-type: none"> - Développer le simulateur et y effectuer les ajouts fonctionnels. - Assurance qualité - revue de code/design
Émile Rouillard	Développeur Unity/C#, C++	<ul style="list-style-type: none"> - Développer une interface en réalité augmentée afin d'afficher les données sur Vuforia - Assurance qualité - revue de code/design
Marc-Antoine Vézina	Développeur Unity/C#, C++ Admin ThingWorx	<ul style="list-style-type: none"> - Administrer les accès sur la plateforme ThingWorx. - Modéliser les éléments requis sur la plateforme ThingWorx - Créer les classes et fonctions nécessaires aux requêtes des données pour l'application de réalité augmentée - Implémentation de la connexion au serveur sur le simulateur. - Assurance qualité - revue de code/design
Alexandre Trépanier	Développeur ThingWorx	<ul style="list-style-type: none"> - Expérimenter sur les solutions d'analyses prédictives sur le serveur d'analyse de ThingWorx - Produire des entités mashup pour l'affichage des données sur la plateforme ThingWorx - Assurance qualité - revue de code/design

Tableau 2.1 : Composition de l'équipe de projet

La figure suivante explique également le rôle de chacun des membres de l'équipe.

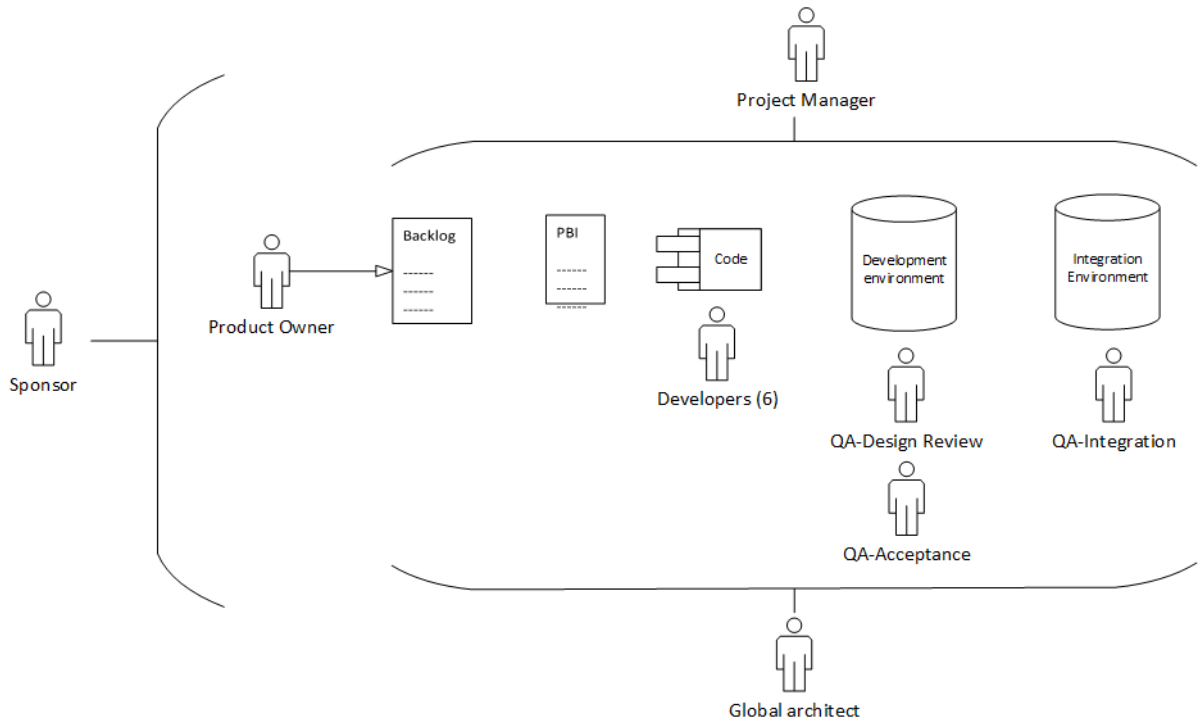


Figure 2.1 - Représentation des rôles de l'équipe. (Cédric Melançon, Wiki du projet)

2.6 Livrables

Tel que mentionné précédemment, lors de la rencontre de lancement du projet, les différents composants du projet furent énumérés et brièvement expliqués. Les sections suivantes présentent les différents composants du projet.

2.6.1 Simulateur

Le simulateur est le premier sous-système à concevoir et est défini comme une application de bureau, qui sera développée en C++, servant à simuler un objet connecté. Le simulateur est lié à un *RemoteThing* dans la plateforme ThingWorx. Le simulateur doit être configurable afin de permettre la simulation de toute sorte d'objets connectés. Il doit donc être possible de

configurer ses entrées et sorties en termes de type de données ainsi que la fréquence d'envoi de celles-ci. L'injection des données doit pouvoir s'effectuer de manière manuelle, aléatoire ou par la lecture d'un fichier de données. Le simulateur devra également pouvoir être lancé dans plusieurs instances concurrentes afin de simuler un volume plus important de trafic provenant de plusieurs objets connectés.

2.6.2 Réalité augmentée

Le module de réalité augmentée est le deuxième sous-système à concevoir. Il offrira aux utilisateurs la possibilité de visualiser l'état actuel de l'objet (le thing) connecté en scannant un marqueur apposé sur celui-ci à l'aide d'un appareil mobile (c.-à-d. un téléphone intelligent, une tablette, etc.), doté d'une caméra et d'une connexion internet. Il permettra d'afficher les données relatives à l'état de l'objet en temps réel permettant la superposition de l'image sur l'écran de l'appareil.

2.6.3 Apprentissage Prédicatif

Le troisième sous-système à concevoir vise à utiliser le module *Analytics* de ThingWorx pour développer un modèle de maintenance prédictive de la durée de vie utile d'un appareil à l'aide d'un ensemble de données décrites dans Saxena et coll. [1]. Une extension à la plateforme devra alors être conçue afin d'effectuer l'agrégation des données des capteurs simulés et produire certaines statistiques. Ce sous-système devra également être en mesure de générer un modèle de prédiction qui permettra d'analyser le flux de données d'entrée avec l'objectif de prédire la durée de vie utile.

Une seconde extension à la plateforme ThingWorx viserait d'automatiser l'entraînement et le déploiement de modèles prédictifs en production. Ainsi, il serait possible de lancer les entraînements et déploiements de façon automatique dans le but d'obtenir une solution clé en main et intégrée.

2.6.4 Modélisation dans ThingWorx

Afin de connecter le simulateur, l'application de réalité augmentée ainsi que les modèles d'apprentissage prédictifs, il sera essentiel d'avoir un point commun pour l'échange d'information. La plateforme ThingWorx de PTC devra servir d'intersection. Pour ce faire, il devient incontournable de modéliser une représentation virtuelle de l'objet connecté.

2.6.5 Interfaces et Alertes

La plateforme ThingWorx offre la possibilité de créer ses propres interfaces. C'est donc cette technologie qui a été choisie afin de démontrer les pleines capacités du produit. L'ajout de vues afin d'afficher les données historiques et de constater l'état actuel des objets connectés à l'aide de tableaux de bord sera nécessaire. ThingWorx permet aussi la configuration d'alertes en réaction à divers événements, par exemple lorsqu'une propriété atteint un certain seuil.

2.6.6 Rapport de synthèse (présent rapport)

En plus des contributions faites au Wiki du projet, et ce à même le répertoire Git hébergé dans l'entreprise Matricis, l'équipe de projet devra également produire un rapport de synthèse. Ce rapport compilera une synthèse de ce qui est effectué dans le projet ainsi que les informations nécessaires pour la pérennité du projet. Le contenu de ce rapport comprendra une description de la problématique technique et son contexte d'affaires, les avancés du projet, les acquis techniques et théoriques, la conception de la solution ainsi que les recommandations pour Matricis en vue de projets futurs.

2.6.7 Présentations de projet

Afin de répondre aux exigences départementales de l'École de Technologie Supérieure concernant les projets de fin d'études, l'équipe devra présenter le fruit de leurs efforts en fin de semestre afin de partager les connaissances acquises lors du projet. Cela dit, une seconde présentation devra également avoir lieu dans les bureaux du promoteur afin que l'entreprise puisse bénéficier de l'expérience de l'équipe de projet dans leurs expérimentations et de leurs acquis techniques. Cette présentation servira également de clôture au projet en s'assurant de bien effectuer le transfert de connaissances.

2.7 Risques

Certains risques furent identifiés en début de projet. D'autres, par contre, furent révélés à mesure que le mandat se clarifie au cours de l'avancement du projet. Le tableau suivant liste les principaux risques pris en compte tout au long de la réalisation.

ID	Description	Prob.	Cons.	Atténuation / Mitigation
R01	Manque d'expérience de l'équipe avec les technologies concernées	Probable	Sévère	Prévoir du temps de formation sur les différentes technologies dans l'attribution des tâches.
R02	Indisponibilité matérielle ou logicielle	Possible	Irrécupérable	Prévoir des plans B non bloquants pour les activités nécessitant un accès à du matériel spécifique.
R03	Sous-estimation de l'effort et de la complexité	Probable	Sévère	Établir un niveau de risque pour chacune des activités afin de prévoir une marge pour les activités risquées.
R04	Limite de temps	Probable	Sévère	Prioriser les activités de développement afin d'atteindre un produit minimum viable (MVP).
R05	Limite de disponibilités	Possible	Sévère	Organiser la gestion du temps et allouer au minimum une journée de travail commune par semaine.
R06	Compatibilité des solutions et des outils	Possible	Modérée	Abandon des tâches de compatibilité si manque de temps.
R07	Manque d'information pour procéder dans une tâche	Probable	Sévère	Prioriser les questions fonctionnelles lors des réunions de mêlées.
R08	Manque de communication entre les membres de l'équipe	Possible	Modérée	Établir une méthode de communication globale à l'équipe dès le début du projet.
R09	Manque de connaissance sur les cas d'utilisation du produit	Probable	Modérée	Prévoir des ressources pouvant répondre aux questions de l'équipe de développement.
R10	Demandes de changement	Probable	Modérée	Rogner certaines tâches ou prioriser les changements, dépendamment du contexte.

Tableau 2.2 : Risques à la réalisation du projet.

Ce chapitre a couvert les différents éléments contextuels du projet. Matricis, une compagnie de solutions d'intégration de données dans le milieu industriel, désire créer un prototype d'engin IoT connecté à l'aide de la plateforme ThingWorx à des fins de démonstration. Pour ce faire, deux équipes d'étudiants travailleront respectivement à l'implémentation logicielle et matérielle de la solution. Le prochain chapitre présentera la méthodologie de travail et les technologies utilisées pour la conception de la solution logicielle.

CHAPITRE 3 - MÉTHODOLOGIE DE TRAVAIL

Cette section du rapport présente les éléments de gestion et de suivi du projet. Les différentes méthodologies et répartitions des tâches adoptées lors de la réunion de lancement y sont décrites. De plus, une vue d'ensemble des technologies utilisés lors de la conception et de la réalisation du prototype expérimental sont également présentées.

3.1 Gestion de projet

La méthodologie de gestion de projet fut adoptée lors de la rencontre de lancement. Étant donné l'ampleur du projet, les contraintes de temps et le manque de ressources, une méthodologie Kanban fut adoptée pour la gestion de projet. Ainsi, une période dédiée à une rencontre d'avancement hebdomadaire fut déterminée à l'aide de l'outil Doodle². Les réunions de mêlées avaient donc lieu les jeudis midi, et une journée de travail fut décidée le mercredi, étant donné la disponibilité commune de la majorité des étudiants. Une salle de travail était réservée aux étudiants pour leur permettre d'avancer les différents modules tout en ayant accès aux ressources de l'entreprise en cas de questionnements sur les attentes ou sur les technologies utilisées.

3.3 Suivi de projet

Le suivi des différentes activités du projet s'effectuait au cours des réunions de mêlées hebdomadaires. Lors de ces rencontres, l'ensemble des membres de l'équipe étaient présents et donnaient l'avancement sur leurs tâches respectives. Ainsi, pendant cette rencontre, l'équipe procédait également à la révision de la progression du tableau de Kanban et aux attributions de tâches, autant au niveau des développeurs que du promoteur. Ces réunions offraient également à l'équipe l'opportunité de décrire les problématiques rencontrées ou même de valider certaines exigences du produit. Les exigences au niveau du promoteur étaient majoritairement axées sur la disponibilité du matériel et des licences logicielles.

² Doodle est un outil de planification permettant de partager un sondage sur les plages de disponibilités de chacun des membres afin de déterminer les dates et heures de rencontres. Pour plus de détails : <https://doodle.com/fr/>

3.4 Outils et technologies utilisées

Les divers modules à concevoir et implémenter font partie d'un écosystème complexe et hétérogène dans lesquels plusieurs technologies seront utilisées. Les sections suivantes décriront plus en détail les dépendances externes au projet, leurs rôles, ainsi que leur fonctionnement global.

3.4.1 ThingWorx

La plateforme ThingWorx est un produit développé par la compagnie américaine PTC permettant le développement rapide de solutions tout-inclus en internet des objets [2]. La plateforme consiste en une application Web pouvant être hébergée sur un service infonuagique ou encore sur un serveur dédié à l'intérieur de la compagnie. La plateforme permet de modéliser un ensemble d'éléments connectés par des principes d'héritages, similairement à de la conception orientée objet. Également, la plateforme permet de concevoir différents services (ou fonctions) en réponse à des événements ou à des appels provenant de l'extérieur par le biais de son API publique intégrée. Le logiciel permet également de gérer les accès à certains composants, objets ou services par une gestion basée sur des droits de lecture, de modification et d'exécution pour des utilisateurs et des groupes.

Un autre élément clé de la plateforme ThingWorx est son extensibilité. Il est en effet possible d'ajouter des fonctionnalités à la plateforme par le biais d'extensions développées en Java ou en JavaScript. Ces extensions peuvent faire appel à différents services existants et ainsi créer un réseau d'automations en réponse à des changements des valeurs de propriétés des objets. Un serveur d'analyse avancé peut également être ajouté à la plateforme et permet l'importation de jeux de données et la création de modèles prédictifs utilisant des techniques d'apprentissage machine. Les modèles entraînés peuvent ensuite être utilisés avec les objets modélisés afin de récolter et analyser leurs propriétés pour prédire une valeur de sortie.

Au niveau de la connectivité, ThingWorx permet à des objets connectés de communiquer avec la plateforme par l'entremise d'une API ReST ou d'un Websocket utilisant un protocole propriétaire, disponible par leur kit de développement pour objet connecté (*Edge SDK*).

3.4.2 Unity + Vuforia

Unity est une technologie de développement 3D utilisée largement dans le domaine architectural et le domaine du jeu vidéo. Cette technologie offre la possibilité de créer des applications et des scènes avec des rendus en 3D ou en 2D et de les déployer sur de multiples plateformes (c.-à-d. Windows, OSX, iOS, Android, etc.). Son moteur comprend également un engin de script permettant d'animer et d'ajouter de la logique aux éléments de la scène par des scripts en C# ou en JavaScript.

Vuforia est un logiciel de réalité augmentée développé par PTC. Il permet entre autres la reconnaissance de marqueurs permettant d'identifier certains éléments à des fins d'utilisations dans des applications de réalité augmentée. Dans le cadre de ce projet, l'équipe a utilisé un plugiciel de Vuforia développé pour l'engin Unity, qui permet d'effectuer une requête HTTP vers le serveur de Vuforia afin de collecter les métadonnées du marqueur détecté. Ce plugiciel englobait des fonctionnalités de requête, de détection d'images et les structures de données nécessaires à ces fonctions.

3.4.3 ThingWorx Edge C SDK

Le « kit de développement (c.-à-d. CDK) » est produit par la compagnie PTC. Il permet de développer des applications pour des objets connectés pouvant communiquer avec la plateforme ThingWorx à l'aide du protocole binaire propriétaire *AlwaysOn* (voir la documentation du Edge SDK). Ce SDK comprend plusieurs définitions d'éléments et de fonctions représentant les entités et services compris dans la plateforme ThingWorx, en addition aux fonctions et configurations requises pour établir la communication avec le serveur. Un « wrapper » C++ englobant certaines fonctions du SDK a été fourni à l'équipe par Matricis, permettant ainsi de faciliter la connexion à ThingWorx.

3.3.4 Boost

Boost est un riche ensemble de bibliothèques gratuites et portables pour le langage C++. Cet ensemble est vu par plusieurs comme un remplacement potentiel à la bibliothèque standard C++ [2]. D'ailleurs, sa licence de distribution encourage son utilisation autant pour développer des logiciels libres de droits que pour des logiciels commerciaux. On y retrouve plusieurs bibliothèques traitant notamment de calcul matriciel, de calcul parallèle, d'expressions régulières et de lecture et écriture de fichiers [3].

3.3.5 Qt

Qt est un cadre qui facilite la création d'interfaces utilisateurs et le développement d'applications C++ multiplateformes. On y retrouve de multiples bibliothèques ainsi qu'un ensemble d'outils de conception adaptés. Qt est maintenu par la *Qt Company* et est offert sous la forme d'une licence gratuite *open source* ou commerciale selon les besoins de l'utilisateur.

3.3.6 CMake

L'utilitaire CMake consiste en une collection de logiciels libres de sources et multiplateformes servant à compiler et à construire une solution logicielle native à l'aide de fichiers de configuration. Il permet, entre autres, d'automatiser le processus de construction indépendamment de la plateforme ou du compilateur utilisé.

3.3.7 Plugiciel ThingWorx pour Éclipse et le SDK d'extension

Le plugiciel ThingWorx pour l'environnement de développement Eclipse est un composant à installer sur le logiciel afin de débloquer certaines fonctionnalités qui facilitent la création d'entités ThingWorx. Le plugiciel s'utilise conjointement avec un SDK spécifique à l'extension qui permet d'inclure les nouvelles fonctionnalités au langage Java. Ainsi, il est possible de créer une structure d'entités ThingWorx (Things, Shapes, Templates) d'une façon plus propice au contrôle de version. La création de ces nouvelles classes est facilitée par l'extension qui crée les structures et annotations nécessaires lors de la création d'entités. Les entités créées peuvent ensuite être importées sur le serveur en exportant la structure sous un format compressé zip.

Ce chapitre a présenté les éléments de méthodologie de travail ainsi que les technologies externes utilisés par l'équipe afin d'appuyer le développement du prototype expérimental. Une méthode allégée de gestion de projet fut adoptée afin de focaliser le temps de chacun des membres au développement et à l'évolution de la solution logicielle, celle-ci étant dépendante de plusieurs outils externes et nécessitant plusieurs types d'expertises en parallèle.

CHAPITRE 4 - CONCEPTION

Ce chapitre porte principalement sur l'architecture logicielle globale du système et la conception propre à chacun des sous-systèmes/modules de la solution proposée. L'architecture globale sera présentée à l'aide d'une vue à haut niveau décrivant chaque élément de la solution. Ainsi, la conception de chaque composant du système est décrite et expliquée en profondeur dans le but de décrire l'utilisation des technologies et leurs interactions avec les interfaces externes.

4.1 Architecture globale

Les divers modules à concevoir et implémenter, lors de ce projet, nécessitent de communiquer entre eux à l'aide de la technologie Websocket ainsi qu'à partir de l'API ReST fournie par l'application ThingWorx. Ainsi tel que présenté à la figure 4.1, le serveur ThingWorx représente le point central de l'échange de ces données, et sert de modèle au système complet pour sa représentation virtuelle des objets modélisés. D'autre part, les modules d'interfaces (*UI/Alerts*) ainsi que ceux de réalité augmentée (*Augmented Reality*) offrent plutôt des vues différentes sur l'état et les attributs des objets connectés. Au niveau du simulateur, celui-ci représente plutôt une émulation d'un objet réel doté de senseurs desquels les données seraient collectées pour être envoyées à ThingWorx. Le module d'analyse prédictif était déjà inclus/disponible dans la partie *analytique* de la plateforme ThingWorks. Il permet d'effectuer de la prédiction sur les données fournies à ThingWorx par le simulateur. Son but sera d'estimer la durée de vie utile restante de l'appareil analysé par les différentes vues.

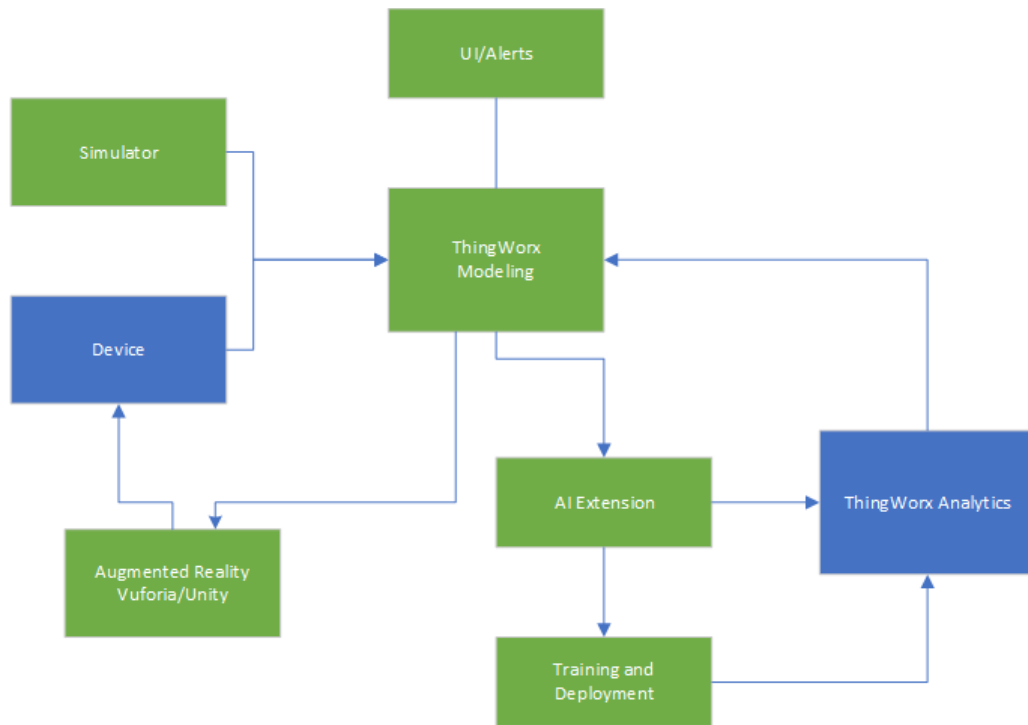


Figure 4.1 - Architecture à haut niveau de la solution. (Cédric Melançon, Wiki du projet)

Les sections suivantes élaboreront davantage sur l'architecture et la conception de chacun des modules. Les interactions requises avec les interfaces externes des autres modules seront également détaillées.

4.2 Simulateur

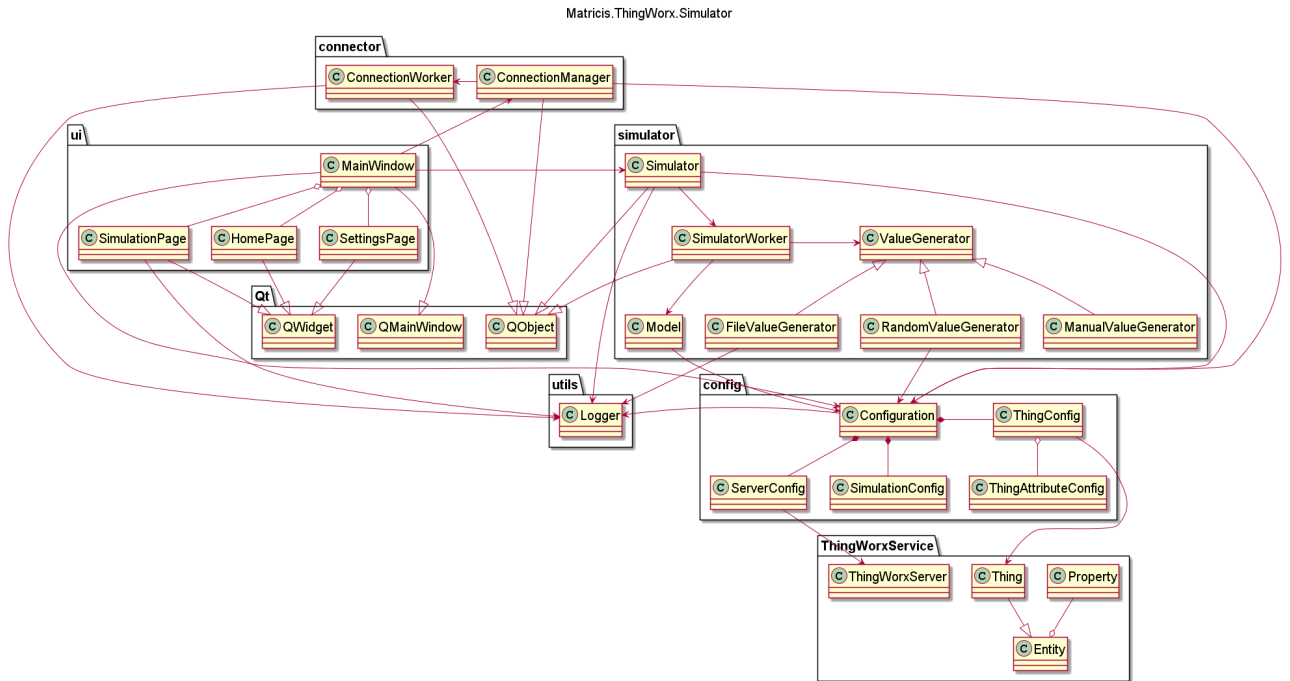


Figure 4.2 - Diagramme de classe du Simulateur

Tel qu'introduit à la section 2.6, le simulateur est une application de bureau qui sera développée en C++. Le projet sera structuré autour de l'utilitaire CMake pour faciliter la compilation sur les environnements de travail variés des différents contributeurs. Chaque dossier du code source contient un fichier CMakeLists.txt dans lequel on retrouve des instructions de compilation et de liaison pour les différents modules. Cette approche permet non seulement le développement multiplateforme, mais aussi la création de multiples sous-projets pouvant être compilés indépendamment les uns des autres. CMake facilite aussi la gestion des dépendances pour le projet. En effet, le Simulateur dépend de plusieurs bibliothèques et cadres pour son bon fonctionnement. Les principaux sont les suivants:

- Le cadre Qt qui est au cœur de l'application. Il est principalement utilisé pour ses fonctionnalités d'interface utilisateur et de multithreading;
- Le ThingWorx Edge C SDK et son wrapper C++ pour l'établissement de la connexion et la transmission des données générées;
- La collection de bibliothèques C++ Boost pour la lecture et l'écriture de fichiers.

Finalement, la figure 4.2 décrit que le code source du simulateur est séparé en cinq modules principaux, soit les modules config, ui, simulator, connector et utils.

4.2.1 Module config

Le module « config » englobe toutes les classes en lien avec la gestion de configuration et permet au simulateur de charger un fichier de configuration au format XML. La structure du fichier de configuration en question est séparée en trois sections. La première comporte les informations relatives au serveur, soit l'adresse IP, le port de connexion, la clé d'application ainsi que le nom du projet tel qu'il est défini sur la plateforme ThingWorx. La seconde section contient les informations relatives à l'objet qui doit être simulé (le Thing). On y retrouve le nom de l'objet (ThingName), le nom de son modèle (ThingTemplate) et sa liste de propriétés. Chaque noeud de propriété contient son nom, sa description textuelle, ses limites minimale et maximale en état d'opération normale ainsi que le pourcentage de dépassement au-delà des limites en état d'opération anormal. Finalement, la dernière section est dédiée aux informations relatives à la simulation. On y retrouve notamment la fréquence en millisecondes à laquelle le simulateur doit générer et envoyer des valeurs à ThingWorx. Au moment où l'utilisateur sélectionne le fichier qu'il désire utiliser, la configuration, telle qu'elle est définie dans le XML, est chargée en mémoire puis transférée dans une structure de donnée prévue à cet effet. Les fonctionnalités de lecture et d'écriture du fichier de configuration sont encapsulées dans la classe « Configuration » qui utilise le module « property_tree » de la librairie Boost pour facilement itérer à travers les noeuds XML et en extraire les valeurs. La classe « Configuration » est composée des conteneurs de données « ServerConfig, ThingConfig et SimulationConfig » qui sont une représentation un pour une des sections du fichier de configuration.

Le simulateur utilise un seul objet « Configuration » à travers toute l'application. Cet objet est instancié et initialisé au démarrage du programme et une référence à celui-ci est passée en paramètre à toutes les classes qui en ont besoin. Ceci fait en sorte que tous les changements apportés à la configuration ont un effet global dans l'application. Les composants sont alertés des changements par l'entremise du système de signaux de Qt.

4.2.2 Module ui

L'interface graphique du simulateur est une simple fenêtre munie de trois onglets permettant d'accéder aux différentes vues de l'application. Le module ui contient l'ensemble des vues de l'application. On y retrouve les classes « MainWindow, HomePage, SettingsPage et SimulationPage » qui implémentent les fonctionnalités pour leurs parties respectives de l'interface utilisateur.

Page Home

L'onglet de la vue principale affiche ce qui se passe actuellement avec le simulateur. Il affiche les données historiques de la propriété sélectionnée. La vue se met à jour en temps réel et affiche les données des cinq dernières minutes. Lorsqu'aucun fichier de configuration n'est chargé, l'écran affiche un bouton de création et d'ouverture de fichier.

Page Settings

L'onglet de la vue de configuration est intimement relié au fichier de configuration. Un *tab control* est visible par section du fichier de configuration. Cette vue contient trois onglets principaux:

Section Server

Le sous-onglet Serveur sert à conserver l'information nécessaire pour se connecter au « Thing » de ThingWorx. Sur cette vue, l'utilisateur devra remplir les différentes informations requises pour se connecter au simulateur. En effet, trois informations seront requises: l'adresse du serveur, l'« Application Key » ainsi que le nom du projet.

Section Thing

Le sous-onglet « Thing » sert à définir les propriétés qui doivent être simulées. Un « combobox » est disponible avec la liste de tous les « Things » liés au projet inscrit dans la section Server. Le « Thing » sélectionné est celui qui sera simulé. Les attributs représentent ce qui doit être simulé avec leurs paramètres. La valeur représente la valeur actuelle dans ThingWorx et ne peut être modifiée. Une case à cocher est présente pour faciliter le choix des attributs à simuler ainsi qu'une corbeille pour choisir les attributs à retirer du fichier de configuration.

Section Simulation

Le sous-onglet Simulation permet à l'utilisateur d'avoir le contrôle sur la fréquence en millisecondes à laquelle le simulateur doit générer et envoyer des valeurs à ThingWorx.

Page Simulation

L'onglet de la vue simulation sert à conserver l'information nécessaire pour l'exécution de la simulation. Celle-ci présente trois options à l'utilisateur:

- Le mode manuel qui permet de changer manuellement les valeurs envoyées à ThingWorx;

- Le mode « random » qui permet de générer de façon aléatoire des valeurs à envoyer à ThingWorx;
- Le mode file qui permet de rejouer une certaine séquence de données contenues dans un fichier CSV.

Les deux derniers modes seront expliqués plus en détail dans la section « Générateurs de données » qui se trouve plus bas. Il est important de mentionner que chaque mode a directement une influence sur l’affichage de l’historique de données en temps réel dans la vue principale de l’application.

4.2.3 Module simulator

Ce module contient les classes responsables de générer des valeurs pour les propriétés sélectionnées lorsque la simulation est lancée.

Pour commencer, la classe « Simulator » agit en tant que gestionnaire pour la simulation. Comme pour la classe « Configuration », il n’y a qu’une seule instance de l’objet « Simulator » dans toute l’application. En effet, la classe « Simulator » instancie un nouveau fil d’exécution (QThread) dans lequel elle place une instance de la classe « SimulatorWorker ». Ce fil d’exécution dédié à la génération de données peut être déclenché ou arrêté à tout moment par le « Simulator » qui joue le rôle d’intermédiaire avec le fil d’exécution principal. Pour ce qui est de la classe « SimulatorWorker », celle-ci est dans son propre fil d’exécution. Cette dernière utilise un compteur interne pour appeler la fonction de génération de valeur périodiquement à la fréquence configurée.

La classe « ValueGenerator » est une classe abstraite qui est héritée par les trois différents types de générateurs. Le « SimulationWorker » utilise son interface générique pour générer des nouvelles valeurs. Avec le concept d’héritage, plusieurs classes bénéficient de la classe « ValueGenerator ». Par exemple, la classe « ManualValueGenerator » peut générer ses valeurs à partir des valeurs des *sliders* dans l’interface, la classe « RandomValueGenerator » peut générer des valeurs aléatoires à partir des valeurs min/max définies dans la configuration, ou encore la classe « FileValueGenerator » peut générer des valeurs à partir d’un fichier CSV.

Finalement, « Model » est une simple structure de données qui répertorie l’historique des 100 dernières valeurs pour chaque propriété simulée. Elle est mise à jour par le « SimulationWorker » et elle est utilisée par le graphique sur la page principale de l’interface.

Tel que mentionné précédemment, à la section qui décrit la conception de l'interface utilisateur du simulateur, il existe trois modes différents pour choisir le type de simulation que l'utilisateur souhaite utiliser, soit les modes « Manual, Random et File ». Pour le mode aléatoire (c.-à-d. « Random »), il permet à l'usager de générer de façon aléatoire des valeurs à envoyer à ThingWorx. Les valeurs sont générées à l'intérieur des valeurs minimums (Min.) et maximums (Max.) définies. Lorsque l'option « anomaly » est activée, les valeurs sortent de cette zone en faisant des dépassements soit au niveau Min. ou Max. Pour ce qui est du mode File, celui-ci permet de rejouer une certaine séquence de données contenue dans un fichier CSV. Il existe aussi un mode supplémentaire à « File », le mode continu. Ce dernier permet de recommencer du début lorsque la fin de fichier est atteinte. Un mode « Play/Pause/Stop » est disponible pour cette option. La pause arrête la simulation et permet de reprendre au même endroit alors que le stop remet au début du fichier.

4.2.4 Module connector

Le module « connector » regroupe les classes « ConnectionManager et ConnectionWorker » qui ont la responsabilité de maintenir une connexion à ThingWorx et de transmettre les données générées. Plus spécifiquement, une instance unique de « ConnectionManager » offre les fonctionnalités de connexion et de déconnexion au serveur de ThingWorx en utilisant les valeurs configurées pour établir la connexion. Pour ne pas bloquer le fil d'exécution principal, une stratégie de thread similaires à celle du module simulator est utilisée. Le « ConnectionManager » possède une instance de « ConnectionWorker » qui s'exécute dans un fil d'exécution dédié à la communication. Lorsque la simulation est en cours, les données générées par le module de simulation sont envoyées au gestionnaire de connexion par l'entremise du système de signaux de Qt puis elles sont relayées au « ConnectionWorker » pour l'envoi au serveur. Le « ConnectionWorker » fait usage du SDK de ThingWorx et de son wrapper C++ pour initialiser (ou fermer) le canal de communication par Websocket et transmettre les données qu'il reçoit du « ConnectionManager ».

4.2.5 Module utils

Le module « utils » englobe principalement tout ce qui concerne la gestion des journaux d'évènement. En effet, ce module contient uniquement le « logger » qui utilise la librairie Log de Boost. Le « logger » est déclaré globalement et initialisé pour produire des journaux d'évènement qui respectent un certain format prédéfini. Il peut donc facilement être inclus et

utilisé par les autres modules de l'application. Le « logger » peut aussi être utilisé à partir de différents fils d'exécution.

4.3 Réalité augmentée

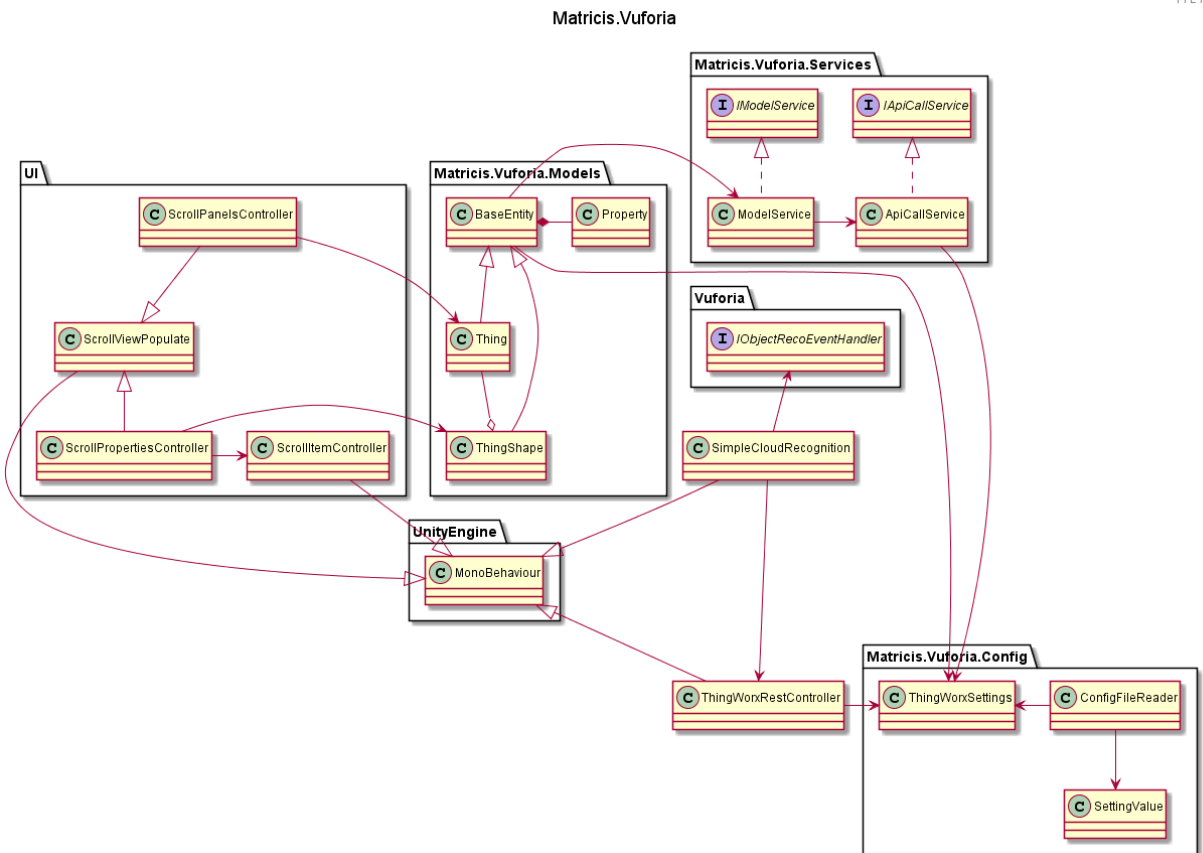


Figure 4.3 - Diagramme de classe de l'application de réalité augmentée.

4.3.1 Modèle

Le modèle de domaine utilisé dans cette application est basé en partie sur un projet existant chez Matricis servant afin d'intégrer ThingWorx plus efficacement dans les systèmes d'information existant. Cela dit, une partie des « namespaces Matricis.Vuforia.Models et Matricis.Vuforia.Services » ont été empruntés de ce projet et ont permis de modéliser plus efficacement les structures de données utilisées. On observe par exemple la classe « BaseEntity » qui représente l'objet de base pouvant être créé sur ThingWorx. Dérivé de «

BaseEntity », les classes « Thing et ThingShape » représentent respectivement l'objet connecté ainsi que les « Shapes » lui étant attribuées. Une modification effectuée au code existant consiste à modéliser le « Thing » comme une composition de « ThingShapes », permettant ainsi de définir la provenance des attributs de l'objet.

4.3.2 Services

Au niveau du package comprenant les services, on retrouve plutôt les classes utilitaires servant à créer et exécuter les requêtes à l'API publique de l'application ThingWorx et à gérer les paramètres de connexion. Les différentes classes provenant de « BaseEntity » connaissent les méthodes appropriées afin d'aller chercher leurs informations sur le serveur de façon autonome. On peut donc demander à un « Thing » ou un « ThingShape » de se rafraîchir. Le cas échéant, celui-ci fera appel à la classe « ModelService » qui appellera « ApiCallService ».

4.3.3 Configurations

Le package de configuration comprend quant à lui une classe partagée par plusieurs entités de l'application. « ThingWorxSettings » contient les informations de configuration pour le délai de rafraîchissement des données, la clé d'application ainsi que les informations de connexion au serveur. Cette classe est configurée par le « ConfigFileReader » lors du lancement de l'application en lisant un fichier de configuration contenant les informations. Un dictionnaire d'entités « SettingValues » est ensuite utilisé pour construire une instance de « ThingWorxSettings ». La figure 4.3.1 représente la séquence d'opération et l'interaction entre les classes concernées par la configuration de l'application à son lancement. Tout comme les autres éléments héritant de la classe « MonoBehaviour », les classes « ThingWorxRestController et SimpleCloudRecognition » peuvent être attachées à des « GameObjects » dans l'éditeur Unity et implémentent les fonctions « *update()* » ainsi que « *start()* ». Cette dernière est automatiquement lancée pour chacun des scripts liés à un élément GameObject. Ainsi, la configuration de l'application fut effectuée dans cette fonction.

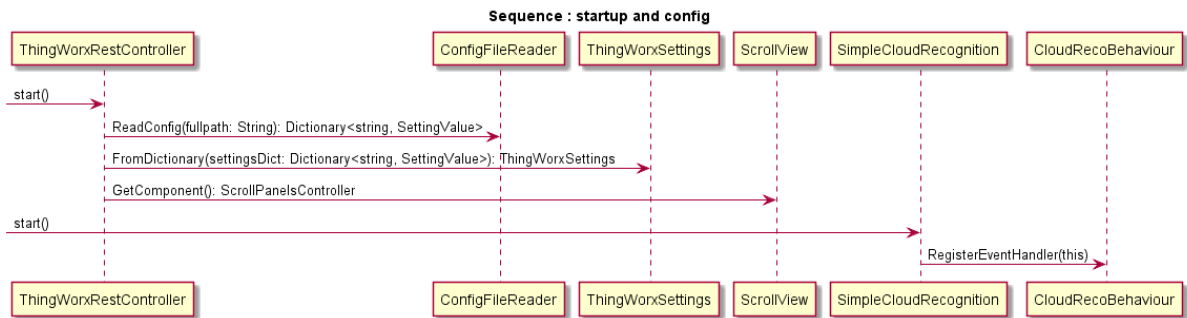


Figure 4.3.1 - Diagramme de séquence de la configuration de l'application

4.3.4 Logique de rafraîchissement des données

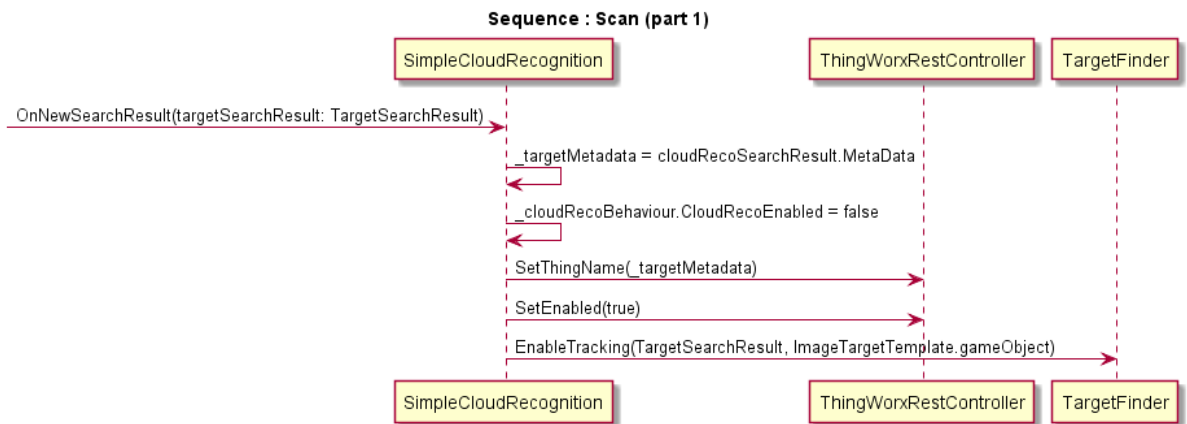


Figure 4.3.2 - Diagramme de séquence de la détection d'une cible (partie 1)

Une fois l'application configurée, la recherche de cible débute et l'utilisateur est amené à scanner une cible donnée afin de démarrer l'affichage des données. Vuforia gère les éléments de la reconnaissance d'image. Il fallait donc implémenter le comportement au retour de l'information du serveur. Une fois une cible détectée, l'application reçoit les métadonnées associées à l'image détectée. Celles-ci sont également préalablement stockées sur le serveur de Vuforia et sont liées à l'image importée sur ce même serveur. Ainsi, dans le cas présent, les métadonnées reçues consistent en le nom du « Thing » duquel on désire observer les données. La chaîne de caractère représentant le nom du Thing est attribuée au « ThingWorxRestController », qui s'occupera de lancer des appels séquentiels au serveur afin d'aller chercher la structure de l'objet ciblé, puis rafraîchir ses données à une fréquence dictée par le fichier de configuration.

La figure 4.3.4 démontre l'interaction qui découle d'un appel de la fonction « *update()* » sur le « ThingWorxRestController ». La fonction « *update()* » est appelée à chaque itération dans une boucle de jeu dans Unity après avoir calculé les éléments physiques de jeu lorsqu'applicable. Dans la méthode « *update()* » du « ThingWorxRestController », la séquence d'action doit d'abord vérifier que cette classe a été activée (voir partie 1 à la figure précédente). Si tel est le cas, la structure de l'objet (le « Thing » et sa composition de « Shapes ») doit être importée du serveur par une requête à l'API. Cette structure est ensuite manipulée afin d'être facilement accessible pour des fins d'affichage. Ainsi, les différentes « Shapes » sont transformées en une structure de dictionnaires comprenant comme clé le nom de la « ThingShape » et comme valeur l'objet « ThingShape » lui-même. Cet objet à son tour contient un dictionnaire ayant comme clé le nom des propriétés et comme valeur un objet « Property », comprenant le nom, la description, la valeur, et l'unité utilisé pour cette propriété. Cette structure vise à faciliter l'attribution des valeurs recueillies de l'API après un rafraîchissement.

Cette solution fut considérée, car l'API ne retourne pas les propriétés d'un objet de façon hiérarchique (par Shape). Par une requête vers l'URI : `/type/ThingName`, il retourne la liste des propriétés dans une liste aplatie en plus de retourner leur source, type, valeurs minimale et maximale, et autres métadonnées propres à ThingWorx. Une fois cette structure importée via la fonction « *ImportAsync()* », qui crée également la structure précédemment décrite, les données sont rafraîchies par la fonction « *RefreshAsync()* », qui s'occupe d'aller changer chacune des valeurs en cherchant dans la structure de dictionnaires afin de remplacer les bonnes valeurs de propriétés selon la valeur actuelle sur le serveur. Cette fonction utilise l'URI : `/type/ThingName/Properties` afin d'aller récupérer que la liste des attributs et leur valeur, sans leur provenance.

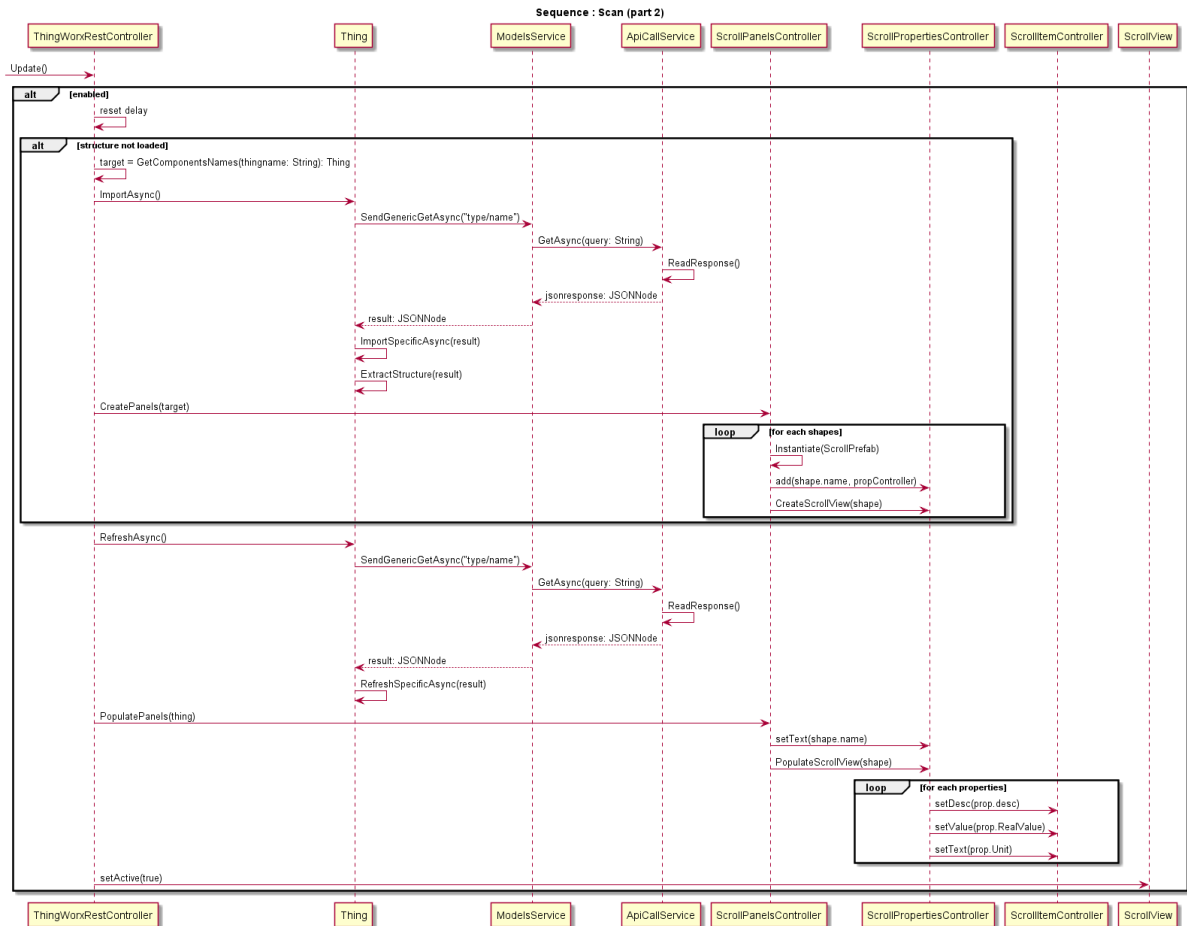


Figure 4.3.4 - Diagramme de séquence de la détection d'une cible (partie 2)

4.3.5 Logique d'affichage

Étant tous des ingénieurs logiciels, une des plus grandes faiblesses de l'équipe est la conception graphique d'interface utilisateur. Premièrement, nous ne sommes pas les ressources les plus doués en graphisme, donc il était clair dès le début que l'esthétique des interfaces utilisateurs du prototype n'allait pas être la priorité de l'équipe. Cependant, il était important pour nous que la convivialité des interfaces utilisateurs soit bonne. C'est pourquoi nous avons décidé de réaliser différents prototypes et d'expérimenter avec des techniques d'affichage d'interface utilisateur adaptées pour la réalité augmentée. Tout d'abord, nous voulions un interface utilisateur qui flotte dans l'espace augmenté de l'application au-dessus du code scanné, mais avons réalisé, après quelques tests, que cette technique n'était pas optimale pour cette application particulière puisque l'utilisateur devait garder l'appareil pointé dans la direction du code pour visualiser l'information, ce qui n'est vraiment pas

pratique lors de la démonstration de l'application. Il a été conclu qu'il était plus avantageux d'utiliser une interface qui serait épinglée dans le bas de l'écran, en tout temps, afin de constamment présenter l'information à l'utilisateur. La figure 4.3.5 présente le premier « *mock up* » de cet interface utilisateur ayant ces caractéristiques. Une seule modification a été apportée à cette première ébauche et par la suite l'équipe était satisfaite. L'interface finale implémentée sera décrite à la section [5.3 Réalité augmentée](#).

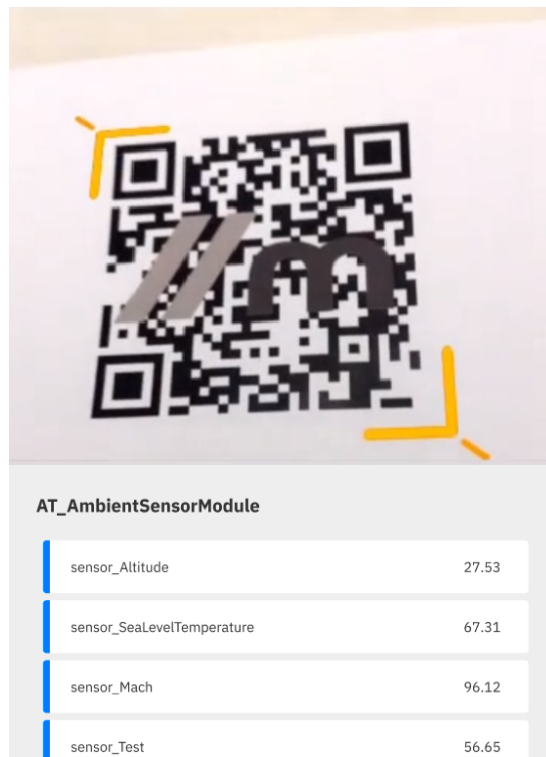


Figure 4.3.5 – Premier « *mock up* » avec un menu épinglé au bas de l'écran

4.4 Modélisation dans ThingWorx

La modélisation des diverses entités à l'aide de ThingWorx fut inspirée, en grande partie, des travaux publiés par Saxena et coll. [1]. Tel que décrit à la figure 4.4.1, un découpage avait été effectué préalablement pour décrire le turboréacteur présenté dans cet article. L'objet à modéliser devait donc être une composition de divers modules, ayant chacun leur lot de propriétés.

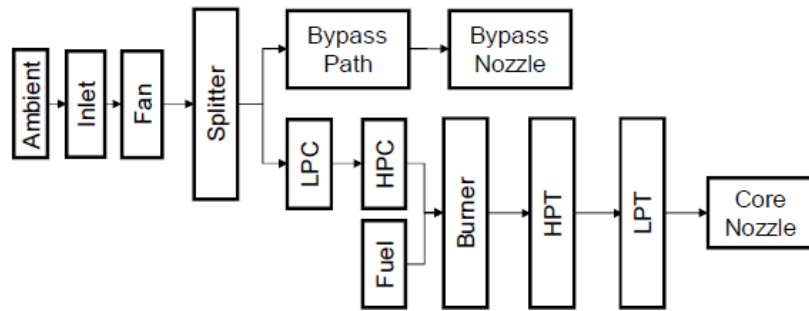


Figure 4.4.1 - Découpage de la turbine en composants [1].

Les primitives présentées dans le jeu de données de l'article de Khan et coll. [4] furent d'abord étudiées par l'équipe, puis divisées arbitrairement parmi les différents modules au meilleur des connaissances de l'équipe. La figure 4.4.2 démontre la sous-division des attributs afin de recréer le turboréacteur par composition d'objets.

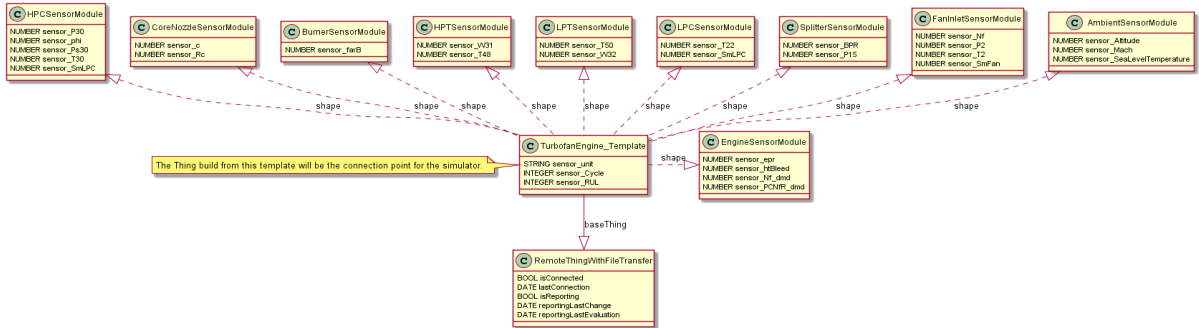


Figure 4.4.2 - Conception de l'engin pour l'intégration dans ThingWorx.

4.5 Interfaces et Alertes

Des maquettes à réaliser ont été fournies par Matricis. Ces maquettes ont été créées au début du projet lorsque l'objet à modéliser était une scie à ruban et n'étaient donc plus utilisables lorsque le projet a évolué. De nouveaux requis ont donc été établis pour les interfaces et leur implémentation a été laissée à l'équipe.

Ce chapitre a présenté une vue haut niveau de l'architecture globale du système et la conception de ses modules. Ainsi, un aperçu des interactions entre les éléments formant les modules a été illustré, permettant une meilleure compréhension de la logique des différents composants formant le système.

CHAPITRE 5 - INTÉGRATION ET RÉSULTATS

Ce chapitre porte principalement sur l'intégration et résultats du système. Une architecture de référence sera premièrement présentée montrant les différents composants de l'architecture réseau. Ainsi, pour le simulateur, l'application de réalité augmentée et les interfaces et alertes, différentes captures d'écran seront présentées et expliquées afin de faciliter la compréhension de chacun des systèmes.

5.1 Architecture de référence

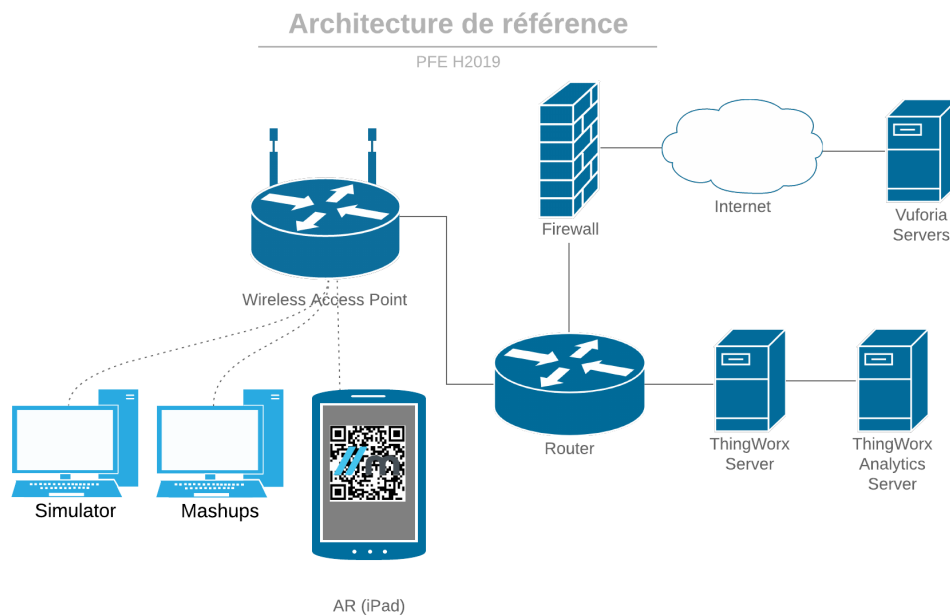


Figure 5.1- Architecture de référence des composants

La Figure 5.1 démontre le déploiement des différents composants en termes d'architecture réseau. Ainsi le serveur de Vuforia est disponible hors du réseau de Matricis. Le simulateur, l'ordinateur accédant aux interfaces ainsi que le iPad utilisant l'application de réalité augmentée doivent être connectés au réseau interne afin de pouvoir accéder à la plateforme ThingWorx, hébergée sur les serveurs de Matricis. Le serveur d'analyse avancé de ThingWorx est également déployé sur le réseau local et lié à l'application ThingWorx.

5.2 Simulateur

Pour ce qui est du simulateur, tel qu'expliqué plus tôt lors de la conception, ce dernier a été divisé en trois grandes vues : la vue principale de l'application (c.-à-d. « Home »), la vue de réglages (c.-à-d. « Settings ») et la vue simulation (c.-à-d. « Simulation »).

Home

Cette vue est la vue principale qui apparaît lorsque l'utilisateur lance l'application du simulateur. Deux boutons sont disposés permettant à l'utilisateur de télécharger un fichier de configuration XML déjà existant dans un répertoire de l'ordinateur ou encore de débiter la simulation à partir d'un fichier vierge.

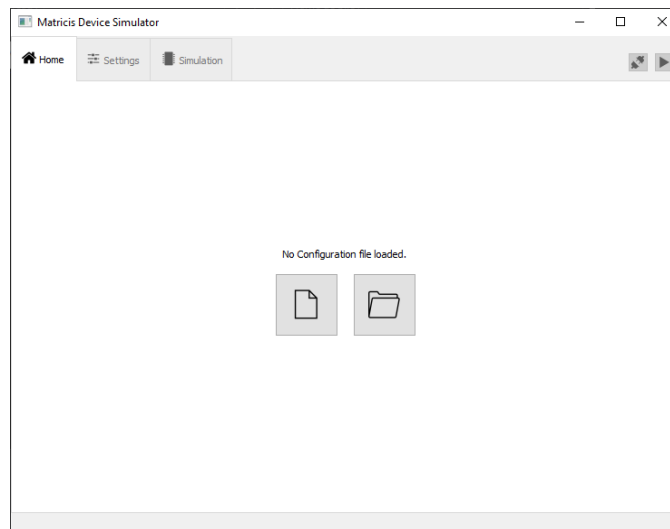


Figure 5.2.1- Vue principale du simulateur - « Home »

Plus tard, une vue sera présentée pour démontrer les données historiques de la propriété au moment où l'utilisateur initie la simulation.

Settings

Une fois que l'utilisateur a fait son choix, l'application présente la vue « Settings » qui permet à l'utilisateur de configurer la simulation à sa guise par l'entremise de trois onglets différents: « Server, Thing et Simulation ».

Le premier onglet, « Server », est une vue liée à la configuration du serveur sur lequel l'utilisateur souhaite se connecter. Ainsi, ce dernier sera demandé d'entre les informations précises du serveur. Idéalement, un bouton de test devrait être ajouté afin de valider la connexion au serveur avec les informations entrées.

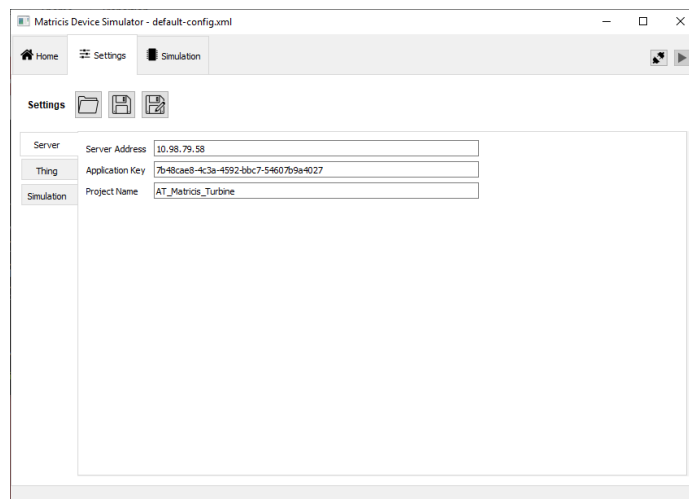


Figure 5.2.2- Vue de l'onglet « Server » du simulateur - Settings

La prochaine figure présente le second onglet, « Thing ». Cette vue permet à l'utilisateur de choisir le « Thing » à simuler. Dans le cas où la connexion a bien pu être établie, cette vue devrait pouvoir afficher un menu déroulant contenant tous les « Things » liés au projet. Pour chaque « Thing » choisi dans le menu, un tableau affichera des attributs et leurs propriétés importantes permettant aussi de choisir quel attribut sera simulé par l'entremise d'une case à cocher.

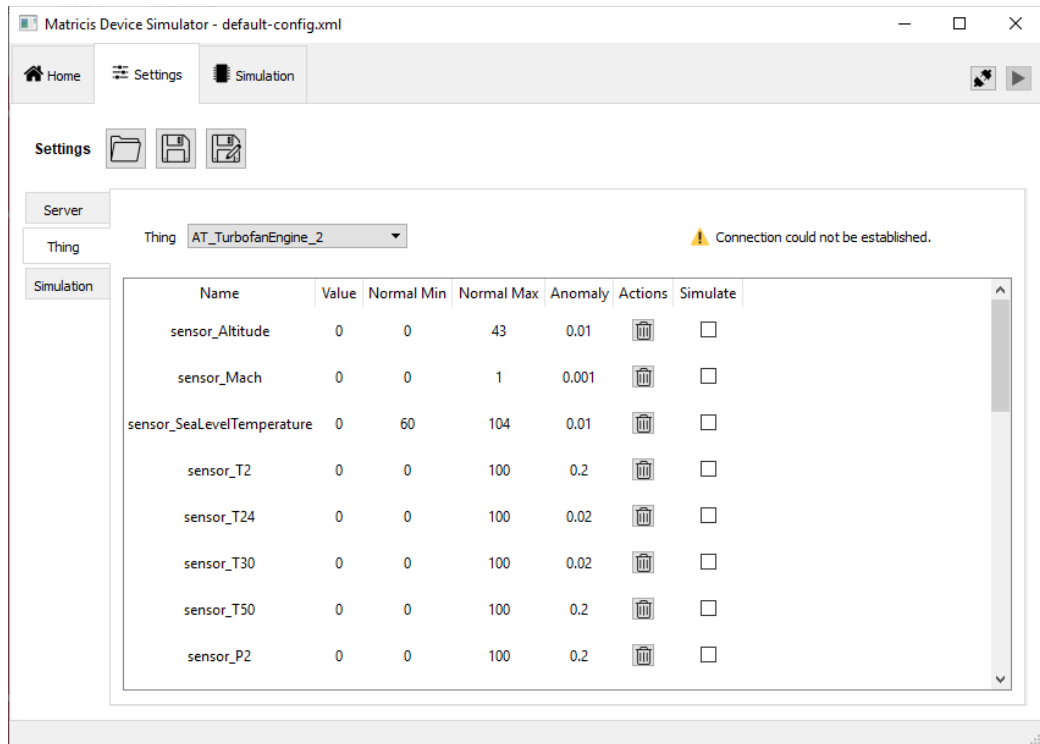


Figure 5.2.3- Vue de l'onglet « Thing » du simulateur - Settings

Enfin, le dernier onglet de la vue « Settings » se trouve à être la vue qui permet à l'utilisateur de contrôler la fréquence en millisecondes à laquelle le simulateur doit générer et envoyer des valeurs à ThingWorx.

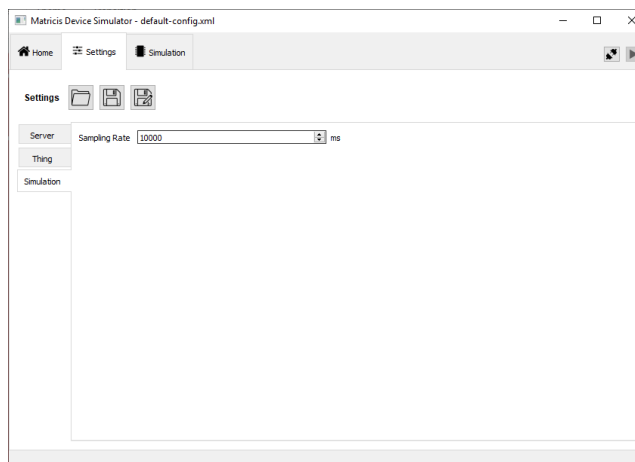


Figure 5.2.4- Vue de l'onglet Simulation du simulateur - Settings

Simulation

Une fois que l'utilisateur a choisi les attributs qu'il souhaite simuler, il est maintenant possible pour lui de débiter la simulation en cliquant sur l'icône « play » disposée en haut à droite de la vue Simulation. Tel que mentionné précédemment dans la section expliquant la conception du simulateur, cette vue présente à l'utilisateur trois façons de simuler les données: un mode manuel (« Manual »), aléatoire (« Random ») et par l'entremise d'un fichier CSV (« File »). L'influence de ces trois modes sur la simulation sera démontrée sur un graphique qui sera disponible tout au long de la simulation dans la vue principale.

Lorsque l'utilisateur choisit le mode manuel comme type de simulation, une interface avec plusieurs « sliders » se présente à lui. Ainsi, les différents curseurs sont facilement manipulables et permettent de changer la valeur des différents capteurs listés. La couleur verte représente l'intervalle de valeurs valides et la couleur rouge représente les valeurs qui se trouvent hors de la zone normale, donc à l'extérieur de la valeur minimale et maximale.

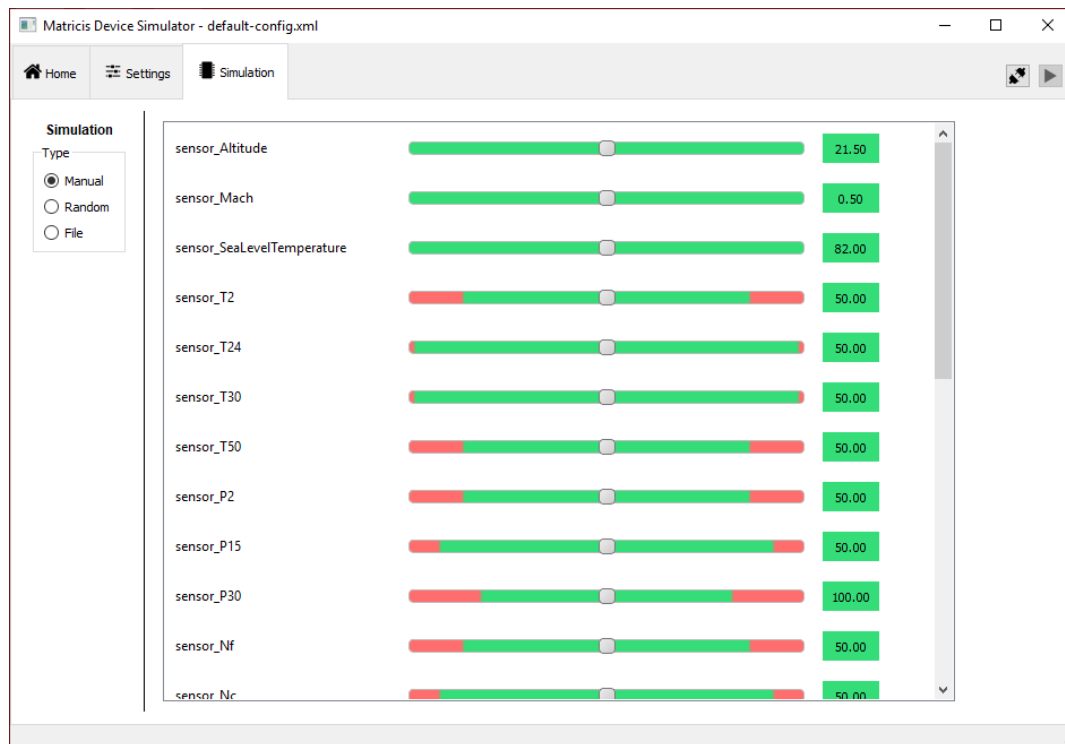


Figure 5.2.5- Vue Simulation du simulateur en mode manuel - « Simulation »

Lorsque l'utilisateur choisit le mode « random » comme type de simulation, l'interface suivante présente un tableau listant tous les attributs ayant la possibilité d'ajouter une

anomalie par l'entremise d'une boîte à cocher. Cette boîte permet de faire sortir les valeurs de la zone Min./Max. en faisant des dépassements.

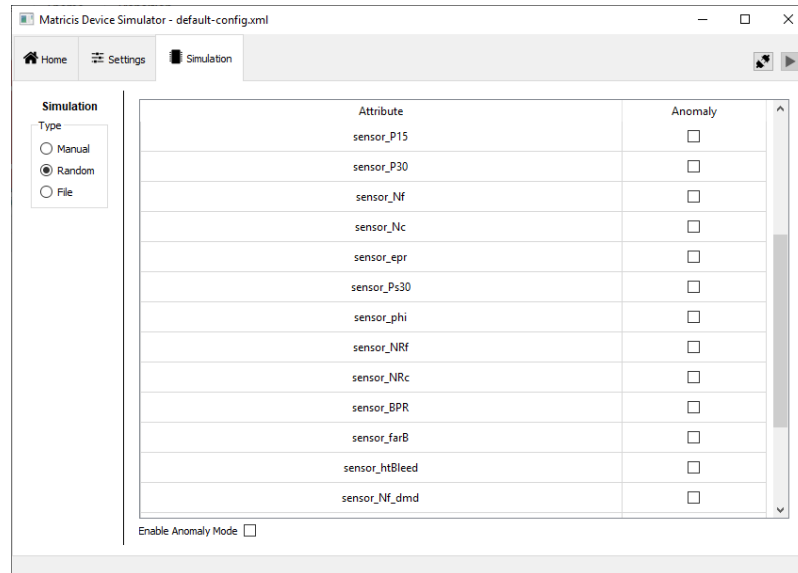


Figure 5.2.6- Vue Simulation du simulateur en mode « Random » - « Simulation »

Lorsque l'utilisateur choisit le mode file comme type de simulation, l'interface présente la possibilité à celui-ci de télécharger un fichier contenant une séquence de données.

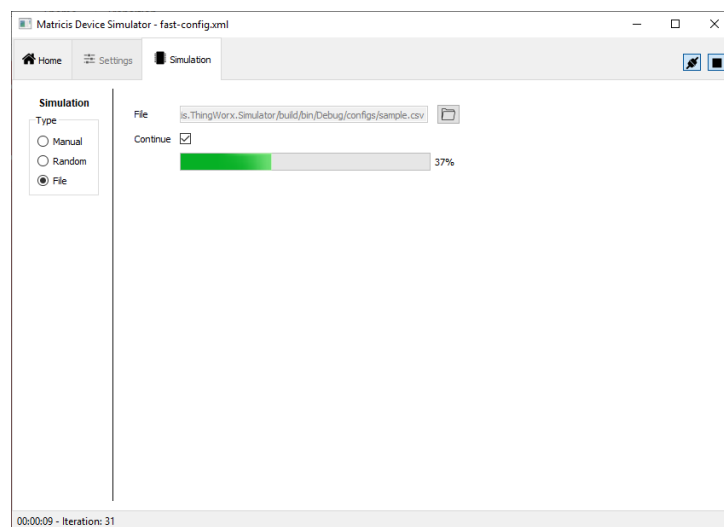


Figure 5.2.7- Vue Simulation du simulateur en mode « File » - « Simulation »

Home au moment de la simulation

Lorsque l'utilisateur aura activé la simulation, la vue principale change pour un graphique qui affiche ce qui se passe avec le simulateur en temps réel. En effet, il devient maintenant possible de voir les données historiques de la propriété, et ce, en fonction de tous les paramètres configurés. Il est aussi possible pour l'utilisateur de modifier ces paramètres, par exemple, sur les « sliders » de l'onglet Simulation sur la page Settings, et de voir l'influence de ceux-ci directement sur le graphique. Il est important de mentionner que la modification de paramètres pendant la simulation n'est possible que si l'utilisateur utilise le mode manuel comme type de simulation.

La vue suivante montre un exemple de graphique qui peut être obtenu lors d'une simulation avec le mode aléatoire:

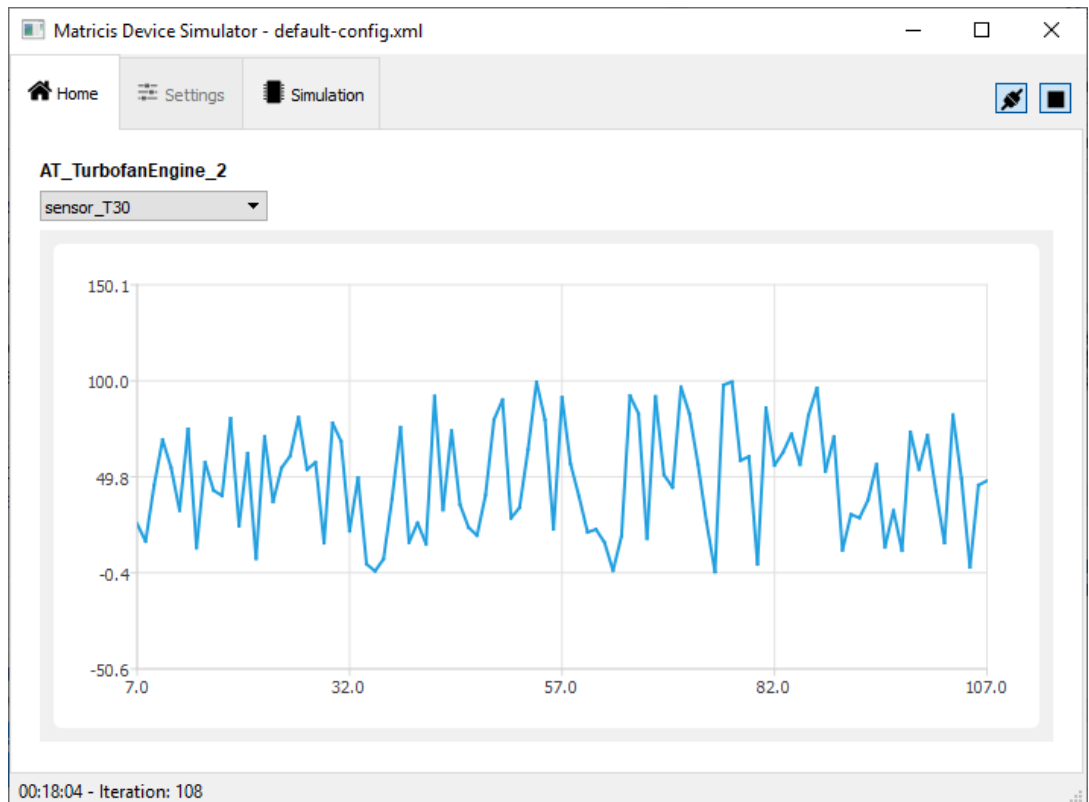


Figure 5.2.8- Vue du graphique avec le simulateur en mode aléatoire - Home

5.3 Réalité augmentée

Au niveau de l'application de réalité augmentée, les résultats sont assez brefs puisque l'application n'a qu'une seule page et les informations sont tous affichées de la même façon. Lorsque l'application est lancée, l'utilisateur se retrouve avec la caméra en plein écran et attend la détection d'un code. Lorsque l'utilisateur scanne un code, un menu qui prend 40% de l'écran est épinglé dans le bas. Ce menu affiche toutes les informations relatives au « Thing » associé au code scanné. L'utilisation de l'application est très simple, il suffit seulement de glisser son doigt de droite à gauche ou de gauche à droite pour faire défiler les différents composants.

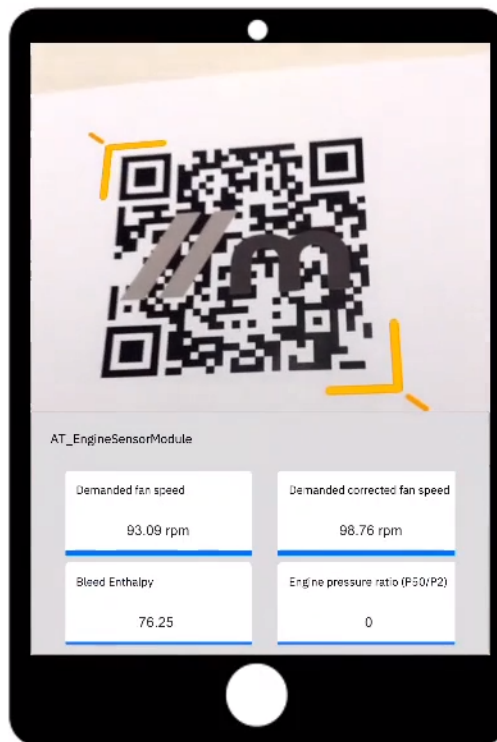


Figure 5.3- Vue de l'interface du module de réalité augmentée

5.4 Modélisation dans ThingWorx

Les résultats de modélisation pour ThingWorx se démontrent par une création d'entités sur la plateforme. Une série de « ThingShapes » fut créée représentant les différents composants du turboréacteur. Ensuite, un « ThingTemplate » implémentant ces « Shapes » et héritant de « RemoteThingWithFileTransfer », permettant ainsi la connexion par Websocket avec le Simulateur. La figure 5.4 montre les résultats d'héritage. Un « ValueStream » fut créé afin d'entreposer les données d'historiques pour l'affichage dans le « Mashup de Dashboard ». Également, un « Thing » héritant de « TimedThing » s'occupe de générer des données aléatoires à des fins de test sur un des objets créés à partir de « TurbofanEngine_Template ».

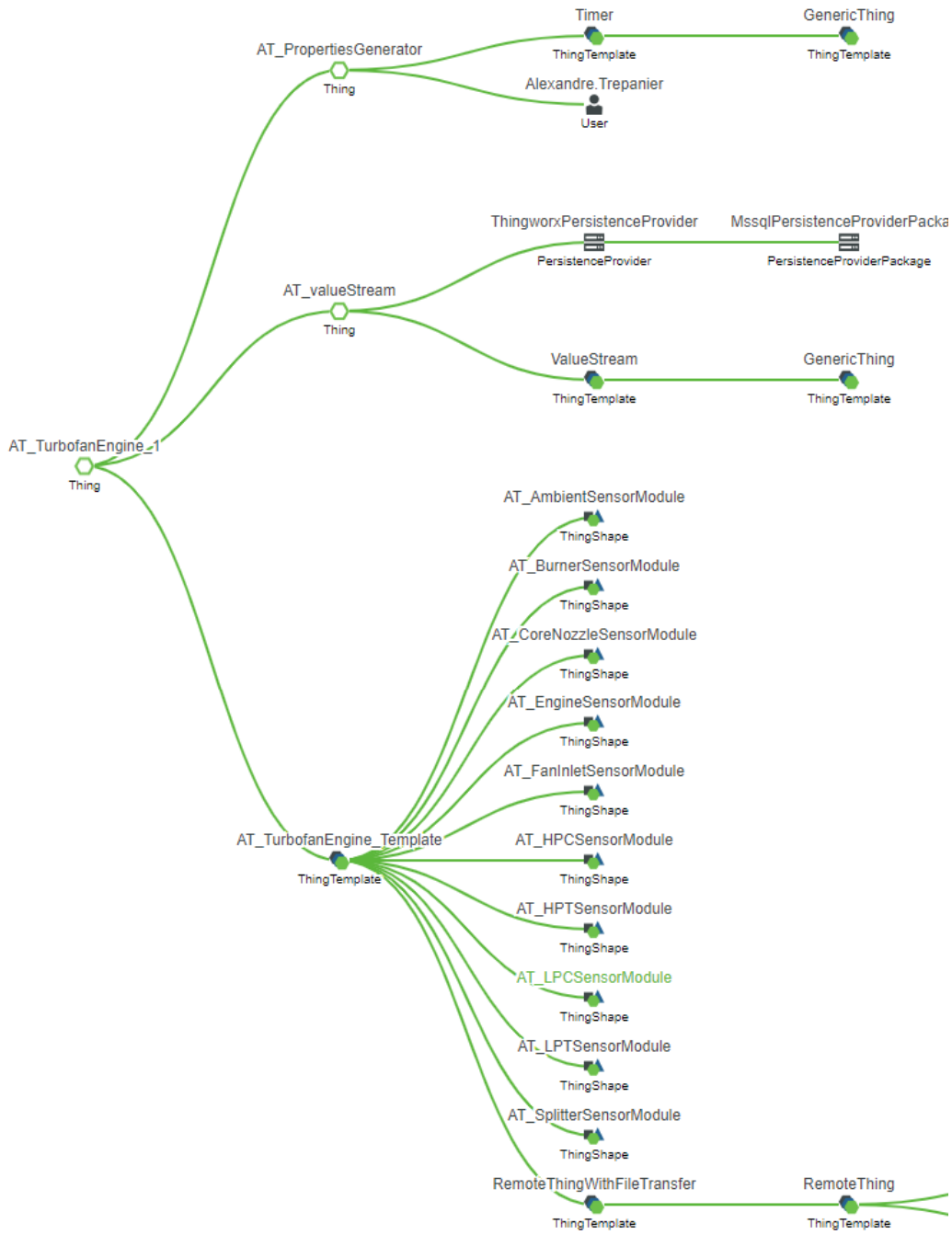


Figure 5.4 - Résultat de la structure sur ThingWorx

5.5 Interfaces et Alertes

La création d'interfaces utilisateur en ThingWorx se fait via des « mashups ». L'outil offre une technologie WYSIWYG, celui-ci rappelle le plugiciel WindowBuilder sur l'IDE Éclipse pour Java. Plusieurs composants sont offerts et peuvent être ajoutés à la page en les sélectionnant et glissant. Chaque élément offre des propriétés paramétrables via l'interface (minimum, maximum, largeur, etc.). Le bon fonctionnement des « mashups » se fait via les services et les liens entre ces services et les éléments ainsi que ceux faits entre les éléments. Il y a donc un menu pour aller récupérer les services offerts par ThingWorx et ceux créés par l'utilisateur et importer dans le « mashup » ceux nécessaires à son fonctionnement. Une fenêtre est présente afin de montrer une carte des liens formés avec l'élément sélectionné.

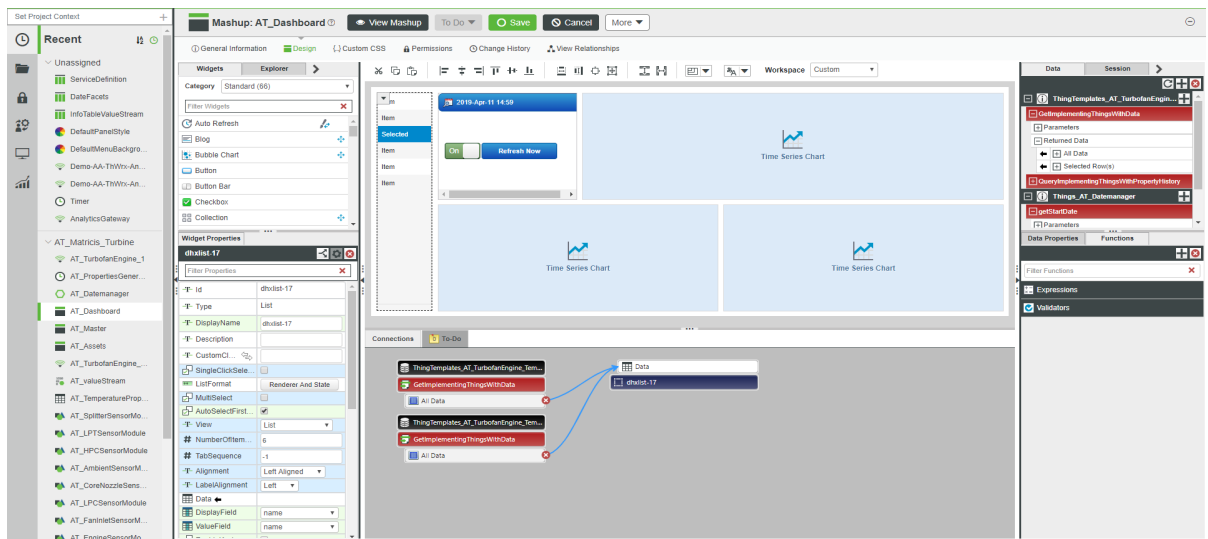


Figure 5.5 - Interface de l'outil de « mashups » de ThingWorx

Un « mashup » Master a été créé, celui-ci est une interface générale dont les différentes pages peuvent hériter. Cet héritage permet un visuel uniforme au travers de l'application, soit le logo de Matricis en haut de page et un menu de navigation à la gauche. Le reste de l'espace sert au contenu des différentes interfaces enfants.

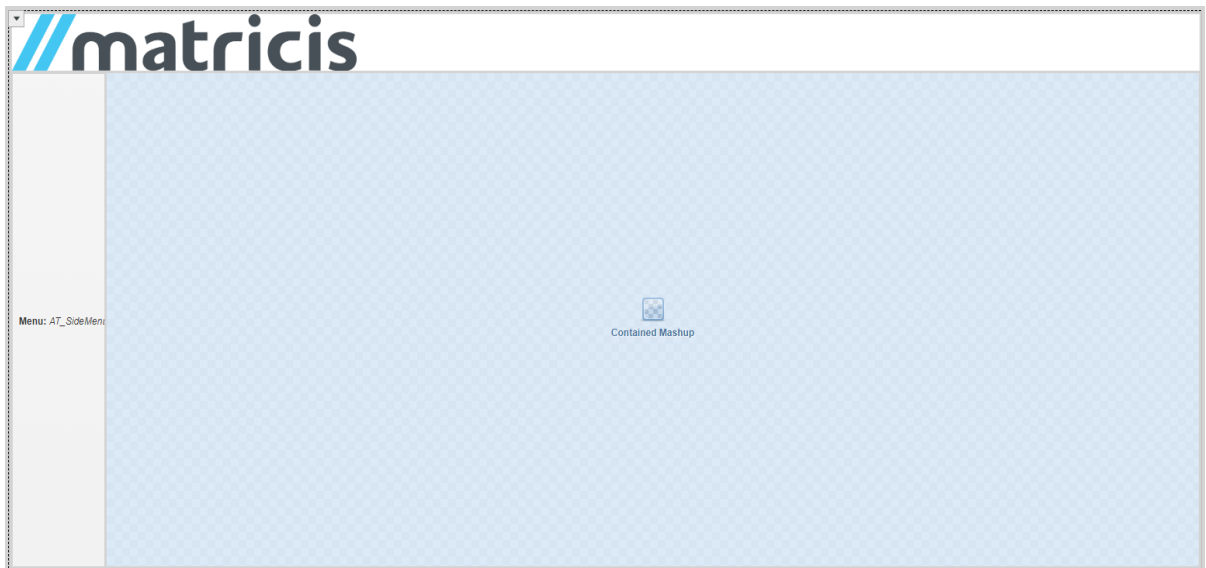


Figure 5.5.1 - « Mashup Master »

Le menu de navigation est lui aussi une interface à part qui est importée par le « Master ». Le menu est donc configuré avec les liens aux différentes pages sans la connaissance du « Master ».



Title	Link	Target	Icon
Dashboard	AT_Dashboard	Replace Page	 MonitoringStatusIcon
Assets	AT_Assets	Replace Page	 ThingsIcon

Figure 5.5.2 - Configuration du menu de navigation

La prochaine figure présente le tableau de bord qui a été réalisé. Comme on peut le voir, le « mashup Master » est hérité par cette page et le tableau de bord se retrouve dans l'espace libre présenté à la figure 5.5.1. À la droite de la barre de navigation se trouve une liste des différentes entités de turboréacteurs. L'élément sélectionné dans cette liste est celui dont les informations seront affichées dans les graphiques. Un composant de sélection de dates est au côté de la liste d'entités, la sélection est la date de départ des historiques affichés par les graphiques. Cette interface contient trois graphiques présentant les historiques de températures, de pressions et de vitesses des différentes composantes du moteur d'avion sélectionné. Finalement, invisible à l'œil, un bouton de rafraîchissement automatique est contenu dans cette interface. Il est important de noter que les graphiques dans la capture présentent des historiques aux valeurs simulées aléatoirement.

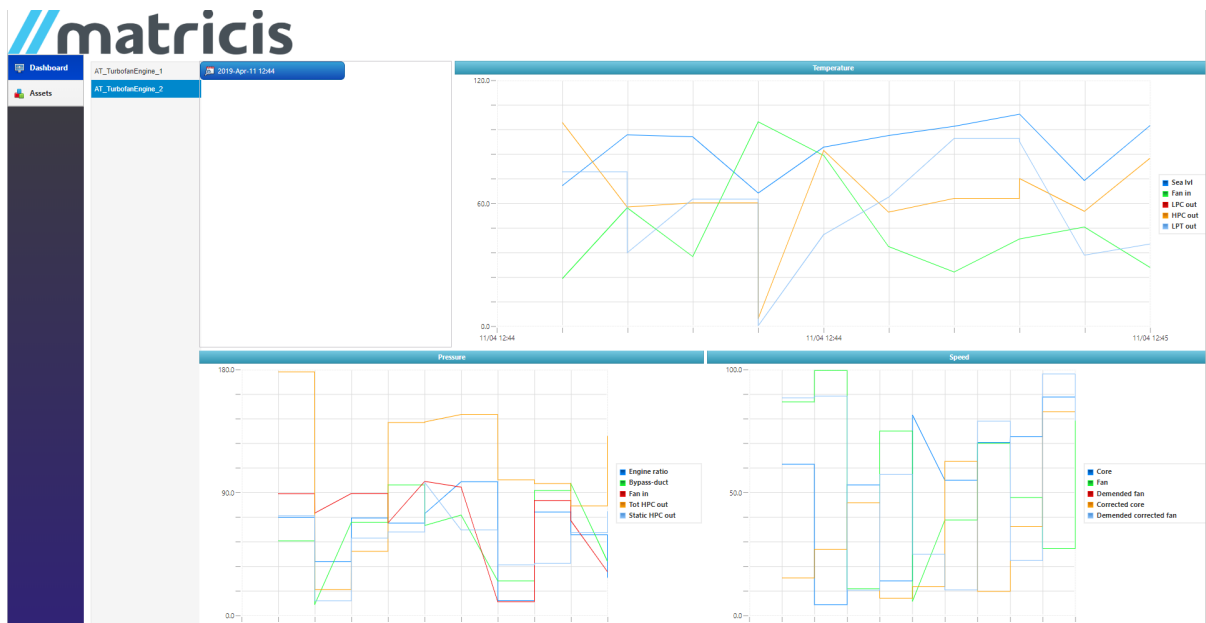


Figure 5.5.3 - « Mashup » tableau de bord

Tout comme l'interface précédente, le « mashup » implémente également le « Master ». Cette page contient elle aussi une liste d'entités sélectionnables et un composant de rafraîchissement automatique invisible. Contrairement au tableau de bord, ce mashup n'affiche que les valeurs actuelles du turboréacteur. La température au niveau de la mer, le Mach et l'altitude sont affichés à l'aide d'aiguilles et le reste des propriétés sont sous forme de tableau.

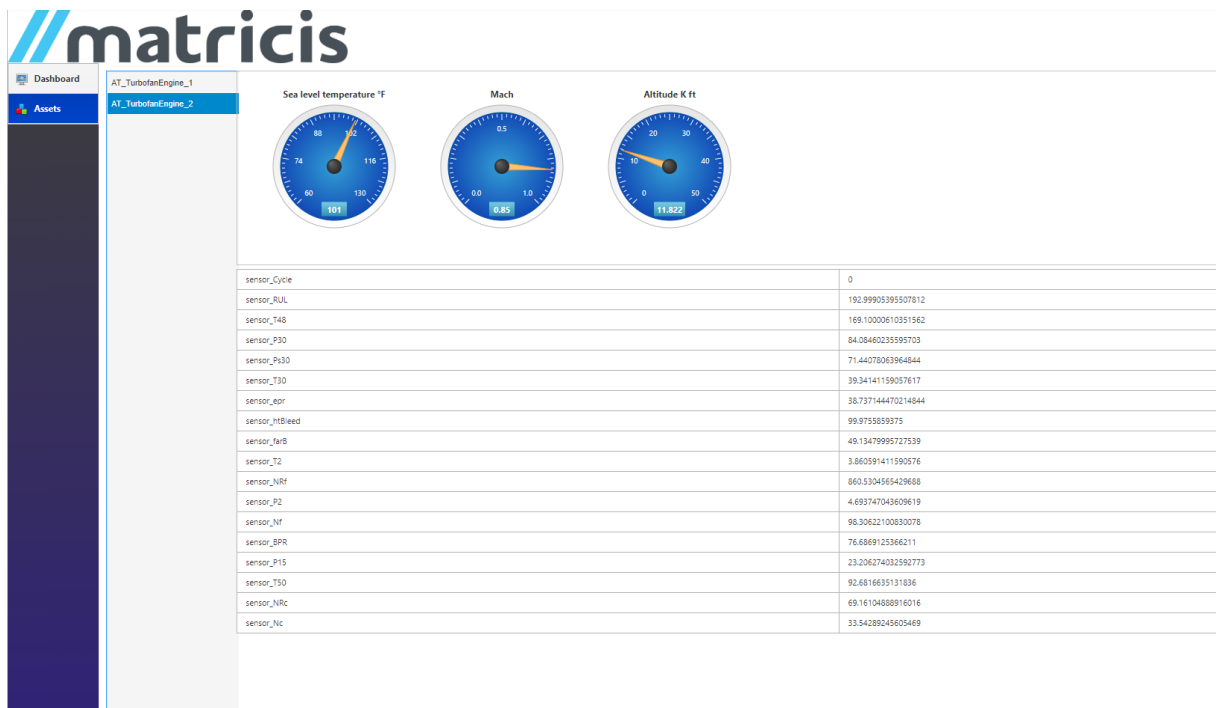


Figure 5.5.4 - « Mashup » assets

Tel que mentionné précédemment, l'utilisateur peut créer ses propres services pour les utiliser dans ses interfaces. ThingWorx offre la possibilité de créer ses fonctions JavaScript à même la plateforme. Ces services doivent absolument être contenus dans un « Thing ». Par exemple, un « Thing » de gestion de date a été créé et le service de la figure lui a été ajouté afin d'initialiser le sélecteur de date du tableau de bord à la date trois jours plus tôt.

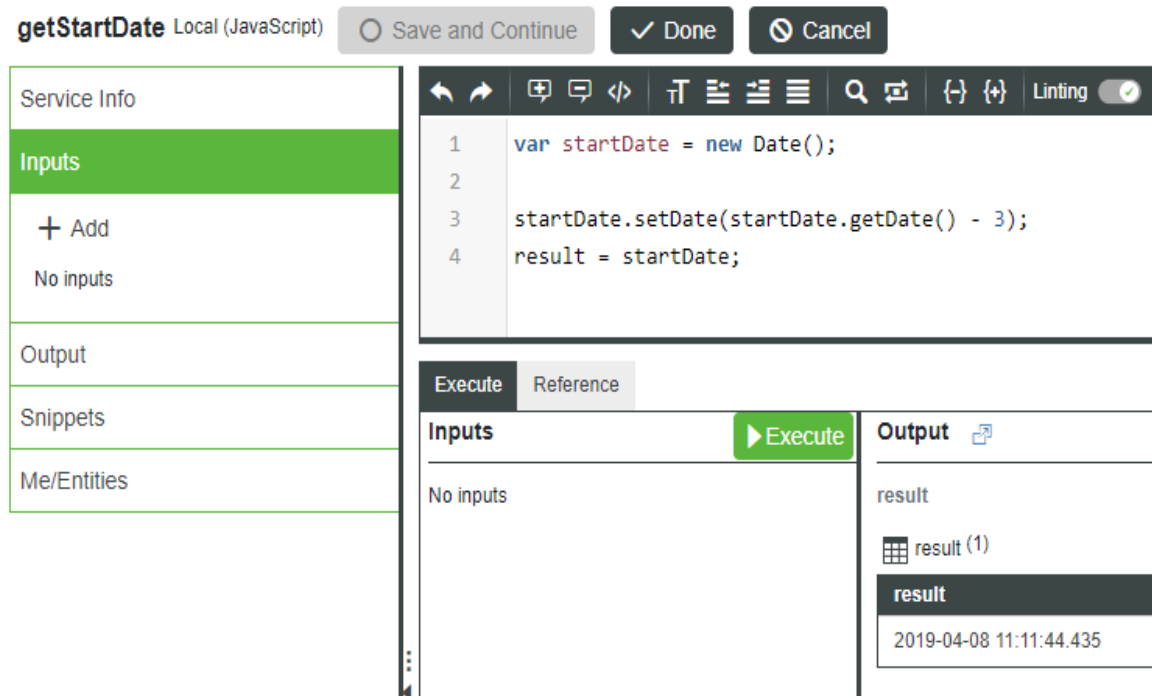


Figure 5.5.5 - Éditeur de texte pour services

Une fois le service créé, le développeur doit retourner sur sa page de mashup, aller dans le menu de services, retrouver le « Thing » créé et finalement sélectionner le ou les service(s) à importer.

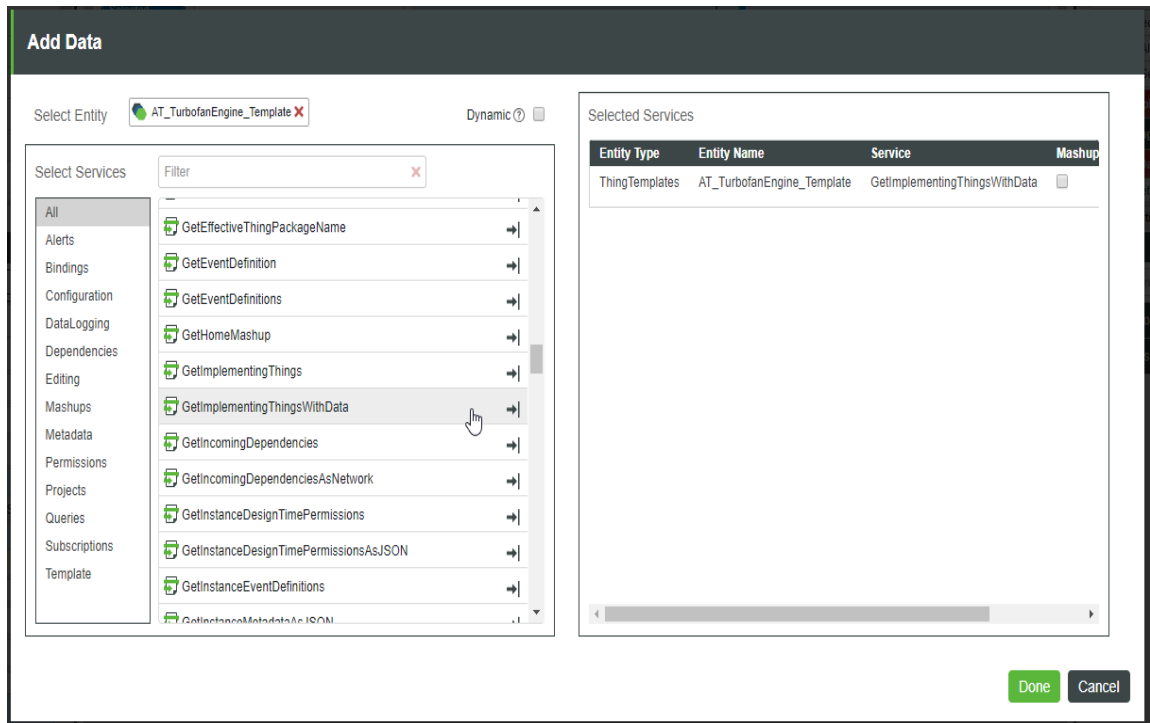


Figure 5.5.6- Menu des services pour « Mashups »

Une fois le service importé, il doit être ajouté à l'élément qui va l'utiliser, dans ce cas la valeur par défaut du sélectionneur de dates. C'est ainsi que les liens se forment. Dans la figure suivante, le service créé plus tôt est pris comme valeur d'entrée et la date sélectionnée est la valeur de sortie qui est envoyée au service qui récupère l'historique des entités.

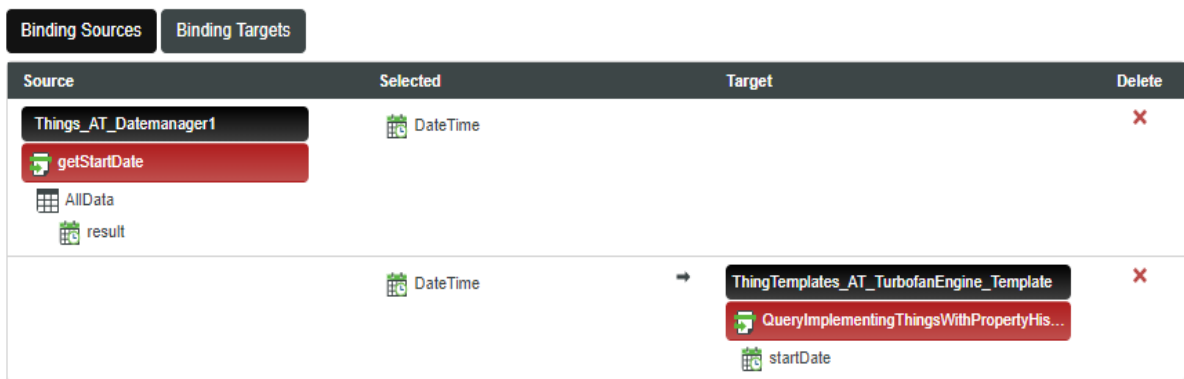
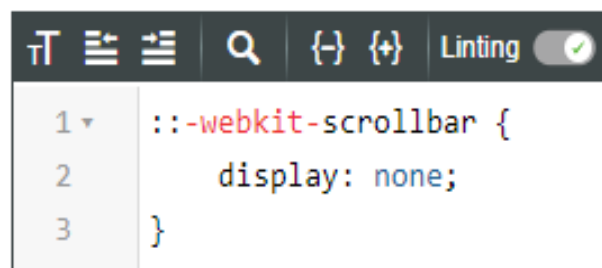


Figure 5.5.7 - Menu des liens d'un élément

Similairement aux services, ThingWorx offre un éditeur de texte à même la plateforme afin de créer ses propres styles. Dans l'exemple suivant, les éléments responsive de ThingWorx prennent toujours plus d'espace que nécessaire, le résultat est donc plusieurs barres de défilement non nécessaires. Le style personnalisé permet donc de retirer toutes les barres de défilement de la page.

Custom CSS



```
TT ≡ ≡ Q {-} {+} Linting   
1 ▾ ::-webkit-scrollbar {  
2     display: none;  
3 }
```

Figure 5.5.8 - Éditeur de texte pour styles

Cette solution a cependant un inconvénient, elle cachera aussi les barres de navigations qui sont utiles. Il est possible, à priori, d'ajouter des classes CSS personnalisées aux éléments des mashups. La solution initiale était donc de créer une classe no-scroll et de retirer les barres des composants avant cette classe. Cependant le lien entre l'éditeur de CSS et la propriété « CustomClass » des éléments ne s'est jamais fait pour des raisons qui n'ont jamais été trouvées. Il est aussi possible d'importer des fichiers CSS qui sont présents dans un certain répertoire du serveur hôte de ThingWorx. Cette fonctionnalité a été découverte vers la fin du projet et n'a donc pas été utilisée. De plus, n'ayant pas accès à l'hôte sur lequel l'application était hébergée, il aurait été difficile d'utiliser cette fonctionnalité.

Custom CSS

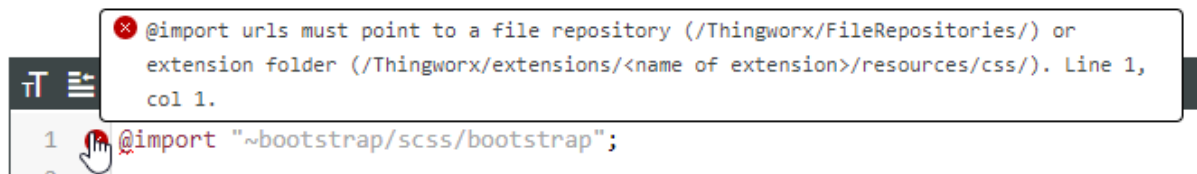


Figure 5.5.9 - Import de fichier CSS

Pour conclure, cet outil cible un grand public, incluant des gens ayant peu ou aucune expérience en programmation. Ce qui a pour résultat d'enlever du contrôle et de la flexibilité à un développeur plus expérimenté. Pour revenir à l'exemple de récupérer une date, c'est un système contre-intuitif qui ajoute plusieurs étapes à une action qui se devrait être simple. L'ajout de ces relations entre les services, les éléments et la plateforme elle-même complexifie la compréhension que peut avoir l'utilisateur du fonctionnement de son interface. À cela s'ajoute un environnement très peu verbeux et peu performant (c.-à-d. le curseur a de la difficulté à suivre la vitesse de frappe dans les éditeurs de texte). La recommandation de l'équipe est donc de se connecter à ThingWorx de l'externe et de s'en servir comme un API afin de pouvoir utiliser un cadriciel plus mature et plus personnalisable qui a déjà fait ses preuves.

CHAPITRE 6 - PROBLÈMES RENCONTRÉS

Ce chapitre présente les multiples problématiques rencontrées qui ont dû être surmontées et leurs impacts sur le projet. Bien certainement, le temps, la disponibilité des ressources humaines et matérielles, ainsi que le manque de connaissances de l'équipe vis-à-vis certaines technologies ont eu un impact considérable sur le démarrage du projet.

6.1 Manque de connaissance global sur ThingWorx

Tout d'abord, comme l'équipe de Pfe n'avait jamais travaillé avec ThingWorx, nous devions tous apprendre cette technologie d'IoT afin de premièrement modéliser le moteur et ensuite communiquer avec nos sous-systèmes. ThingWorx n'est pas un logiciel conçu pour être utilisé par des ingénieurs logiciels c'est pourquoi parfois certaines actions/fonctions requises qui étaient évidentes pour un ingénieur logiciel n'étaient pas disponibles dans ThingWorx. Par chance, la section « learning » du site de ThingWorx est très bien faite et les tutoriels sont assez complets, mais cela a quand même pris un certain temps avant que l'équipe soit confortable avec cette technologie. Il va sans dire que ce problème était inévitable et qu'il a fallu allouer du temps, lors du démarrage du projet, afin que nous nous familiarisions avec ThingWorx.

6.2 Licences ThingWorx et Serveur Analytics

L'équipe a aussi fait face à des problèmes administratifs plutôt que techniques. En effet, l'équipe a travaillé sur ThingWorx avec des comptes d'essai (c.-à-d. académiques) durant six semaines. Ce type de compte avait beaucoup de limitations par rapport aux vraies licences. L'équipe n'avait entre autres pas les droits pour installer de nouvelles extensions, téléverser des fichiers et utiliser le module d'entraînement de données. Une fois les licences obtenues des problèmes de connectivités au serveur « Analytics » sont survenues, laissant ainsi très peu de temps à l'équipe pour implémenter le module d'apprentissage prédictif.

6.3 Configuration et Compatibilité

Certains problèmes, en lien avec la compatibilité de la solution, ont aussi été rencontrés en cours de projet. Par exemple, le simulateur fut multiplateforme pendant la majeure partie du travail. Cependant, avec l'intégration du SDK, l'équipe a rencontré certains problèmes avec le script « CMAKE », qui aurait demandé de le reconstruire afin d'être compatible avec iOS.

Un autre problème de compatibilité survenu lorsque l'équipe a voulu déployer l'application d'AR sur le iPad. En effet, le développement sur iOS est plutôt difficile au niveau de la compatibilité requise. Le MacBook fourni par Matricis avait une version de MacOS qui ne permettait pas de télécharger la version de XCode disponible sur l'App Store. Cela dit, il fut possible de trouver une ancienne version de XCode sur le site d'Apple et ainsi de pouvoir construire la solution. Afin de supporter Vuforia, Unity requiert une version minimale de 11.0 pour iOS. La version maximale supportée par le Mac était 10.3 et le iPad opérait sur la version 12.1. Il fallut donc utiliser l'ordinateur personnel d'un des membres de l'équipe afin de déployer la solution correctement sur le iPad.

6.4 Contraintes de temps

Il s'agit bien évidemment d'un projet de fin d'études pour des finissants en génie logiciel et en technologie de l'information. Il est donc normal pour un projet comme celui-ci d'avoir une contrainte de temps. En effet, l'équipe de ce projet a eu à gérer le temps accordé entre l'avancement du projet et les autres travaux de session et examens à l'école. Cela a donc eu des répercussions sur le temps accordé au projet lorsque les étudiants étaient en mi-session et fin de session. Il est incontournable de mentionner que les embûches mentionnées dans ce chapitre ont eu un impact négatif sur le temps et l'effort possible. Avec plus de temps il aurait été possible de compléter la partie d'apprentissage prédictif du turboréacteur ou encore d'améliorer certaines parties du prototype.

6.5 Disponibilité des équipements

Comme la partie de réalité augmentée devait être exécutée sur un iPad et que la majorité de l'équipe possédait des équipements opérant sous le système d'exploitation *Windows*, un ordinateur utilisant Mac OS ainsi qu'un iPad devait être mis à notre disposition. Cependant, nous avons dû attendre longtemps après ces équipements et cela a retardé l'intégration et les essais de l'application de réalité augmentée.

CHAPITRE 7 - AMÉLIORATIONS ET TRAVAUX FUTURS

Ce chapitre présente les éléments qui n'ont pas pu être réalisés au cours de ce projet. Ainsi, cette section offre une piste pour les équipes de développements futures concernant des améliorations à apporter à ce premier prototype expérimental livré à Matricis.

Tel que mentionné précédemment, le temps et le manque de familiarité de l'équipe avec la technologie ThingWorks furent les deux enjeux majeurs de ce projet. En effet, plusieurs tâches prévues au début du projet n'ont pu être réalisées. Ce chapitre, discute des améliorations possibles au prototype livré pouvant ainsi servir de guide des prochains objectifs de projet pour les équipes de développement futures.

7.1 Apprentissage prédictif

Le plan initial, pour le module d'apprentissage prédictif, était d'importer les données provenant de Boost [2] dans le module d'analyse de la plateforme ThingWorx et d'agrèger les données à partir d'une extension développée en Java via le SDK et l'extension pour l'IDE Eclipse. Les agrégations prévues consistaient à faire une moyenne mobile sur les données ainsi que d'effectuer certaines statistiques de base sur l'ensemble des primitives provenant du jeu de données (c.-à-d. moyennes, écarts types, distributions, etc.). Une fois ces statistiques récupérées, il est possible avec la plateforme de créer un modèle en spécifiant la colonne correspondant à la valeur à estimer, ainsi que celle correspondant à l'identifiant de l'unité afin de distinguer les différentes entrées du jeu de données. Également prévue pour ce module était la réalisation d'une seconde extension à la plateforme qui permettrait d'automatiser l'apprentissage des modèles et leur déploiement en production afin d'être utilisés conjointement avec l'écosystème réalisé dans le cadre de ce projet pour effectuer de la maintenance prédictive. En pouvant prédire la durée de vie utile d'un appareil, l'information aurait pu être transférée aux différentes vues afin de pouvoir réagir correctement et ainsi apporter les correctifs et préventions nécessaires. Malheureusement, dû à des contraintes de temps, l'équipe fut dans l'impossibilité de réaliser le module d'apprentissage prédictif et de l'intégrer dans le système produit.

7.2 Améliorations au simulateur

Une amélioration proposée, en cours de développement, au simulateur consistait en l'ajout d'une requête qui allait importer la liste des propriétés d'un objet connecté depuis la plateforme ThingWorx et ainsi grandement faciliter la création d'un nouveau fichier de configuration. Pour ce faire, une requête vers l'API ReST devra s'effectuer vers le serveur ThingWorx afin de récupérer ces informations. Cependant, les connexions effectuées dans le simulateur à ce jour ne s'effectuent que par « Websocket » en poussant les données vers l'objet cible. Il serait donc nécessaire d'ajouter des fonctionnalités de récupération de données via une requête HTTP pour rendre possible cette amélioration.

Certains éléments dans le code source actuel seraient également à améliorer. Par exemple, la page de configuration actuelle ne comprend pas une entrée pour le port du serveur. Celui-ci a été encodé en dur à même la création de la configuration de serveur dans l'application. Également, certaines améliorations à l'interface furent également considérées. Les cases à cocher dans la page de configuration du « Thing » pourraient être plus grandes et il serait intéressant d'avoir une option pour sélectionner et désélectionner l'ensemble des attributs. Dans le fichier de configuration, il serait intéressant d'ajouter une valeur par défaut à chaque attribut afin de peupler les valeurs de la colonne Value dans la page de configuration du « Thing ». D'autres améliorations seraient également possibles au niveau du SDK de ThingWorx. En effet, il a fallu aller modifier légèrement le SDK afin d'utiliser un timeout nul pour que les appels d'API soient moins bloquants sur le fil d'exécution dédié à la connexion. Il fallut également désactiver le logger du SDK, car celui-ci était beaucoup trop verbeux et ralentissait l'exécution du logiciel.

Afin d'assurer le fonctionnement des modules après avoir effectué des changements sur le code, il serait nécessaire de créer une suite de tests unitaires pour l'application. Ceux-ci garantissent le fonctionnement des autres modules par régression après avoir effectué des changements à la structure ou au fonctionnement de certaines classes.

7.3 Réalité augmentée

Le plugiciel de Vuforia pour Unity permet de rendre des objets 3D sur l'écran d'appareils mobiles afin d'être superposés au marqueur détecté. Cela dit, il aurait été intéressant d'obtenir un fichier CAD représentant le turboréacteur pour ainsi l'afficher avec ses données regroupées par modules. Ainsi, les différentes propriétés auraient pu être juxtaposées aux

bons composants de l'appareil. Cela aurait été un meilleur cas d'usage pour l'application. Une autre amélioration possible aurait été un menu de configuration permettant de changer certains paramètres tels que l'URL du serveur et le taux de rafraîchissement. Également, il faudrait permettre de fermer l'écran affichant les données pour scanner un second marqueur. De plus, les valeurs de chaque composant pourraient avoir un certain code de couleur là où est affichée présentement une petite barre bleue en dessus de chaque boîte. Ce code de couleur pourrait, par exemple, donner des informations sur la valeur relative à son contexte. On pourrait penser à un code Vert-Jaune-Rouge qui indique si la valeur est normale, limite ou anormale.

7.4 Interfaces et alertes

Au niveau de l'implémentation actuelle des interfaces (Mashups), il serait intéressant d'explorer un cadriciel plus mature et plus personnalisable. Bien que le but du projet ait été de démontrer la puissance, la polyvalence et le caractère "clé en main" de la plateforme ThingWorx à de futurs clients, les outils d'interfaces graphiques de la plateforme sont plutôt limitatifs au niveau de la personnalisation pour des développeurs en plus de ne pas offrir une performance satisfaisante. Cela dit, l'équipe propose donc d'utiliser un cadriciel d'interfaces web permettant une plus grande flexibilité et ainsi d'obtenir les données à afficher par des appels à l'API.

Au niveau des alertes, les fonctionnalités d'avertissement à l'aide de courriels ou de SMS n'ont également pas pu être réalisées. Afin de réaliser ces fonctions, il serait nécessaire de déterminer les bornes délimitant le fonctionnement normal de l'appareil, et ce, pour chacune des propriétés du turboréacteur. Ainsi, il serait possible de créer des événements dans la plateforme ThingWorx, ainsi qu'un Thing responsable de gérer le comportement de la plateforme par une souscription à ces événements.

Ce chapitre a couvert les différentes améliorations possibles sur l'ensemble des modules développés au cours de ce projet. Bien que certaines de ces améliorations étaient prévues au projet initialement, certaines autres offrent plutôt des pistes d'amélioration pour les équipes de Pfe futures ainsi que pour Matricis.

CHAPITRE 8 - CONCLUSION

Pour conclure, la réalisation de ce projet multidisciplinaire a demandé à l'équipe de réaliser cinq applications distinctes, lesquelles devaient communiquer entre elles à l'aide de divers protocoles. Un simulateur jouait le rôle d'un objet connecté réel, alors que la plateforme ThingWorx servait de point d'échange central pour les données. Une application de réalité augmentée offrait une vue alternative aux tableaux de bord développés à même la plateforme sur l'état des équipements analysés. Un module d'apprentissage prédictif, qui n'a pas pu être réalisé au cours de ce projet, visait à traiter l'analyse des données afin de prédire la durée de vie utile restante des appareils.

Après avoir élaboré une conception complète et passé quatre mois à développer le prototype expérimental, l'équipe est satisfaite d'avoir réussi à intégrer quatre des cinq modules initialement planifiés dans le projet. Sommes toutes, l'équipe est satisfaite des avancements réalisés au cours du projet. En effet, l'opportunité de produire un résultat comblant de réels besoins dans l'industrie a permis à l'équipe de se familiariser avec des situations où les exigences ne sont pas toujours aussi claires que dans les cas de travaux scolaires, spécifiquement dans des cas de recherche et développement à caractère exploratoire. L'équipe de travail est également satisfaite de la collaboration dont elle a pu bénéficier avec Matricis tout au long du projet.

Ce projet a permis aux membres de l'équipe de se familiariser avec certains enjeux de l'industrie 4.0, ainsi qu'avec certaines technologies IoT telles que la plateforme ThingWorx, qui leur étaient inconnues jusqu'ici. De la même façon, les principes liés à la réalité augmentée ont également pu être explorés.

RÉFÉRENCES

(Boost) Boost C++ Libraries: <https://www.boost.org/>

CMAKE : <https://cmake.org/>

(PTC) ThingWorx. Site officiel de PTC :
<https://www.ptc.com/en/products/iot/thingworx-platform>.

(Edge SDK) ThingWorx Edge SDKs and WebSocket-based Edge MicroServer (WS EMS)
Help Center : http://support.ptc.com/help/thingworx_hc/thingworx_edge_sdks_ems/.

(Extension SDK) Extension SDK v8.3: <https://marketplace.ptc.com/apps/193544/extension-sdk>

(Eclipse Plugin) Eclipse Plugin for ThingWorx Extensions:
<https://marketplace.ptc.com/apps/193295/eclipse-plugin-for-thingworx-extensions---version-720>

(Qt) Qt - Cross-platform software development for embedded & desktop : <https://www.qt.io/>

(Unity) Page officielle de l'engin 3D Unity : <https://unity3d.com/unity/>.

(Vuforia) Page officielle de la solution : <https://www.vuforia.com/>.

(Degré Rankine) Wikipédia : https://fr.wikipedia.org/wiki/Échelle_Rankine

BIBLIOGRAPHIE

- [1] Saxena, A., Goebel, K., Simon, D., & Eklund, N. (2008, October). Damage propagation modeling for aircraft engine run-to-failure simulation. In *IEEE 2008 international conference on prognostics and health management*, pp. 1-9.
- [2] Boost (bibliothèques). (2019, février 14). Wikipédia, l'encyclopédie libre. Page consultée le 08:47, février 14, 2019 à partir de [http://fr.wikipedia.org/w/index.php?title=Boost_\(biblioth%C3%A8ques\)&oldid=156732563](http://fr.wikipedia.org/w/index.php?title=Boost_(biblioth%C3%A8ques)&oldid=156732563)
- [3] FAQ C++. (2018). Qu'est-ce que Boost ? Développez, le club des développeurs et IT Pro. Repéré à <https://cpp.developpez.com/faq/cpp/?page=Boost#Qu-est-ce-que-Boost>
- [4] Khan, F., Eker, O., Khan, A., & Orfali, W. (2018). Adaptive Degradation Prognostic Reasoning by Particle Filter with a Neural Network Degradation Model for Turbofan Jet Engine. *Data*, 3(4): 49.

ANNEXE 1 - DIAGRAMMES

Voir le dossier en annexe pour une meilleure vue des figures présentées dans ce rapport.