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

PFE-022 DÉVELOPPEMENT D'UN SUIVI DE PRODUCTION INTELLIGENT ÉQUIPE SUPREMEX

KEVIN DELORME - DELK09088909
KEYVEN-STEVE LEGAGNEUR - LEGK30069401
QUENTIN PATAULT - PATQ11079204
VANESSA BAQUERO - BAQV29529200

DÉPARTEMENT DE GÉNIE LOGICIEL ET DES TI

Professeur-superviseur

ALAIN APRIL

MICHEL RIOUX

MONTRÉAL, 14 AOÛT 2019
ÉTÉ 2019

REMERCIEMENTS

L’équipe tient à remercier toutes les personnes qui ont permis que ce projet se réalise, leur contribution a été grandement appréciée.

Pour commencer, nous voudrions remercier la compagnie SupremeX pour nous avoir permis de réaliser le projet dans leurs installations et d’avoir cru dans notre projet, en particulier à monsieur Wassim Azem, directeur de la technologie chez SupremeX.

Nous aimerions remercier les deux professeurs qui nous ont encadrés tout le long de la session et qui nous ont aidés dans nos démarches. Merci au professeur Alain April du département de Génie logiciel et merci au professeur Michel Rioux du département de Génie des systèmes, leurs connaissances dans des domaines spécifiques nous ont été très utiles.

Et finalement, nous voudrions remercier François Blackburn-Grenon, doctorant à l’École de technologie supérieure, qui nous a fait part de ses connaissances sur internet des objets (IdO) pour ce projet.

PFE-022 DÉVELOPPEMENT D'UN SUIVI DE PRODUCTION INTELLIGENT
ÉQUIPE SUPREMEX

KEVIN DELORME - DELK09088909
KEYVEN-STEVE LEGAGNEUR - LEGK30069401
QUENTIN PATAULT - PATQ11079204
VANESSA BAQUERO - BAQV29529200

RÉSUMÉ

Notre équipe a eu comme mandat de développer un système de production intelligent afin de répondre à une problématique chez SupremeX. L'entreprise désire diminuer les tâches manuelles et opter pour la collecte automatique des données-machine afin de déterminer le temps de production, le temps de réglage des commandes et le temps que la machine soit non utilisée.

Le prototype de collecte des données a été installé sur une machine et il a été possible de prouver que c'était possible de recevoir les informations grâce à un dispositif IoT, de les traiter dans ce dispositif et de les envoyer au serveur pour les enregistrer sur une base de données et créer un tableau de bord pour les utilisateurs cibles.

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

C# : C Sharp
ÉTS : École de technologie supérieure
HTTP: Hypertext Transfer Protocol
I2C: Inter Integrated Circuit
IDE : Environnement de développement
IdO (IoT) : Internet des objets
KPI: Key performance Indicator
UART: Universal Asynchronous Receiver Transmitter
USB: Universal Serial Bus
MSSQL: Microsoft SQL
PFE : Projet de fin d'études
PGI : Progiciel de gestion intégré
SPI: Serial Peripheral Interface
SQL: Structured Query Language
SSH: Secure shell
MOSI: Master Output, Slave Input
MISO: Master Input, Slave Output
GPIO: General Purpose Input/Output
SPI : Serial Peripheral Interface

LISTE DES SYMBOLES ET UNITÉS DE MESURE

Volt (V) : Unité de mesure électromotrice et de différence de tension électrique.

INTRODUCTION

L'entreprise SupremeX, où travaille notre chef de projet Vanessa Baquero, est un chef de file nord-américain dans la fabrication et la distribution d'enveloppes. Celle-ci fournit aussi des solutions d'emballage. SupremeX, c'est 12 installations réparties dans sept provinces au Canada et trois aux États-Unis employant 830 personnes.

Ces installations sont des chaînes de production qui nécessitent du personnel pour faire fonctionner les machines et relever l'état des machines manuellement. SupremeX voudrait pouvoir avoir une vision de la productivité de machines les plus importantes dans la ligne de production en temps réel par le biais de son progiciel de gestion intégré (PGI), Microsoft Dynamics.

Présentement, tout est fait manuellement par les employés dans le PGI. Une solution existe sous la forme d'un module développé par des partenaires suisses, mais SupremeX voudrait d'abord tester une solution à l'interne.

La donnée importante à récolter est le temps, et non la capacité de production, pour diminuer le coût de production. Cela comprend la mesure de :

- temps de production : nombre d'heures où l'équipement a produit des enveloppes ;
- temps de mise en route : nombre d'heures pour initialiser l'équipement.

Actuellement, un employé appuie sur bouton start (début) pour indiquer le début de sa tâche sur la machine puis sur end quand son travail est terminé. La différence entre le temps de début et le temps de fin permet de déterminer la durée du travail réalisé pour produire une certaine quantité d'enveloppes. Sauf que lorsqu'un employé, et cela arrivent trop souvent d'après la direction de SupremeX, oublie d'appuyer sur end, car le système continue d'enregistrer le temps, mais la machine ne produit plus. En conséquence, ces productions se retrouvent avec un coût énorme. Dans leurs rapports financiers, ces commandes de production sont identifiées comme non rentables alors que cela ne reflète pas la réalité. L'erreur est humaine et pourrait être évitée.

Ces machines ne sont pas conçues pour exporter les données qu'elles génèrent, et il est donc nécessaire de développer une interface afin de récolter ces données afin de les envoyer au PGI. Notre projet serait de récolter les données, de les envoyer dans une machine virtuelle dans le réseau interne de l'entreprise et de les sauvegarder dans une base de données pour sortir de statiques.

SupremeX voudrait que ces données soient analysées et que les résultats de ces analyses puissent être consultables sur un tableau de bord à destination des employés. L'interface de récolte et d'envoi des données serait développée en interne et testée tout d'abord sur certaines machines.
Une fois la partie récolte de données validée, notre équipe mettra en place une logique d'analyse de ces données de sorte qu'il soit possible de les consulter sur un tableau de bord développé pour les besoins de ce projet.

CHAPITRE 1

DESCRIPTION DU PROJET

1.1 Objectifs

Au cours de la session d'été 2019, nous planifions de mettre en place une plateforme matérielle externe qui permettra d'acquérir des données issues de machines d'une chaîne de production. Ces données pourront ensuite être utilisées afin de mesurer la performance des machines dans un premier temps, puis idéalement, de l'ensemble des machines d'un même site de production, et de fait, les autres sites de production. Les machines ne disposent pas toutes des interfaces de sortie depuis lesquelles il est possible d'acquérir des données relatives à leur état, nous prévoyons d'ajouter à cette plateforme des dispositifs d'acquisition, au besoin.

Conformément avec le contexte actuel, les objectifs du présent projet sont donc :

- le développement d'une interface de collecte des données d'état provenant de machines de production;
- la mise en place d'une logique d'acheminement de ces données vers le progiciel de gestion intégrée de l'entreprise (objectif finalement abandonné, SupremeX voulant une preuve de concept rapide et n'ayant qu'un seul développeur autorisé sur Microsoft Dynamics);
- la mise en œuvre d'une procédure de stockage de ces données vers la base de données de la machine virtuelle;
- le développement d'une logique d'analyse de ces données d'état;
- la construction d'un tableau de bord permettant de consulter ces données et les analyses associées.

À terme le projet aboutira sur la mise en place d'un système IoT permettant d'observer la productivité d'une chaîne de production sur la base de données issue des machines connectées, ce qui permettra à SupremeX d'avoir une meilleure visibilité de l'état des machines de la chaîne de production, et donc de la performance, ce qui devrait conduire à une amélioration de la gestion en interne.

1.2 Personnes ressources

Dans le tableau suivant se retrouvent les membres de l'équipe de projet et les personnes ressources qui ont été appelées à interagir avec l'équipe lors de la session :

Prénom	Rôle(s)	Responsabilités
Vanessa Baquero	Chef d'équipe	• S'assurer du respect des objectifs du projet en lien avec les demandes de l'entreprise • Communication avec l'entreprise partenaire • S'assurer du respect des normes de sécurité défini par l'entreprise • Suivi de l'avancement du projet avec l'entreprise
Keyven-Steeve Legagneur	Responsable intégration	• Définir les outils de développement • S'assurer que ces outils soient adéquats pour atteindre les objectifs du projet et prendre des décisions en conséquence si ce n'est pas le cas
Quentin Patault	Responsable projet	• Planifier les différentes tâches du projet • Faire le suivi de l'avancement des itérations et des tâches • Définir les tâches les plus importantes et orienter les efforts des membres
Kevin Delorme	Responsable technologie	• Définir les technologies qui seront utilisées • S'assurer que ces technologies sont adéquates pour atteindre les objectifs du projet et prendre des décisions en conséquence si ce n'est pas le cas
Alain April	Professeur	Ressource de l'ETS chargée de la supervision du projet.
Michel Rioux	Professeur	Ressource de l'ETS chargée de la supervision du projet.
Wassim Azem	Directeur de la technologie	Approuve les requis et les objectifs du projet.
Suzie Gaudreault	Vice-présidente	Remplacera Wassim quand il sera en vacances.
Stéphane Simard	Directeur de l'usine	Responsable de l'usine et des employés.
Pierre Bernier	Électricien	Ressource pour l'emplacement des capteurs sur les équipements.

Tableau 1: Personnes ressources

CHAPITRE 2

GESTION DE PROJET

2.1 Méthodologie

Pour atteindre les objectifs de notre projet, nous utiliserons différents outils, expliqués dans le chapitre 3 sur les technologies, et la méthodologie Kanban.

Nous avions opté au début pour la méthodologie Agile/Scrum puis après notre première réunion avec nos professeurs, nous avons finalement choisi la méthodologie Kanban.

Les avantages de la méthodologie Kanban par rapport à Scrum sont multiples. Kanban ne définit pas les rôles et les responsabilités particulières. Dans notre cas, nous avons défini Quentin Patault comme gestionnaire du projet, car nous avons quand même besoin d'une personne ayant une vision claire et précise de la progression des tâches de notre projet en tout temps. Par conséquent, les membres de notre équipe sont encouragés à collaborer et à prêter main-forte à un coéquipier si celui-ci rencontre un problème bloquant dans une tâche.

Les dates de livraison de nos tâches seront définies selon les besoins du projet et au fur et à mesure de son avancement. Donc nous n'aurons pas de périodes définies durant lesquelles une tâche devra être terminée pour être validée. Ces tâches peuvent être modifiées tout au long du projet selon l'évolution des exigences du projet.

Aussi, afin de suivre la charge de travail effectuée sur le projet, nous avons créé un fichier Excel de suivi des heures des membres de l'équipe[1]. Nos membres doivent ajouter les heures réalisées ainsi que les tâches sur lesquelles ils ont travaillé. Un récapitulatif hebdomadaire et un récapitulatif mensuel seront utilisés pour assurer un suivi.

2.2 Livrables

Une liste des artéfacts qui seront livrés pour le projet de fin d'études est présentée ici :

Nom de l'artefact	**Description**
Plan de projet	Une explication de la problématique, du contexte, des objectifs du projet, de la méthodologie qui sera utilisée, la composition de l'équipe et les rôles, des livrables , de la description des artefacts, des risques et des techniques et outils qui seront utilisés.
Rapport final	Décris les résultats obtenus du PFE.
Présentation	Une présentation orale du projet et du résultat final.
Outil de suivi (Trello)	Outil permettant de suivre la méthodologie Kanban pour les tâches liées au projet.
Document de suivi des heures	Fichier Excel comptabilisant les heures des membres de l'équipe durant toute la durée du projet.
Code source niveau	Code en Python présent sur la caméra OpenMV.

[1] Annexe II

caméra	
Code source niveau IoT	Code en C# ou Python sur notre contrôleur connecté à notre caméra OpenMV.
Code source niveau BD	Code en SQL pour les requêtes sur la base de données.
Architecture système	Diagramme expliquant l'architecture du système voulant être réalisé par notre équipe.

Tableau 2: livrables

2.3 Risques

Il est important de comprendre les risques reliés à un projet avant de le mettre en œuvre, car on pourra ainsi évaluer leur probabilité, leur impact et faire un plan d'atténuation.

Risque	Impact	Probabilité	Mitigation / atténuation
Délai sur l'installation des capteurs	Impossibilité de démontrer le concept en production	Haute	Prototype statique à faire.
Délai d'apprentissage pour la reconnaissance de caractères si le capteur est une caméra	Retard dans l'implémentation	Moyenne	Essai d'autres bibliothèques ou d'autres techniques de reconnaissance.
Délai sur la disponibilité du technicien	Retard / impossibilité d'avancer dans la conception	Basse	Travail en temps supplémentaire
Résistance aux changements de la part des opérateurs	Contournement du processus	Moyenne	Parler avec le syndicat, le directeur d'usine et les chefs de chacun des quarts de travail. Ne pas expliquer l'objectif réel du capteur, se limiter à la partie de collecte de données de l'équipement.
Stabilité de la solution en production	Bris ou panne.	Haute	Faire plusieurs essais pour s'assurer de la stabilité du système.

Tableau 3: Risques

CHAPITRE 3

TECHNOLOGIES ET OUTILS

3.1 Outils de gestion de projet

La gestion de projet est une partie importante, c'est celle qui garantira la réussite du projet. Plusieurs outils sont mis en place pour permettre au chargé de projet et à toute l'équipe de garder un contrôle sur les différents livrables. Les outils qui ont été utilisés ont été adaptés a la méthodologie Kanban qui a été préalablement expliquée.

Voici la liste d'outils utilisés :

- La suite bureautique de Google pour pouvoir travailler de pair sur nos livrables;
- Lucidchart pour l'élaboration de nos diagrammes;
- Calendrier de Google pour la gestion de nos rencontres et réunions;
- Slack pour communiquer entre les membres;
- Trello pour gérer notre tableau Kanban;
- Fichier de suivi des heures.

Certains outils seront détaillés plus en amplement dans les sections qui suivront.

3.1.1 Trello

Trello est l'outil de gestion de projet qui a été sélectionné lors du projet. C'est un outil en ligne ce qui facile l'accès, car il n'est pas nécessaire de l'installer sur un poste de travail, simplement utiliser un navigateur pour l'ouvrir.

En ce qui concerne notre tableau Kanban, celui-ci a été divisé en 5 sections :

1. Tâches à faire : Contenant les tâches prévues du projet;
2. Conception en cours (maximum de 8) : Contiens les tâches de conception en cours de réalisation;
3. Dév en cours (maximum de 4) : Contiens les tâches de développement en cours de réalisation;
4. Bloquants (maximum de 8) : Contiens les tâches non terminées dont dépendent d'autres tâches qui ne peuvent donc pas être commencées;
5. Tâches terminées (maximum de 8) : Contiens les tâches qui ont été terminées par les membres (cette section est régulièrement nettoyée en archivant les tâches qui y sont incluses);
6. Production : Contiens les tâches de développement réalisé et ayant été validées pour être dans l'environnement de production.

Voici un aperçu du tableau, lors d'une des dernières mises à jour :

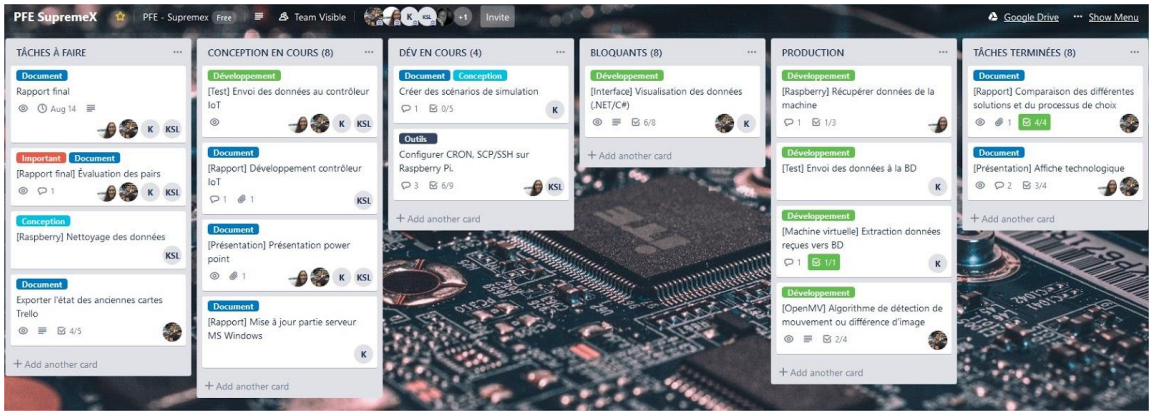


Figure 1: Aperçu du tableau Trello

Nous avons également mis en place les étiquettes suivantes pour distinguer plus clairement les différentes tâches dans les cartes (2 catégories des étiquettes) :
Catégories de cartes:

- Conception
- Développement
- Document
- Outils
- Réunions

Statuts des cartes:

- Important
- En attente
- A valider

Chaque tâche complétée dans la section « Dév en cours » devra être validée par un autre membre afin de pouvoir être archivée. Ce tableau a été utilisé pour définir la priorité des différentes tâches, les tâches en première position étant les plus prioritaires.

Cette méthodologie nous permettra de savoir facilement quelles sont les tâches en cours, lesquelles sont à réaliser le plus rapidement et par conséquent, nous permettra de réaliser le travail plus rapidement.

3.1.2 Fichier de suivi des heures

Afin de faire un bon suivi du temps consacré par chaque membre de l'équipe dans le projet, un système de gestion du temps a été mis en place. Lorsque les membres de l'équipe travaillaient sur le projet, ils inscrivaient les heures dans un fichier Google Sheets. Ceci facilitait le suivi et nous a permis de sortir des statiques KPI.

Dans l'annexe II, il est possible de retrouver le détail par mois et par personne des heures travaillées sur le projet. Ce sont ces informations, qui ont servi comme base pour la préparation du graphique ci-dessous.

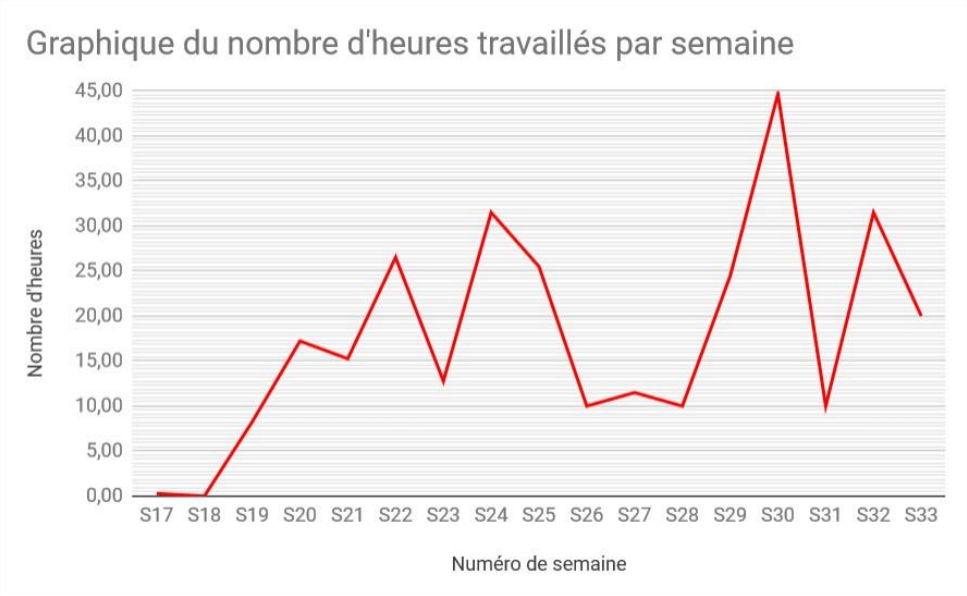


Figure 2: Graphique du nombre d'heures travaillées par semaine

Nous pouvons observer dans la figure précédente, les variations qui ont eu lieu tout le long de la session. Notamment, au niveau des semaines 26 à 28 où plusieurs de nos membres avaient des examens et des travaux à remettre. La baisse du nombre d'heures à la semaine 31 correspond aux quelques jours que certains de nos membres ont pris comme vacances.

Au total, c'est 303,15 heures qui ont été investies dans le projet pour une moyenne par semaine de 17 heures.

3.2 Outils de développement

Le projet requiert certains outils de développement propres à chaque composant qui ont été utilisés. Voici la liste d'outils :

- OpenMV IDE
- Raspbian pour le Raspberry Pi
- Visual Studio
- Git Hub
- Machine virtuelle Vmware
- Microsoft SQL

3.2.1 OpenMV IDE

Un langage de programmation allégé issu de Python, MicroPython, a été utilisé, pour contrôler le comportement souhaité et l'envoi des données vers le dispositif IoT par le biais de l'environnement de développement OpenMV IDE.

3.2.2 Rapsbian et Python pour le Raspberry Pi

Afin d'exploiter le Raspberry Pi, le système d'exploitation Raspbian et pour des raisons de conformité envers l'entreprise SupremeX, un programme en Python sera utilisé pour le programme qui sera installé sur l'objet IOT (Raspberry Pi).

3.2.3 Visual Studio

Cet environnement de programmation sera utilisé côté serveur pour développer l'interface graphique en C# et pour créer les scripts qui seront utilisés dans PowerShell pour la gestion des fichiers reçus du Raspberry Pi.

3.2.4 Git Hub

Pour l'hébergement du code source, un dépôt privé sera utilisé par les membres de l'équipe pour déployer les différentes versions du code des programmes que nous développerons dans ce projet.

3.2.5 Machine virtuelle

Enfin, pour assurer de ne pas être bloqués au niveau de l'avancement du projet par rapport aux accès à l'environnement réel de production, des accès à une machine virtuelle nous seront alloués. Cela permettra de pouvoir tester le prototype dans un environnement réel et d'avoir des résultats pertinents. Cette machine virtuelle utilisera un système d'exploitation Windows Server 2012.

3.2.6 Microsoft SQL

La base de données qui a été mise à la disposition de l'équipe par l'équipe est celle de Microsoft. Le langage SQL utilisé est propre à mssql et nous permettra de créer et gérer notre base de données.

CHAPITRE 4

CAPTEURS

4.1 Contexte

SupremeX possède de nombreux équipements pouvant réaliser différents formats et impression d'enveloppes. La plupart de ces équipements possèdent un écran de contrôle. L'entreprise voudrait pouvoir mesurer la performance de ces équipements en temps réel sans avoir à envoyer des techniciens relever l'information.

Deux choix s'offraient à l'équipe, soit d'installer des capteurs directement sur l'équipement et à des endroits précis qui mesureront les données désirées. Soit d'installer des caméras de type OpenMV qui interpréteront les données des tableaux de bord des équipements.

Lors de notre rencontre avec le directeur des technologies chez SupremeX, il a été vérifié au niveau légal s'il était possible d'utiliser des caméras pour récupérer l'information des écrans des machines. La seule consigne qui a été reçue est de ne pas filmer la provenance des équipements. Pour ce qui est de l'installation de capteurs, l'entreprise n'y a pas vu de problème.

Afin d'obtenir toutes les informations nécessaires pour développer le suivi de production, il a été décidé d'utiliser un capteur de proximité, un capteur pour la vitesse du moteur, une caméra et un clavier. Dans ce chapitre, on traitera du capteur de vitesse et du capteur de proximité seulement.

4.2 Installation des capteurs

L'installation implique une connaissance approfondie de chaque type d'équipement au niveau de l'architecture électrique. Les usines ayant plusieurs types de machines pour réaliser différents formats d'enveloppes, les emplacements des capteurs ne seront pas les mêmes suivant l'équipement. En conséquence, l'application de cette solution à l'échelle de toutes les usines de l'entreprise pourrait s'avérer longue, fastidieuse et source de problème.

Malgré tout, cette solution a été préférée au détriment de celle des caméras, car elle permet d'obtenir des données plus fiables sur l'équipement.

4.3 Vitesse moteur

La première donnée qui sera collectée sera la vitesse du moteur de la machine qui a été sélectionnée. Cette information nous permettra principalement de savoir si le moteur est en marche ou non et de donner la vitesse à laquelle la machine fonctionne. Ce capteur a été installé dans le moteur de la machine et nécessitait d'une prise 24 V afin de le faire fonctionner. Par contre, le capteur était placé loin du dispositif IoT, ce qui nous fait craindre d'une perte de données, à cause du long fil utilisé.

Figure 3: S8VK-C06024

4.4 Capteur de proximité

Le deuxième capteur sera utilisé dans le but de compter le nombre d'enveloppes qui passe devant le capteur. Ce capteur est installé sur la machine et lorsqu'il détecte une nouvelle enveloppe, il envoie un signal qui sera interprété par le Raspberry Pi. C'est un capteur photoélectrique qui est alimenté par le Raspberry Pi lui-même.

Figure 4: FE7B-DD6-M

4.5 Convertisseur

Les deux capteurs qui ont été installés aux abords de la machine doivent maintenant acheminer les données au Raspbery Pi, et pour ce faire nous allons utiliser un convertisseur PCF8591 qui servira de bus. Un des capteurs a été connecté sur AIN0 et l'autre sur AIN1. Les deux autres ports resteront libres.

Le protocole qui est utilisé par le convertisseur est l'i2c et c'est de cette façon que les données seront envoyées au Raspberry Pi.

Figure 5: PCF8591

4.6 Montage final

Le montage du prototype final est temporaire et il serait impératif de l'améliorer si SupremeX désire les garder à long terme. Comme on peut le voir dans la figure, les fils sont lousses et le montage est près de la machine qui fait subir au montage des vibrations constantes. Ces conditions combinées avec la chaleur de l'usine et la poussière peuvent causer des bris aux matériaux électroniques. Par contre, pour ce qui est du prototype nous croyons que c'était un bon choix, car c'était facile à configurer et à mettre à jour.

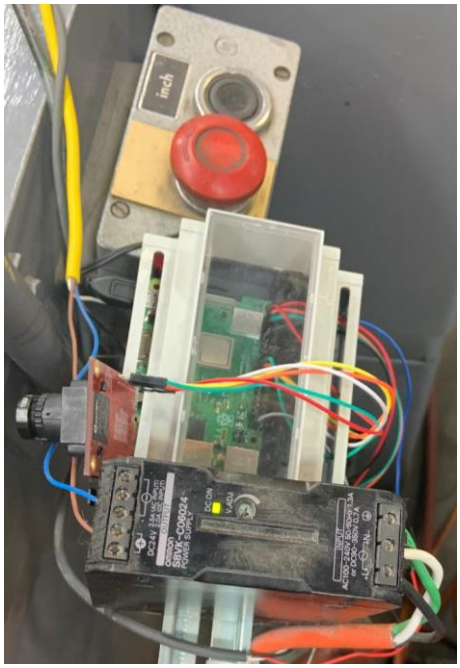


Figure 6: Montage final

CHAPITRE 5

CAMÉRAS

5.1 Introduction

La plupart des équipements possédant un écran de contrôle où des informations sur l'équipement sont affichées sous forme de chiffres ou de D.E.L.. Ces informations peuvent être détectées par une caméra placée devant l'écran de contrôle. Les données récoltées seraient envoyées à un microcontrôleur.

Lors de notre rencontre avec le directeur des technologies chez Supremex, nous avons vérifié au niveau légal si nous pouvions utiliser des caméras pour récupérer l'information des écrans des machines. La seule consigne que nous avons eue est de ne pas filmer la provenance des équipements.

5.2 Caméra

Nos professeurs nous ont conseillé de regarder du côté de la compagnie OpenMV fabriquant des petites caméras programmables.
OpenMV est un projet de caméra programmable très abordable issu d'une campagne de financement sur Kickstarter qui a eu beaucoup de succès[2].
Cette petite caméra se rapproche des projets comme les plateformes Raspberry ou Arduino. Celle-ci est facilement programmable dans une version allégée du langage de programmation Python appelé MicroPython[3].

[2] https://hackaday.com/2016/12/29/the-story-of-kickstarting-the-openmv

L’objectif de ce projet était de créer une caméra peu chère, facilement programmable et comprenant un microcontrôleur puissant et une caméra pouvant enregistrer des images JPEG[4]. La caméra est utilisée dans de nombreux projets de vision machine. En effet, celle-ci permet de détecter les contours, les formes, les images, les caractères, les différences de couleurs, etc.
Grâce à son succès, la compagnie OpenMV a sorti plusieurs versions de sa caméra.

Cette méthode de récolte de données est assez récente dans l’industrie et apprécié des spécialistes pour sa polyvalence, son faible impact sur la structure de l’équipement et son utilisation de l’apprentissage profond pour la détection des caractères.

5.3 Algorithme de détection de caractères

La caméra utiliserait un algorithme de détection de caractères avec un jeu de données, le chars74k pour détecter les chiffres et lettres du l’écran de contrôle de l’équipement et les envoyer au microcontrôleur.

La reconnaissance de caractère ne date pas d’hier. C’est un projet abordé par les spécialistes depuis les débuts de la vision par ordinateur.

Chars74k est un jeu de données ou en anglais, dataset. Ce jeu de données contient :

- 64 classes (0-9, A-Z, a-z) ;
- 7705 caractères obtenus à partir d'images naturelles ;
- 3410 caractères dessinés à la main à l'aide d'une tablette ;
- 62992 caractères synthétisés à partir de polices informatiques.

---

[3] http://micropython.org/

[4] https://openmv.io/pages/

Ce jeu de données nécessite une mémoire de stockage conséquente sur l'appareil qui l'implémentera ainsi qu'un processeur capable de faire du traitement d'image complexe en temps réel.

5.4 Solution

Nous avons testé en premier le modèle M4 V1 d'OpenMV. C'est une caméra qui nous a été fournie par le professeur, Michel Rioux.

5.4.1 Points importants

Après plusieurs tests, nous avons remarqués identifiés plusieurs points à considérer (voir Annexe III) :

- La M4 V1 possède peu de mémoire interne ;
- La M4 V1 possède un microprocesseur dont la puissance n'est pas adéquate pour du traitement complexe d'image.

Aussi, on peut remarquer que la M7 et la H7 possèdent la fonctionnalité de détection de rectangle qui nous sera utile pour détecter des enveloppes (voir Annexe IV).

5.4.2 Première version

Compte tenu de ces points, la première version de notre solution comprenait :

- une caméra OpenMV M4 V1 ;
- un microcontrôleur Raspberry Pie ou Arduino ;
- un module réseau Ethernet ou WIFI sur le Raspberry Pie ;
- un serveur avec OpenCV ;
- une base de données Microsoft SQL.

La caméra capturerait une image de l'écran de l'équipement puis l'enverrait au microcontrôleur Raspberry Pie/Arduino.

Le microcontrôleur se chargerait ensuite de l'envoyer à notre serveur par le biais de son module réseau Ethernet ou WIFI.
C'est le serveur qui analyserait l'image reçue avec OpenCV, extracterait les données de celle-ci et les enverrait à la base de données Microsoft SQL.

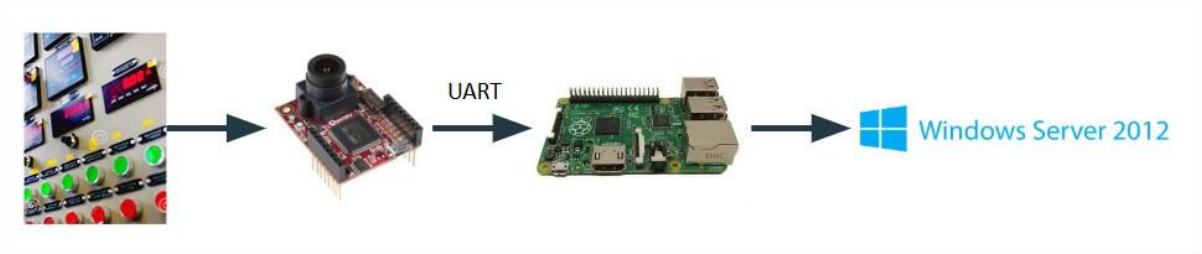


Figure 7 : Diagramme simplifié de la structure de la solution initiale

On peut résoudre le manque de mémoire interne en branchant une carte micro SD sur la caméra, mais on se heurte toujours au problème du microprocesseur.

En conséquence, l'analyse de l'image est une tâche attribuée au serveur, car le modèle M4 V1 d'OpenCV ne possède pas les caractéristiques nécessaires (puissance et mémoire interne) pour faire efficacement du traitement d'image en temps réel dans un milieu industriel.

Une deuxième caméra serait présente pour détecter la présence de l'opérateur en train de travailler en utilisant un algorithme de détection de visage.

5.4.3 Deuxième version

Après comparaisons des différentes caméras, nous avons décidé de prendre la caméra la plus performante, la H7. En effet, celle-ci comprend un algorithme de détection de caractères déjà entraîné sur le jeu de données chars74k.
Notre solution s'est donc simplifiée grandement. La caméra détecterait les caractères en temps réel et enverrait au Raspberry Pie les données récoltées pour ensuite les envoyer dans un format adéquat au serveur.

5.4.4 Troisième version

Pendant le développement, nous nous sommes aperçus que l'implantation pour détecter de multiples caractères de différentes tailles, de différentes couleurs ou encore de différentes formes, à des endroits différents dans la vue n'était pas faisable en l'espace de 3 mois.

Après discussion avec l'équipe, nous ne nous voyons pas développer la solution, la tester, identifier les problèmes, trouver des solutions, résoudre les problèmes, re-tester et ainsi de suite alors qu'il y avait une solution bien plus simple à notre portée.
En conséquence, nous avons pris la décision d'installer des capteurs sur les équipements pour récolter les données de mise en production et d'arrêt de la machine et d'utiliser les caméras pour détecter la présence d'un opérateur devant l'équipement et détecter les enveloppes produites par l'équipement.

Pour détecter la présence d'un opérateur, nous utilisons la détection de visage intégré à la caméra.

Pour les enveloppes, nous utilisons la détection de mouvement en combinaison avec la détection de rectangle que j'ai énoncé lors de la comparaison des caméras.

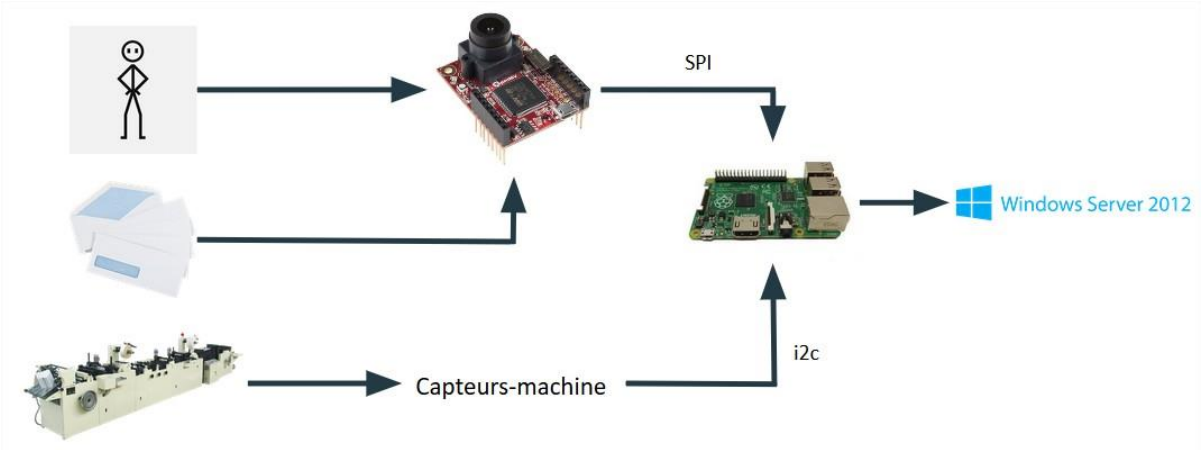


Figure 8 : Diagramme simplifié de la structure de la solution finale

CHAPITRE 6

RASPBERRY PI

6.1 Choix du système d'exploitation pour le Raspberry Pi

Le Raspberry Pi supporte plusieurs types de systèmes d'exploitation dont la majorité sont des distributions Linux telles que Raspbian, Kali Linux, Arc Linux, etc. Dans le cadre de ce projet, seulement deux systèmes d'exploitation ont été considérés. La première étant le système d'exploitation « Rasbian », car il est reconnu dans l'industrie pour être léger et facile d'utilisation. Le deuxième système envisagé a été « Windows IOT ». Une contrainte technologique imposée par le bénéficiaire du projet (SupremeX) nous obligeait à considérer ce système d'exploitation, car la compagnie utilise majoritaires des systèmes Microsoft. Afin d'assurer une intégration plus simple avec les autres systèmes, SupremeX préférait nettement une solution Microsoft. Suite à des tests fonctionnels des deux solutions, le choix s'est rapidement arrêté au système d'exploitation Rasbian. Les prochaines sections décrivent les observations ressorties de ces tests entre les différents systèmes d'exploitation.

6.1.1 Test avec le système d'exploitation Rasbian

L'installation du système d'exploitation Rasbian s'effectue sur la carte SD. Il suffit d'extraire l'image à la racine de la carte SD et redémarré le Raspberry avec celle-ci inséré. Lors du premier démarrage, des étapes de configurations de base sont présentées à l'écran tel que la définition du mot de passe administrateur (root), la configuration du fuseau horaire, la mise à jour des librairies de logiciel, etc. L'utilisation de l'interface utilisateur est très fluide et conviviale. Ce système possède un indicateur pour la consommation électrique. Il a été remarqué que la consommation dépassait très rarement le 15 %. Le développement des scripts sur ce système est très simple à s'approprier. Il supporte les scripts Python autant que les scripts Shell.

6.1.2 Test avec le système d'exploitation Windows IOT

Pour installer le système d'exploitation Windows IOT, il faut préalablement installer le logiciel « Windows 10 IoT Core Dashboard » de Microsoft. Il faut ensuite suivre les étapes du GUI afin d'installer le système d'exploitation « Windows IOT » sur la carte SD. Le démarrage du système s'effectue automatiquement sans avoir besoin de faire une manipulation particulière. Ensuite, il faut appliquer certains paramètres tels que le mot de passe administrateur, le fuseau horaire, l'autorisation de la localisation, etc. Pendant les premiers tests, la navigation dans les différents menus a été particulièrement difficile. Les transitions entre les options n'étaient pas assez fluides. Il y avait peu d'application « built in » comparée au système Rasbian.

6.2 Connexion des différents équipements au Raspberry Pi

Le Raspberry Pi est un point central pour la réception des différents types de données. Il possède plusieurs types d'entrées telles que les ports USB, les GPIO, une connexion WIFI et une connexion Bluetooth. Dans la section qui suit, nous expliquerons comment les intrants de données sont envoyés au Raspberry.

6.2.1 Intrant USB

Le seul équipement connecté au Raspberry via le port USB est un clavier numérique. Ce clavier est utilisé essentiellement pour être en mesure de récupérer les numéros de commande saisie par les opérateurs des machines. L'une des premières idées était d'avoir un clavier possédant des D.E.L. rétro éclairé sur les touches pour offrir un retour visuel à l'utilisateur. Par exemple, lorsque les D.E.L. sont actives, l'opérateur peut saisir un numéro de commande. Lorsqu'aucune D.E.L. n'est allumée, aucune saisie au clavier ne sera prise en compte.

Figure 9 : Prototype du clavier numérique rétro éclairé

Pour le produit final de la preuve de concept, un clavier numérique standard a été utilisé. Il répond au besoin primaire qui est l'envoi de donnée saisie par l'opérateur. Un enjeu avec le temps alloué pour avoir une solution fonctionnelle explique entre autres pourquoi la première idée n'a pas été retenue.

Figure 10 : Clavier numérique

6.2.2 Intrant GPIO : Série

La connexion série en utilisant les GPIO est très utile pour différent contexte. Elle permet d'établir une connexion qui facilite l'échange de données entre 2 équipements. Peu de paramètres se doivent d'être définis pour établir une connexion. La vitesse de transmission, la grosseur des « bits », le timeout sont quelques paramètres à définir dans l'objet dans un script en Python.

```
9   ser = serial.Serial (
10   port='/dev/ttyS0',
11   baudrate = 9600,
12   parity=serial.PARITY_NONE,
13   stopbits=serial.STOPBITS_ONE,
14   bytesize=serial.EIGHTBITS,
15   timeout=1
16  )
```

Figure 11: Extrait du code pour définir un objet « Serial » en Python

L'un des équipements que nous avons testés en utilisant une connexion série est la caméra OpenMV. La transmission des données entre le Raspberry et celle-ci s'effectue sans problème. La « Pin » 14 du Raspberry est utilisée pour le transfert de données (Tx) et la « Pin » 15 est utilisée pour la réception des données (Rx). Sur la caméra OpenMV, c'est la « Pin » P1 pour la transmission de donnée et la « Pin » P0 pour la réception des données. Il faut connecter les « Pin » Tx et Rx ensemble entre les 2 équipements et alimenter l'équipement grâce à la « Pin » 1 (3,3 V), 2 (5V) et 4 (5V). Le voltage est important afin d'éviter d'endommager l'équipement.

6.2.3 Intrant GPIO : SPI

La connexion SPI utilise un concept très connu et utilisé depuis très longtemps du nom de «Master/Slave ». En termes de flexibilité, ce n'est pas le meilleur type de connexion à implémenter entre deux équipements. Cependant, pour le besoin du projet, ce type de connexion était l'idéal par sa simplicité d'implantation et de fonctionnement.

```
17  #Initialisation SPI
18  spi = SPI(2, SPI.SLAVE, baudrate=600000, polarity=1, phase=0, crc=0x7)
```

Figure 12: Extrait du code pour définir un objet « SPI » Slave en Python

Des tests avec la caméra OpenMV ont été effectués avec ce type de connexion. Il a été décidé surtout à cause de sa simplicité de conserver ce montage pour la preuve de concept de notre solution. Le code est légèrement réduit et le fonctionnement est ajusté parfaitement au besoin du projet. Les « Pin » 10(MOSI), 9 (MISO), 11 (SCLK) et 8 (CEO) ou 7 (CE1) sont utilisés pour la connexion SPI au niveau du Raspberry. Sur la caméra OpenMV, les « Pin » P0(MOSI), P1(MISO) et P2(SCLK) peuvent être utilisés pour la connexion.

6.2.4 Intrant GPIO : i2c

La connexion i2c via les GPIO permet d'échanger des données numériques entre un capteur et un appareil tel que le Raspberry. C'est un bus qui n'est pas activé par défaut. Il suffit de se rendre dans l'interface de configuration du Raspberry (raspi-config) pour permettre ce type d'échange de données. Nous avons utilisé ce type de connexion avec le capteur branché à la machine afin d'interpréter les pulsions électriques. Les « Pin » 3 et 5 sont dédiés à la connexion i2c.

```
addresse_0 = 0x48 # adresse i2c du module PCF8691
entree_ain0 = 0x40  # utiliser 0x40 pour A0
addresse_1 = 0x48 # adresse i2c du module PCF8691
entree_ain1 = 0x41  # utiliser 0x41 pour A1

bus = smbus.SMBus(1) # définition du bus i2c
```

Figure 13: Extrait du code pour la connexion avec i2c

CHAPITRE 7

SERVEUR

7.1 ETL

Avant d'utiliser le sigle ETL, il convient de le définir, car il correspond à la solution mise en place sur le serveur (Windows Server 2012 R2). ETL est l'acronyme utilisé pour l'extraction des données en anglais c'est: Extract-transform-load.

Le principe est d'avoir un interlogiciel qui permet de faire le transfert des données entre différentes sources. Dans notre cas, nous voulons utiliser l'interlogiciel pour transférer les données des fichiers texte qui sont sur le serveur pour les ajouter à la base de données.

7.2 Choix technologiques initiaux

La conception initiale de la solution sur le serveur Windows comportait la mise en place de plusieurs éléments :

- Un système de transfert direct de fichiers au format XML ;
- Un système d'interprétation du contenu de ces fichiers ;
- Une base de données ;
- Un système d'alimentation de cette base de données.

Par le choix initial d'effectuer le développement en utilisant le langage Python du côté microcontrôleur, il avait d'abord été décidé d'utiliser le même langage sur le serveur Windows. La conception initiale de l'architecture cible comportait la mise en place d'un micro service en Python chargé de réaliser les traitements sur le serveur, c'est-à-dire l'analyse du contenu du fichier, la construction des requêtes d'insertion de données dans la base de données et l'insertion de ces données.

Au niveau des fichiers transférés, le langage XML avait été retenu, car ce format semblait correspondre aux besoins du projet. Cependant, après plusieurs essais et en revoyant l'analyse de la conception proposée, ce choix n'a finalement pas été retenu, car la structure des données contenues au sein du fichier ne nécessitait pas la mise en place d'une hiérarchie particulière, hiérarchie qui aurait pu justifier ce choix de format. Le choix d'envoyer des fichiers au format texte brut, utilisant un symbole particulier pour séparer les données (le point-virgule : « ; »), a finalement été retenu.

La structure des lignes de ce fichier texte est la suivante :

```
### equimpentName;equipmentModel;equipmentCost;ordersReference;signalDescription;timestamp;
### TABLES :
### equipment    ;equipment     ;equipment    ;orders         ;signal           ;all_events;
### SOURCES :
### script       ;script        ;script       ;keyboard       ;script           ;script;
```

Figure 14: structure des lignes d'un fichier source

7.3 Base de données

Après la première étape de conception du modèle de données, directement issu de l'analyse des données collectées à la suite de la première visite de l'usine de SupremeX concernée par ce projet, le schéma suivant a été proposé :

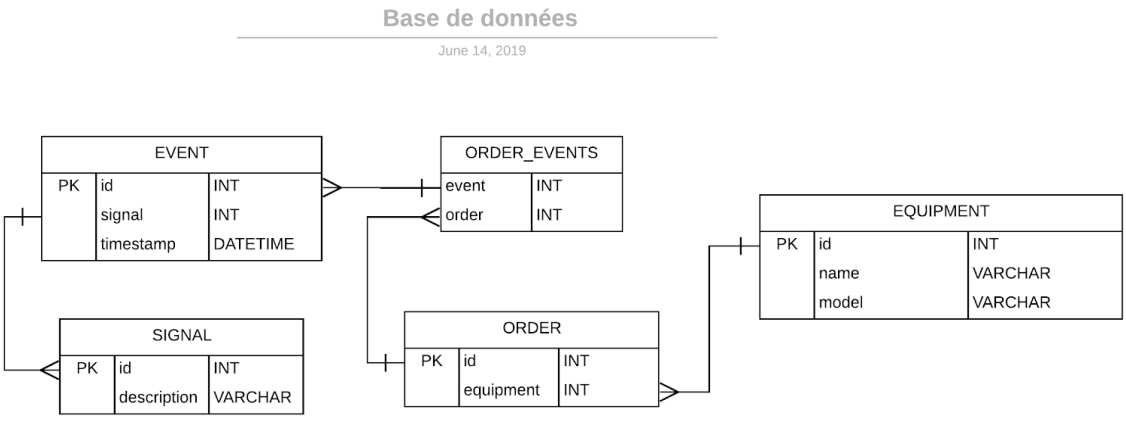


Figure 15: architecture initiale de la base de données

Les raisons du choix d'une telle conception du modèle de données seront détaillées dans les prochaines sections.

Premièrement, nous avons pris la décision de créer une table relativement simple modélisant les commandes des clients de l'entreprise SupremeX. En effet, la raison d'être du projet étant de mettre en place un système IoT permettant de suivre en temps réel l'état d'une machine de production réalisant des commandes faites par les opérateurs de l'entreprise SupremeX, la décision de regrouper au sein d'une table, centrale, ces différentes commandes suivies a semblé évidente.

Ensuite, nous avions au préalable (de cette étape de conception et de modélisation) d'ores et déjà identifié plusieurs types de signaux importants qui allaient se révéler utile dans notre mise en place du système final pour mener à bien ce projet. Ces différents types de signaux ont été identifiés lors de la

première visite de l'usine de Montréal le 28 Mai 2019, et ont été validés pendant la rencontre d'équipe avec les deux professeurs du projet le 31 Mai 2019.

Ces signaux sont donc actuellement au nombre de six. Ce nombre étant actuellement limité, mais n'étant pas nécessairement définitif, il a donc paru évident de créer une table pour les regrouper.

Ces signaux sont les suivants (la sémantique venant directement de la sémantique ayant été utilisée lors des premières propositions de squelettes XML mentionnés plus haut) :

- SETUP_START : signal indiquant que la préparation de la machine pour la commande à effectuer a débuté.
- SETUP_STOP : signal indiquant que la préparation de la machine pour la commande à effectuer est terminée.
- PRODUCTION_START : signal indiquant que la commande à effectuer a débuté.
- PRODUCTION_STOP : signal indiquant que la commande à effectuer est terminée.
- EMERGENCY_STOP : signal indiquant que le bouton d'arrêt d'urgence de la machine a été utilisé (ce qui signifie donc que la machine est à l'arrêt pour des raisons d'urgence).
- MAINTENANCE : signal indiquant que la machine est en maintenance.

Cette table des signaux se limitera donc à regrouper les divers types de signaux (leur nom ou leur description) ainsi que leur *id* au sein de la table, qui pourra être utilisée pour faire le lien avec les autres tables.

NB : attention à la sémantique utilisée dans ce rapport, car le terme signal est utilisé ici comme si l'on récupérait l'information dans un format analogique en s'étant raccordé électroniquement à la machine. Or ça n'est pas le cas. Ce signal pourrait en effet être de nature numérique et provenir des suites du traitement de l'information obtenu depuis une caméra puis traité par un microcontrôleur puis soumis à une base de données dans un format de fichier XML.

Ces signaux étant de plus nécessairement associés à des évènements se produisant lors de la production d'une commande, il a donc été implicite de créer une table générale des événements. Cette table des

événements sera utilisée pour regrouper tous les événements et faire le lien avec le signal associé à un évènement ainsi que la date à laquelle cet évènement a été déclenché (*timestamp*).

De fait l'adjonction d'une table d'association entre les commandes (*order*) et les événements s'est imposée comme étant le choix le plus simple pour relier signaux, événements et commandes.

Enfin, bien que dans le cadre de ce projet un seul équipement de la chaîne de production ne devrait être utilisé, il a semblé utile d'ajouter une table recensant les équipements sur lesquels les données pourraient être collectées, et ce, dans un but de facilitation de la capacité d'évolution du système final vers un système permettant de capturer les données provenant de plusieurs équipements distincts.

L'architecture de cette base de données a été modifiée de sorte à y ajouter une entrée relative aux coûts horaires des machines de production et des scripts utilitaires permettant le déploiement de cette base de données ont été mis en place lors des dernières phases du projet.

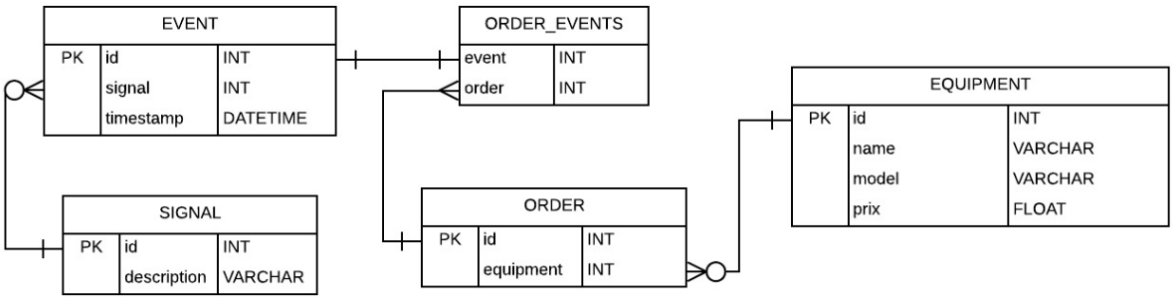


Figure 16: architecture finale de la base de données

7.4 Prototype

Un premier prototype a été mis en place. Ce prototype utilisait le langage Python, tel qu'initialement choisi, et dans sa version 2.7. Ce prototype a été réalisé sur une machine utilisant un système d'exploitation Unix et en arrangeant un extrait de code trouvé sur internet et répondant aux critères suivants :

- Langage Python;
- Système d'observation des événements d'un système de fichiers;
- Présence d'un module vide, mais néanmoins à configurer, permettant l'arrimage avec une base de données et de fait l'insertion dans une base de données.

Du fait du choix initial de mettre en place un micro service en Python, l'origine de ce prototype provient d'un article ayant été publié en 2007 et consultable uniquement sur le site internet Archive.org, permettant l'accès à des ressources n'étant plus en ligne depuis.

Le code issu de cet article a donc été modifié de sorte qu'un produit logiciel conforme aux exigences fonctionnelles du projet soit mis en place.

Afin de faciliter l'implémentation de ce prototype, l'outil Docker (plateforme de virtualisation par conteneur permettant de concevoir, tester et déployer des applications rapidement) a été utilisé pour mettre en place un conteneur au sein duquel la base de données a été implémentée directement dans un système d'exploitation CentOS, conteneur pouvant interagir avec le code en Python exécuté par le système d'exploitation de la machine hôte et permettant donc de valider le bon fonctionnement de l'ensemble côté serveur.

La base de données de ce prototype avait été mise en place en utilisant SQL SERVER EXPRESS, et l'utilisation de Docker a de plus permis de s'affranchir des tâches correspondant à l'installation complète d'un SGBD (système de gestion de base de données) et à sa configuration.

Ce prototype a donné des résultats satisfaisants et a conforté l'équipe dans ses choix quant à la conduite du projet, et à l'idée de porter ce code fonctionnel dans un environnement Unix, vers un environnement Microsoft Windows.

Malheureusement, le code de ce prototype utilise des fonctionnalités implémentées pour les systèmes Unix et non présentes au sein des systèmes utilisant Windows :

- os.fork(), une action permettant de créer des copies d'un processus à partir de lui-même;

- iNotify, une fonctionnalité caractéristique aux systèmes Unix et permettant d'associer des actions suite à des événements sur un système de fichiers.

C'est la raison pour laquelle ce prototype n'a finalement pas été modifié pour être adapté à un environnement Windows.

7.5 Choix technologiques définitifs

Afin de mettre en place un produit logiciel équivalent au prototype, et en raison du souhait de le déployer dans un environnement Windows, il a été choisi d'utiliser des fonctionnalités du cadriciel Microsoft .NET pour faciliter le développement. De plus le choix du langage de programmation PowerShell, dans sa version 4.1, s'est imposé comme étant l'un des meilleurs à disposition. L'envoi des fichiers de données depuis le Raspberry Pi vers le serveur Microsoft Windows de destination a lui aussi été revu pour créer un lien entre les deux machines via l'outil SSH, et en utilisant ce lien avec l'outil SCP de sorte à pouvoir automatiser des transferts de fichiers sécuriser.

L'application côté serveur est comportée d'un unique script au format PowerShell. Ce script est à la fois responsable de la surveillance des événements du système de fichiers, du traitement des données issues de fichiers et de l'insertion de ces données dans la base de données. En tout temps, et pour chacun des traitements réalisés, cette application est munie d'un système de log permettant de valider son bon fonctionnement et, au besoin, de déceler les erreurs de fonctionnement et d'en déterminer leur cause. Ce système de log est subdivisé en deux sous-systèmes distincts, l'un étant responsable du fonctionnement global de l'application, et l'autre étant dédié à la surveillance des interactions avec la base de données.

La surveillance des événements survenant sur le système de fichiers du serveur est utilisée pour permettre à l'application de savoir lorsque des fichiers sont déposés au sein d'un répertoire destiné à la réception. Lorsque cette réception est terminée, le traitement des données et l'insertion dans la base de données peuvent être effectués. Plusieurs observateurs d'événements ont été mis en place et sont associés à différents types d'événements.

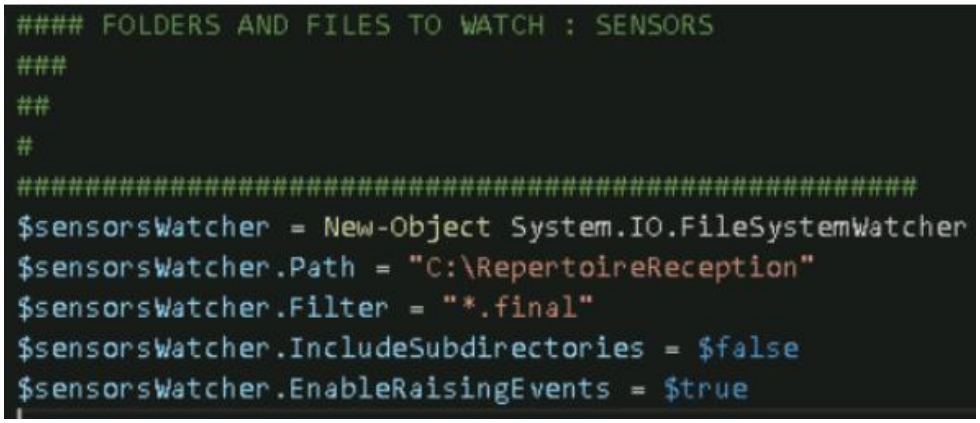


Figure 17: exemple de déclaration d'un observateur

Il existe trois répertoires étant surveillés par le script :

- Le répertoire C:\IOT\work\input il s'agit du répertoire de réception des fichiers. Les fichiers y sont déposés par le microcontrôleur avec une extension .part. Une fois le transfert terminé, le microcontrôleur remplace l'extension .part par .final ;
- Le répertoire C:\IOT\work\input\fileCam qui correspond au répertoire de travail pour les fichiers de données provenant de la caméra ;
- Le répertoire C:\IOT\work\input\fileSensors qui correspond au répertoire de travail pour les fichiers de données provenant des capteurs de signaux électriques.

Le premier observateur d'événements vérifie donc, en permanence, si des fichiers dont l'extension est .part ont été créés au sein du répertoire de réception. Lorsque le transfert de fichier est terminé et que l'extension .part est remplacée par .final, un autre observateur est chargé de déplacer le ficher vers le répertoire de travail correspondant au type de source associé au fichier. La différenciation du type de source est déterminée directement à partir du nom d'un fichier : si ce nom comporte fileCam, alors c'est un fichier provenant de la caméra, s'il contient fileSensors alors il provient des autres capteurs. Tous les fichiers ne respectant pas cette nomenclature ne sont pas pris en compte par le script.

Lors du déplacement vers les sous-répertoires de travail, deux opérations supplémentaires sont effectuées : premièrement, un horodatage au format de temps Unix est ajouté au début du nom du fichier (principalement pour permettre de différencier deux fichiers distincts provenant de la même source de

données), et à l'issue du déplacement, l'extension .final est remplacée par .moved, ce qui a pour effet de déclencher les derniers observateurs, structurellement identiques, mais associés à deux répertoires de travail distincts.

La dernière catégorie d'observateurs est responsable du déclenchement de l'analyse du contenu du fichier, de la création de variables intermédiaires et de leur insertion en base de données. L'analyse du contenu du fichier ainsi que l'insertion des données se font de manière séquentielle, ligne par ligne. L'utilisation d'expressions régulières a été utilisée afin d'omettre du traitement les lignes du fichier source n'étant pas conformes à la structure de ce qui est attendu par le script pour éviter de déclencher des erreurs de base de données. La séquence des opérations est la suivante :
Tant qu'il y a des lignes non vides dans le fichier :

1. La ligne courante est récupérée
2. Si la ligne correspond à l'expression régulière attendue :
    a. Le contenu de la ligne est séparé en plusieurs variables
    b. Les variables sont utilisées pour créer des requêtes
    c. Les requêtes sont envoyées à la base de données
3. Sinon, une ligne d'erreur est ajoutée au fichier de log des insertions en base de données de l'application

Finalement, de sorte que la portion serveur Windows du produit soit systématiquement en fonction sur le serveur, le planificateur de tâches du système d'exploitation du serveur a été utilisé pour construire une tâche étant toujours fonctionnelle, et responsable de l'exécution du script PowerShell en tout temps. De fait le serveur est en permanence prêt à recevoir des fichiers, à les interpréter, et à transférer leur contenu vers la base de données.

CHAPITRE 8

TABLEAU DE BORD

8.1 Rappel

SupremeX voulait avoir le temps de mise en route et le temps de production de ces machines afin de déterminer par un calcul le coût de production par machine, qui peut rapidement être erronée à cause d'erreurs humaines.
Ces informations sont illustrées dans un tableau de bord disponible pour l'exécutif de SupremeX à l'interne disponible à l'annexe IV.

8.2 Informations affichées

Le tableau de bord présente les informations suivantes :

- Le numéro de l'équipement ;
- Le numéro de la commande rentré au clavier par le technicien ;
- Une indication si la présence du technicien est détectée ;
- Le nombre de capteurs sur l'équipement ;
- Le temps de mise en route de l'équipement ou temps de setup, c'est-à-dire le nombre d'heures mis pour initialiser l'équipement pour la phase de production ;
- Le temps de production de l'équipement, c'est-à-dire le nombre d'heures où l'équipement a produit des enveloppes ;
- Le nombre d'enveloppes produites par l'équipement (nombre approximatif) ;
- Le coût de production en dollars canadiens par heure = (Temps de setup + Temps de production) * Coût fixe (= 95,97$) ;
- L'heure de la dernière mise à jour des données.

CONCLUSION

Dans le cadre de ce projet de fin d'études, nous avions pour objectif en tant qu'équipe de monter une preuve de concept pour l'entreprise SupremeX selon leur besoin exprimé. Nous devions entre autres exploiter les avantages d'un appareil de type IOT afin de faire la collecte de données des machines de production. Tout ça dans le but ultime d'éliminer l'incertitude opérationnelle causée par les manipulations humaines.

Nous sommes partis au début avec une solution avec des caméras comme capteurs principaux puis finalement ces derniers sont devenus secondaires au profit des capteurs sur les équipements, plus précis et plus facile à mettre en place.

La phase de conception est primordiale dans un projet de cette envergure et permet de multiples séances de remue-méninges qui permettront d'éviter les surprises en cours de développement.

Nous avons appris que ce sont les petits détails qui ralentissent le plus le projet comme les problèmes de connexion à la base de données pour un utilisateur externe au réseau ou les problèmes de sécurité liés à la communication de notre microcontrôleur avec la base de données.

À l'aide de nos capteurs sur équipement et nos caméras, nous avons réussi à atteindre nos objectifs et répondre aux attentes du client. Notre solution est en mesure de récupérer les impulsions électriques de la machine de production, détecter la présence d'un opérateur à l'aide d'une caméra, recevoir le numéro de la commande via un clavier numérique et détecter la présence d'enveloppe. Toutes ces données sont envoyées au Raspberry Pi qui le traite avec différents scripts. Ces mêmes données sont envoyées vers un serveur centralisé qui s'assure de les envoyer à la base de données. Pour finir, une interface a été développée pour pouvoir visualiser ces données récupérées. Le diagramme de contexte IoT de la solution finale de ce projet est disponible en annexe VI.

Il y a tout de même plusieurs pistes d'amélioration avant de pouvoir profiter pleinement de ce produit tel que l'optimisation des différents scripts au niveau du microcontrôleur, mais aussi des caméras et le temps d'insertion dans la base de données. Ce sont des défis à prioriser pour ajouter de la valeur à ce projet. Nous nous concentrerions sur les 3V, c'est-à-dire Volume, Vitesse et Variété des données. Nous avons un volume de données important provenant de différentes sources et à envoyer et traiter rapidement de multiples fois par heure.

Ce type de solution permettrait à SupremeX de développer une intelligence d'affaires vis-à-vis de ces équipements. Son implémentation permettrait de diminuer l'impact des erreurs humaines sur les rapports financiers de SupremeX et lui procurer une vision en temps réel de sa production.

En conclusion, le véritable défi des entreprises d'aujourd'hui est de trouver des moyens d'interpréter le flot de données disponibles provenant de multiples sources comme le dit Chris Murphy, le directeur du contenu infonuagique d'Oracle :

> *One of the myths about the Internet of Things is that companies have all the data they need, but their real challenge* ***is making sense of it.*** *In reality, the cost of collecting some kinds of data remains too high, the quality of the data isn't always good enough, and* ***it remains difficult to integrate multiple data sources****.*

RECOMMANDATIONS

Le système présenté dans le chapitre sur le serveur souffre d'un défaut de conception et n'est pas réellement un ETL au sens propre du terme pour la raison suivante : le traitement des données et l'insertion en base de données sont effectués l'un à la suite de l'autre pour chaque ligne conforme du fichier source. Cette conception a pour effet de ralentir considérablement le traitement des fichiers sources lorsque ces fichiers sont volumineux. Pour des fichiers sources de 1500 lignes, le temps de traitement a été mesuré et l'application met en moyenne 0,65 seconde pour traiter et insérer une ligne dans la base de données. Ce temps est satisfaisant pour une preuve de concept, mais ne semble pas adapté pour une utilisation industrielle en milieu professionnel. La première amélioration possible serait donc la mise en place d'une réelle séparation entre le traitement des données provenant des fichiers sources d'une part, et l'insertion de ces données dans la base de données d'autre part. Pour ce faire, au lieu de réaliser l'insertion séquentiellement pour chaque ligne, il serait préférable d'utiliser une variable, un tableau de longueur non fixe de chaines de caractères par exemple, afin d'y enregistrer les requêtes construites, dans un premier temps. Puis ce tableau pourrait être utilisé pour envoyer les requêtes par lots (*batch*). En procédant de la sorte, le traitement et l'insertion seraient totalement disjoints, ce qui aurait pour effet d'améliorer la performance de l'application.

Enfin, à la suite des tests fonctionnels réalisés lors de la validation du bon fonctionnement du système, l'équipe a remarqué que la rapidité des processeurs actuels pouvait amener un système informatisé à associer à deux événements normalement distincts, néanmoins très rapprochés dans le temps, le même horodatage (*timestamp*). Ce fonctionnement pourrait donc amener le système conçu et développé à ne pas être en mesure de différencier deux événements pourtant distincts du fait de leur horodatage similaire (même source de données, signal identique horodatage identique). Pour remédier à ce type d'événements, deux solutions sont envisageables. La première ne nécessite pas beaucoup d'effort de développement : il suffit d'améliorer la précision des *timestamp* associés aux événements, auquel cas il serait possible d'avoir à modifier le type de données associé au *timestamp* dans la base de données, mais aussi l'expression régulière utilisée pour valider le format des lignes d'un fichier source. La seconde, qui semble être plus complexe à mettre en place, mais qui semble néanmoins plus adaptée pour le développement d'un système industriel, nécessiterait une refonte de la base de données d'une part, et

d’autre part de plus lourdes modifications au niveau du code source. Cette seconde option concernerait l’ajout d’un identifiant associé aux événements (permettant ainsi l’existence d’événements distincts, mais cependant strictement similaires) dès lors de l’insertion en base de données. Finalement cette seconde méthode pourrait de plus encore être améliorée via l’adjonction d’une logique de remédiation au sein de la base de données, qui permettrait l’élimination des éventuels doublons (mêmes caractéristiques à l’exception de l’identifiant d’événement.

Par rapport au Raspberry, il faudrait considérer le développement d’une solution pour réduire la chaleur qui émane de l’équipement. Selon nos tests, le Raspberry surchauffait après quelques minutes d’utilisation. C’est un enjeu considérable à prendre en compte si l’on veut implanter ce genre de solution dans un entrepôt.

LISTE DE RÉFÉRENCES

Altassian (entreprise). Consulté en 2019. « Kanban - A brief introduction ». En ligne. < https://fr.atlassian.com/agile/kanban >.

Brian Benchoff. 2016. « The Story Of Kickstarting The OpenMV ». En ligne. < https://hackaday.com/2016/12/29/the-story-of-kickstarting-the-OpenMV/ >.

Contributeurs multiples (forum). Consulté en 2019. « How to monitor a folder and trigger a command-line action when a file is created or edited? ». En ligne. < https://superuser.com/questions/226828/how-to-monitor-a-folder-and-trigger-a-command-line-action-when-a-file-is-created >.

Contributeurs multiples (OpenMV, entreprise). 2019. « OpenMV/scripts/examples at master · OpenMV/OpenMV · GitHub ». En ligne. < https://github.com/OpenMV/OpenMV/tree/master/scripts/examples >.

Contributeurs multiples (forum). 2019. « OpenMV Cam M7 Text Recognition ». En ligne. < http://forums.OpenMV.io/viewtopic.php?t=1320 >.

Damien P. George, Paul Sokolovsky, OpenMV LLC, and contributors. 2019. « MicroPython Libraries - MicroPython 1.9.4 documentation ». En ligne. < https://docs.OpenMV.io/library/index.html#Python-standard-libraries-and-micro-libraries >.

Krishnakumar Rukmangathan. 2016. « Connecting to SQL Server Using PowerShell ». En ligne. < https://blogs.msdn.microsoft.com/dataaccesstechnologies/2016/11/30/connecting-to-sql-server-using-powershell/ >.

LeanKit® (entreprise). Consulté en 2019. « Kanban vs. Scrum: What are the differences? ». En ligne. < https://leankit.com/learn/kanban/kanban-vs-scrum/ >.

Teófilo Emídio de Campos. 2012. « The Chars74K dataset ». En ligne. < http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/ >.

Sander Marechal. Publication de 2007 mise à jour en 2009, consultation d'une sauvegarde d'un article tel qu'il était en 2013. « A simple unix/linux daemon in Python ». En ligne. < http://web.archive.org/web/20131017130434/http://www.jejik.com/articles/2007/02/a_simple_unix_linux_daemon_in_Python/ >.

ANNEXE I

TABLEAU DE COMPARAISONS DES RASPBERRY PI

Modèle du Raspberry	**Raspberry Pi Zero**	**Raspberry Pi Model B+**	**Raspberry Pi 2 Model B**	**Raspberry Pi 3 Model B+**
Spécifications				
Processeur	1GHz, single-core CPU	BCM2835	900MHz quad-core ARM Cortex-A7 CPU	Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
RAM	512MB	512MB	1GB	1GB
Connectivité	Bluetooth 4.1, WIFI, USB On-The-Go ports	4x USB, 40x GPIO, Ethernet port	4x USB, 40x GPIO, Ethernet port	4x USB, 40x GPIO, Ethernet port, WIFI, Bluetooth 4.2

ANNEXE II

TABLEAU DES HEURES CONSACRÉES AU PROJET

Mois	Semaine	Vanessa B.	Keyven-Steve L.	Quentin P.	Kévin D.	Total / semaine
Avril	S17	1,00	1,00	1,00	1,00	0,29
Mai	S18	0,00	0,00	0,00	0,00	0,00
	S19	1,50	3,00	1,50	2,25	8,25
	S20	2,75	2,75	5,47	6,25	17,22
	S21	4,20	3,70	5,16	2,20	15,26
	S22	6,00	6,50	10,02	4,00	26,52
Juin	S23	3,50	2,50	3,28	3,50	12,78
	S24	4,00	11,00	7,50	9,00	31,50
	S25	4,50	9,50	7,00	4,50	25,50
	S26	3,00	1,00	2,00	4,00	10,00
Juillet	S27	6,50	0,00	0,00	5,00	11,50
	S28	4,00	1,50	1,50	3,00	10,00
	S29	4,50	10,00	7,45	2,50	24,45
	S30	13,75	6,75	5,67	18,50	44,67
Août	S31	3,75	2,75	2,50	1,00	10,00
	S32	10,00	4,00	10,50	5,00	29,50
	S33	0,00	0,00	0,00	0,00	0,00
	S34	0,00	0,00	0,00	0,00	0,00
	S35	0,00	0,00	0,00	0,00	0,00

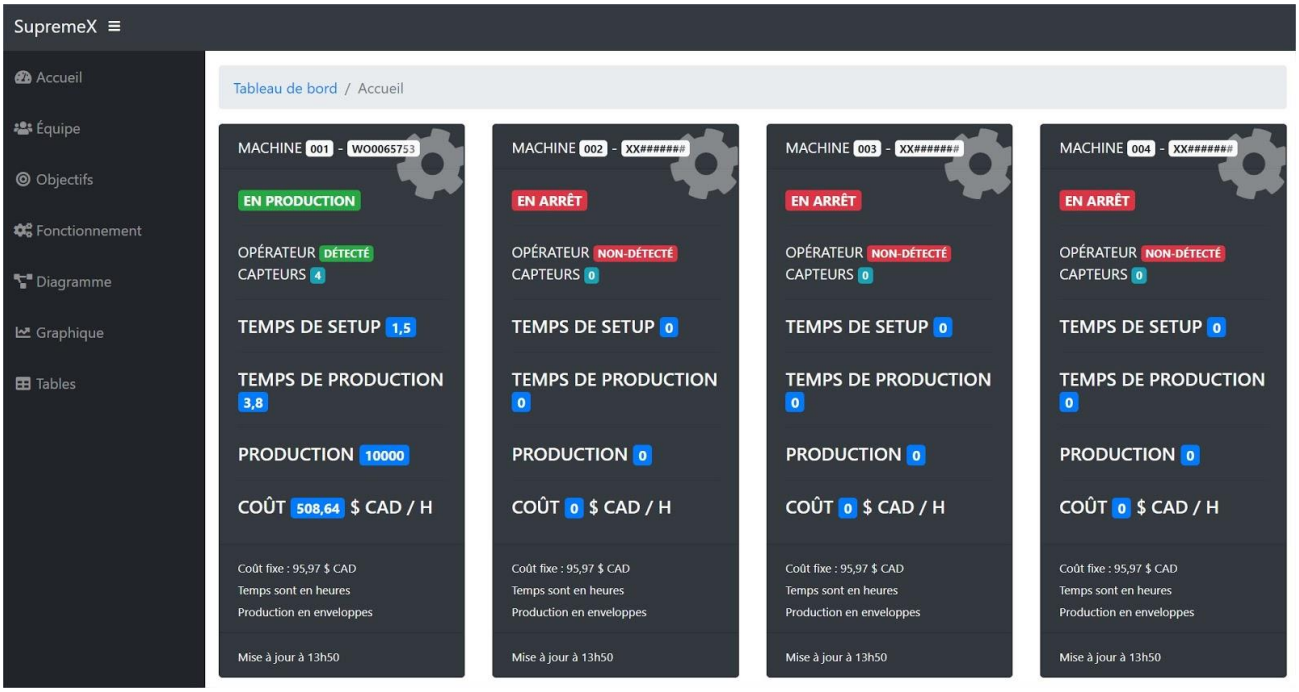
ANNEXE III

TABLEAU DE COMPARAISONS DES SPÉCIFICATIONS DES CAMÉRAS OPENMV

Modèle de caméras	**M4 V1**	**M4 V2**	**M7**	**H7**
Spécifications				
Processeur	ARM® 32-bit Cortex®-M4 CPU w/ Single Precision FPU 180 MHz (225 DMIPS) Core Mark Score: 608 (compare w/ Raspberry Pi Zero: 2060)	ARM® 32-bit Cortex®-M4 CPU w/ Single Precision FPU 180 MHz (225 DMIPS) Core Mark Score: 608 (compare w/ Raspberry Pi Zero: 2060)	ARM® 32-bit Cortex®-M7 CPU w/ Double Precision FPU 216 MHz (462 DMIPS) Core Mark Score: 1082 (compare w/ Raspberry Pi Zero: 2060)	ARM® 32-bit Cortex®-M7 CPU w/ Double Precision FPU 480 MHz (1027 DMIPS) Core Mark Score: 2400 (compare w/ Raspberry Pi 2: 2340)
RAM	64KB .DATA/.BSS/Heap/Stack 192KB Frame Buffer/Stack (256KB Total)	64KB .DATA/.BSS/Heap/Stack 192KB Frame Buffer/Stack (256KB Total)	128KB .DATA/.BSS/Heap/Stack 384KB Frame Buffer/Stack (512KB Total)	256KB .DATA/.BSS/Heap/Stack 512KB Frame Buffer/Stack 256 KB DMA Buffers (1MB Total)
Flash	16KB Bootloader 48KB Embedded Flash Drive 960KB Firmware (1MB Total)	16KB Bootloader 48KB Embedded Flash Drive 960KB Firmware (1MB Total)	32KB Bootloader 96KB Embedded Flash Drive 1920KB Firmware (2MB Total)	128KB Bootloader 128KB Embedded Flash Drive 1792KB Firmware (2MB Total)
Mémoire disponible pour bibliothèque (selon test)		24KB		
Supported Image Formats	Grayscale RGB565 JPEG	Grayscale RGB565 JPEG	Grayscale RGB565 JPEG (and BAYER)	Grayscale RGB565 JPEG (and BAYER)
Maximum Supported Resolutions	Grayscale: 320x240 and under RGB565: 320x240 and under Grayscale JPEG: 320x240 and under RGB565 JPEG: 320x240 and under	Grayscale: 320x240 and under RGB565: 320x240 and under Grayscale JPEG: 320x240 and under RGB565 JPEG: 320x240 and under	Grayscale: 640x480 and under RGB565: 320x240 and under Grayscale JPEG: 640x480 and under RGB565 JPEG: 640x480 and under	Grayscale: 640x480 and under RGB565: 320x240 and under Grayscale JPEG: 640x480 and under RGB565 JPEG: 640x480 and under
Lens Info	Focal Length: 2.8mm Aperture: F2.0 Format: 1/3" HFOV = 70.8°, VFOV = 55.6° Mount: M12*0.5 IR Cut Filter: 650nm (removable)	Focal Length: 2.8mm Aperture: F2.0 Format: 1/3" HFOV = 70.8°, VFOV = 55.6° Mount: M12*0.5 IR Cut Filter: 650nm (removable)	Focal Length: 2.8mm Aperture: F2.0 Format: 1/3" HFOV = 70.8°, VFOV = 55.6° Mount: M12*0.5 IR Cut Filter: 650nm (removable)	Focal Length: 2.8mm Aperture: F2.0 Format: 1/3" HFOV = 70.8°, VFOV = 55.6° Mount: M12*0.5 IR Cut Filter: 650nm (removable)

ANNEXE IV

TABLEAU DE BORD



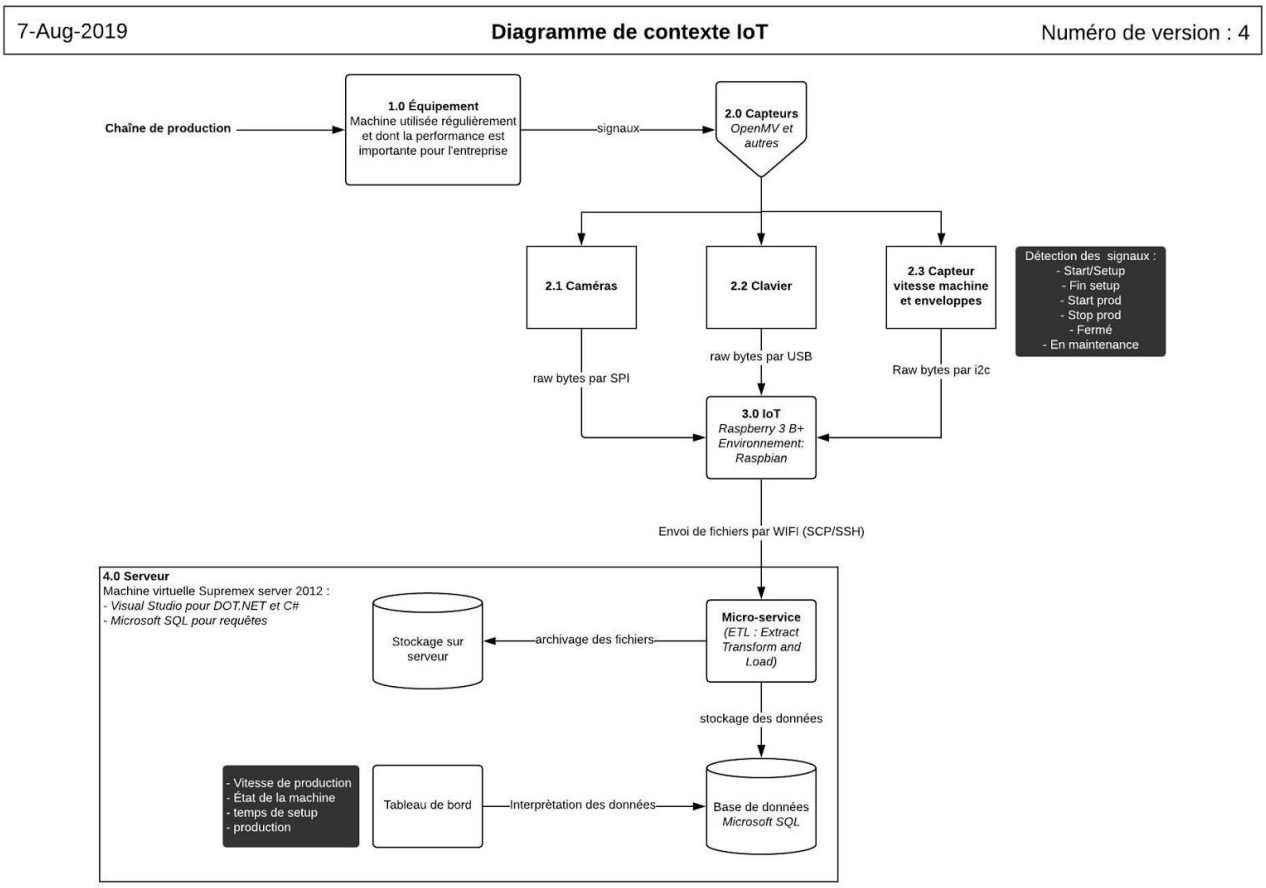
ANNEXE V

TABLEAU DE COMPARAISONS DES FONCTIONNALITÉS DES CAMÉRAS OPENMV

	M4 V1	M4 V2	M7	H7
Frame Differencing				
Color Tracking				
Marker Tracking				
Face Detection				
Eye Tracking				
Optical Flow				
QR Code Detection/Decoding				
Data Matrix Detection/Decoding				
Linear Barcode Decoding				
AprilTag Tracking				
Line Detection				
Circle Detection				
Rectangle Detection				
Template Matching				
Image Capture				
Video Recording				

ANNEXE VI

DIAGRAMME DE CONTEXTE IOT



ANNEXE VII

CONTRAT D'ÉQUIPE

Nom de l'équipe : SupremeX - 022

Coordonnées des membres de l'équipe

Prénom, NOM	Vanessa, BAQUERO
Prénom, NOM	Quentin, PATAULT
Prénom, NOM	Keyven-Steeve, LEGAGNEUR
Prénom, NOM	Kévin, DELORME

Vos exigences personnelles pour bien travailler en équipe

Prénom	**Exigences personnelles**
Keyven-Steeve Legagneur	Participation active au projet et remettre les livrables qui ont préalablement été acceptés dans les rencontres avec l'équipe
Vanessa Baquero	Participation active au projet et remettre les livrables qui ont préalablement été acceptés dans les rencontres avec l'équipe
Quentin Patault	Suivre les étapes du projet peu importe ses tâches en faisant preuve de disponibilité et de compréhension vis-à-vis des autres membres de l'équipe.
Kévin Delorme	Faire preuve d'engagement pour atteindre les objectifs fixés dans le cadre du projet. Participation régulière au projet.

Les objectifs de l'équipe :

- Travailler régulièrement pour atteindre les objectifs du projet.
- Communiquer afin de travailler en équipe et pouvoir s'entraider.
- Apporter une solution pour les besoins du client.
- Rester réaliste dans les délais et les exigences du projet.

Les règles de conduite de l'équipe :

- Être disponible sur Slack selon un délai acceptable.
- Mettre à jour le tableau Trello.
- Demander de l'aide lorsqu'on est coincé devant un problème.
- Faire preuve de compréhension et aider les membres de l'équipe devant un problème.
- Être présent au plus de rencontres d'équipe que possible, que ce soit au téléphone ou en personne.
- Participer activement aux réunions d'équipe et de suivi.
- S'assurer de l'avancement et de la complétion de ces tâches.
- Justifier lorsqu'on ne peut pas être présent à une réunion d'équipe ou de suivi.

Conséquences du non-respect des règles :

Avis verbaux au membre concerné.

Avis écrits aux professeurs et rencontre avec le membre concerné.

Signatures des membres de l'équipe :

Quentin Patault
Vanessa Baquero
Keyven-Steeve Legagneur
Kevin Delorme**DEDeDeloDelorme**