Développement d¿un framework modulaire intégrant les nouvelles technologies des automates programmables et application à une ligne de tri robotisée

Auteur : Baudinet, Robert
Promoteur(s) : Bruls, Olivier
Faculté : Faculté des Sciences appliquées
Diplôme : Master en ingénieur civil mécanicien, à finalité spécialisée en génie mécanique
Année académique : 2017-2018
URI/URL : http://hdl.handle.net/2268.2/5057

Université de Liège

Faculté des Sciences Appliquées

Master en Ingénieur Civil

Développement d'un framework modulaire intégrant les nouvelles technologies des automates programmables et application à une ligne de tri robotisée

Travail de fin d'études réalisé en vue de l'obtention du grade de master "Ingénieur Civil en Mécanique" par Robert Baudinet

Auteur
Robert Baudinet

Promoteur académique
Pr. Olivier Brüls

Promoteur industriel
Sébastien Themlin

Année Académique 2017-2018

Abstract

Les techniques de programmation des automates industriels sont très éloignées des techniques de programmation des ordinateurs modernes. Historiquement, cet écart se justifiait par la différence de performance, par la transmission d'habitudes désuètes et par l'utilisation de langages propriétaires. Pourtant, les ordinateurs industriels modernes et le développement de nouvelles normes offrent des opportunités inédites dans les méthodes de développement des applications.

L'objectif de ce travail de fin d'études est d'explorer les nouvelles fonctionnalités de la norme CEI 61131-3 pour harmoniser le développement des applications d'automates programmables. Un framework a donc été créé sur base des points communs entre différents projets d'automate tout en implémentant un paradigme orienté-objet pour lui assurer une bonne modularité. Comme le développement d'une interface homme-machine est un besoin souvent lié à la programmation des automates industriels, un HMI indépendant de la marque de l'automate a aussi été développé afin d'être réutilisable à chaque projet.

Ce framework a ensuite été appliqué au cas concret d'une chaîne de tri robotisée. Cette ligne, développée dans le cadre du projet Pick-it, a pour objectif de trier les déchets métalliques. Après avoir analysé les besoins et les éléments qui forment la ligne, la solution adaptée a finalement été intégrée à la chaîne. Le contrôle de certaines machines et le déroulement de certaines séquences ont ainsi pu être testés avec succès bien qu'un travail d'intégration plus conséquent que prévu ne soit nécessaire pour permettre l'automatisation complète de la ligne.

En conclusion, l'ensemble des éléments développés sont passés en revue et des pistes d'améliorations sont suggérées. Les résultats obtenus laissent penser que le développement de ces technologies devrait continuer à l'avenir.

Remerciements

À l'issue de la rédaction de ce travail de fin d'études, je voudrais tout d'abord remercier Fabien Defays et Grégory Reichling pour l'opportunité qui m'a été donnée de découvrir le monde de l'ingénierie en entreprise chez Citius Engineering. Plus particulièrement, j'aimerais remercier Sébastien Themlin pour ses précieux conseils et sa supervision ainsi que Thomas, Thierry et Vincent pour leur aide technique.

Mes remerciements vont également à l'équipe du Laboratoire GeMMe. Je tiens à remercier Godefroid pour sa confiance et son accueil ainsi que Pierre et François pour leur aide et leur collaboration.

Je souhaiterais aussi exprimer ma gratitude envers le Professeur Olivier Brüls pour son accompagnement et sa disponibilité en tant que superviseur.

Enfin, je remercie ma famille et mes proches pour le soutien qu'ils m'ont apporté jusqu'à l'aboutissement de ce projet.

Je suis donc reconnaissant envers toutes ces personnes pour cette expérience positive qui m'a permis de confirmer mon choix de me diriger dans l'ingénierie mécatronique et robotique.

Table des matières

Liste des tableaux

Table des figures

Chapitre 1

Introduction

Les systèmes de contrôle automatisés ont toujours eu une place prédominante dans l'industrie.

Au début du 20ème siècle, des relais électromagnétiques étaient ainsi intégrés à des circuits électriques dans le but de pouvoir faire basculer un système d'un état à un autre. Ces systèmes ont été communément utilisés jusqu'à la fin des années soixante lorsqu'une interface virtuelle, un automate programmable, a permis de ne plus devoir recâbler systématiquement toute la logique du circuit. À cette époque, comme les automaticiens n'avaient pas de connaissance en informatique, la configuration de ces automates s'effectuait de manière graphique en implémentant virtuellement les circuits électriques à relais.

Quelques dizaines d'années plus tard, force est de constater que les méthodes de programmation des automates n'ont pas évolué à la même vitesse que leur capacité de calcul. La différence de formation entre un automaticien et un informaticien a empêché la transmission des connaissances d'un milieu à l'autre. À l'heure actuelle, pour un grand nombre de techniciens, la programmation d'une séquence automatisée passe encore par cette implémentation graphique des circuits électriques, appelée langage ladder. Bien que d'autres

langages aient été développés pour palier aux limitations fonctionnelles de ce langage graphique, les méthodes de programmation restent très éloignées de celles de l'informatique moderne, car apparaissent souvent comme étant trop abstraites pour un automaticien.

Pourtant, une nouvelle génération d'automates programmables a vu le jour et a visé à réduire les différences qui existaient entre les automates et les ordinateurs. Ces PC industriels ont été accompagnés de nouvelles normes permettant d'harmoniser le développement des applications automates et l'utilisation de nouveaux paradigmes de programmation. Au vu des besoins de l'industrie 4.0., il est dès lors à prévoir que les anciennes méthodes de programmation ne suffisent plus pour répondre à la modularité nécessaire aux nouveaux projets.

L'objectif de ce travail de fin d'études est d'explorer les solutions offertes par les nouvelles technologies des automates comme la norme CEI 61131-3 afin de développer un paradigme de programmation systématique et modulaire. L'idée est ainsi de rassembler les points communs qui existent entre les projets pour harmoniser les méthodes de développement. Parallèlement, le développement d'une solution d'interface homme-machine flexible devra être étudié pour affranchir les développements de la marque d'automate utilisée.

Ensuite, les technologies développées seront appliquées à l'automatisation d'une chaîne de tri robotisée. Cette chaîne de tri est un prototype développé conjointement par Citius Engineering, le GeMMe et Comet Traitements et s'inscrit dans le cadre du projet Reverse Metallurgy qui vise à revaloriser les déchets métalliques. Afin d'adapter les solutions de manière adéquate, le besoin du client devra donc être caractérisé et le rôle de chaque élément de la ligne devra être explicité. Lorsque l'adaptation sera terminée, l'intégration de la solution sera testée et elle sera confrontée aux réalités du terrain.

Lors des étapes clefs de ce TFE, la rigueur industrielle et la curiosité scientifique seront transmises par une équipe mixte d'experts industriels de chez Citius Engineering et de chercheurs acédémiques.

Chapitre 2

Projet Reverse Metallurgy

2.1 Contexte

Le projet Reverse Metallurgy est un projet porté conjointement par la région Wallonne et des acteurs industriels et académiques wallons afin de favoriser l'économie circulaire. L'idée est en effet de développer des technologies dont le but est de revaloriser les déchets métalliques arrivés en fin de vie. Les métaux sont particulièrement intéressants, car, contrairement aux polymères, ils ont la capacité d'être réutilisables plusieurs fois tout en conservant leurs propriétés. Les technologies ainsi développées permettraient d'une part de stabiliser les sources de matières premières de la métallurgie wallonne et diminuer son empreinte écologique. D'autre part, elles permettraient également de développer de nouveaux domaines d'expertises pour les entreprises qui participent au projet. Alors que l'industrie liégeoise s'était spécialisée dans la transformation du minerai en produit fini, cette revalorisation de déchets pour remplacer des minerais coûteux peut s'assimiler à de la métallurgie *à l'envers* ou *Reverse Metallurgy*.

FIGURE 2.1 – Logo de la S.C.R.L. Reverse Metallurgy.

2.1.1 Organisation[21]

Le projet a été lancé en 2015 sous l'impulsion du Ministre de l'Économie et de l'Enseignement supérieur, Jean-Claude Marcourt, et le Ministre de la Recherche et du Développement durable, Jean-Marc Nollet. Le gouvernement wallon a décidé de financer le projet à hauteur de 41.5 millions d'euros dans le cadre de la politique industrielle wallonne "NEXT Économie circulaire". De plus, le projet RM bénéficie d'autres supports financiers et organisationnels notamment de la part de Pôle MécaTech, pôle de compétitivité wallon en génie mécanique, le GRE-Liège, ensemblier pour la création d'emplois durables en province de Liège et de B.E. Fin, une filiale de la Société Régionale d'Investissement de Wallonie.

Afin de profiter d'une part de la créativité offerte par le monde de la recherche scientifique et d'autre part de la rigueur industrielle, le consortium suivant a été bâti, combinant acteurs à expertise industrielle et acteurs à expertise académique et de recherche.

Acteurs à expertise industrielle :

- *Citius Engineering S.A.*, entreprise liégeoise spécialisée dans la conception et la réalisation de machines sur-mesure ainsi que dans la consultance pour tout le secteur industriel,
- *Comet Traitements*, entreprise active dans le traitement, la valorisation et le recyclage des déchets métalliques arrivés en fin de vie,
- *Carmeuse*, groupe expérimenté dans l'extraction et transformation de la chaux,

du calcaire et produits dérivés,

— *Hydroeral*, société de recyclage de métaux non ferreux par traitements hydrométallurgiques,

— *Magotteaux*, entreprise à expertise dans le recyclage et le dragage par optimisation de procédés à haute abrasion et concasseurs, pour les secteurs des mines, du ciment et des carrières,

— *Marichal Ketin*, fonderie se chargeant de la conception, la fabrication et la commercialisation de cylindres pour le laminage de métaux ferreux.

Acteurs à expertise académique et de recherche :

— *Université de Liège - Laboratoire GeMMe*, laboratoire de Génie Minéral, Matériaux et Environnement, développant des procédés innovants pour le traitement des minéraux et des métaux,

— *CRM Group*, centre de recherche axé sur la production et la transformation des métaux ainsi que le coating métallique

— *CTP - Centre Terre et Piège*, centre de Recherches Agréé spécialisé dans le traitement de minerais.

FIGURE 2.2 – Logos des partenaires tels que présentés sur le site du projet Reverse Metallurgy.

2.2 Axes principaux

Le projet RM a été subdivisé en 4 projets, ou axes, dont les spécificités sont décrites dans les sections ci-dessous.

2.2.1 Axe Pick-it

Le consortium Pick-It regroupe le laboratoire du GeMMe (Université de Liège), Citius Engineering et Comet Traitements qui agissent en coordinateurs du projet. Cet axe concerne l'identification et le tri des déchets métalliques. Le développement d'une solution automatisée est nécessaire pour réaliser ces tâches, l'utilisation d'êtres humains n'étant en effet pas envisageable au vu du coût et des limites de performances du personnel.

Les aboutissants de ce projet sont :

— Pour le GeMMe, d'offrir un service de caractérisation des matériaux et de configuration de cellules de tri,

— Pour Comet Traitements, d'accentuer leur expertise en tri de métaux non-ferreux,

— Pour Citius Engineering, de faire valoir leur expérience en développement de chaîne de tri.

FIGURE 2.3 – Déchets métalliques en cours de tri sur la bande convoyeuse du prototype du projet Pick-It.

Après avoir obtenu des résultats convaincants à l'aide du prototype de tri installé au GeMMe, le consortium a décidé de poursuivre le projet et développer un démonstrateur industriel. La majorité du travail actuel porte ainsi sur l'amélioration du prototype ainsi que l'adaptation des technologies existantes au démonstrateur industriel.

2.2.2 Axe Biolix

Le consortium du projet Biolix est dirigé par Comet Traitements en collaboration avec le GeMMe. Cet axe a pour but le développement d'une cellule de démonstration de recyclage par hydrométallurgie. Les deux collaborateurs développent depuis 2011 un procédé capable de dissoudre sélectivement le cuivre contenu dans certains déchets. L'idée étant ensuite de solidifier les atomes de cuivre dissous sur des cathodes de manière à produire jusqu'à 750 tonnes de cuivre par an.

Le projet s'effectue en deux temps avec la validation technologique puis le développement à grande échelle. Une unité de pilote a donc d'abord été construite chez Comet Traitements à Mons et s'est soldée par la création de plus de 14 emplois. Cette unité a permis de quantifier le potentiel de l'hydrométallurgie ainsi que définir les matériaux à revaloriser via cette méthode.

2.2.3 Axe Plasmetric

Le consortium du projet Plasmetric est composé d'Hydrométal, de Carmeuse, du CRM et du GeMMe. Cet axe a pour objectif de développer un four à plasma capable d'extraire certains métaux non-ferreux des déchets métalliques. À cet effet, une cellule pilote a donc été installée chez Hydrométal. Cette unité est constituée d'un four, d'équipements de traitements des gaz ainsi que du matériel pour la préparation et concentration des métaux. La charge de travail liée au lavage des gaz est assurée par Carmeuse tandis que l'Université de Liège apporte son expertise au niveau de la caractérisation des matériaux.

FIGURE 2.4 – Générateur d'hydrométallurgie développé dans le cadre du projet Biolix.

2.2.4 Axe Pyrométallurgie

Cet axe s'oriente principalement sur les différentes méthodes d'obtention et de récupération des déchets. L'objectif est d'élaborer différents circuits de récupération pour valoriser au mieux les déchets issus de fonderies, les déchets industriels ou même les déchets ménagers pour éviter leur mise en décharge dans des centres d'enfouissement technique.

2.3 Focus sur l'axe Pick-It

Ce travail de fin d'études a été réalisé dans le cadre d'un stage en entreprise chez Citius Engineering et porte sur l'axe Pick-it. Au vu de la taille et des multiples orientations de ce projet, il convient donc de définir clairement le cadre dans lequel s'inscrit ce travail. L'objet de cette section est donc de préciser plus en détail le rôle des différents intervenants puis de spécifier les éléments traités et présentés tout au long de ce rapport.

Bien que chaque membre du consortium possède ses propres besoins, le travail à fournir peut être clairement exprimé en considérant uniquement ceux de Comet Traitements puisque celui-ci peut être identifié comme le client de Citius d'un point de vue macrosco-

pique. Cette abstraction permet de réaliser une analyse BOSCAR du projet. Bien que les informations présentes dans une telle analyse puissent être redondantes avec les explications données auparavant, l'intérêt du BOSCAR est justement d'être autosuffisant pour permettre la compréhension du projet dans son ensemble. Au vu de la longueur de cette analyse, il a été décidé de la présenter à l'annexe 8.

2.3.1 Flux de matière souhaité par Comet Traitement

Cette section porte sur la description du parcours attendu d'une pièce métallique depuis le broyage jusqu'à sa dernière étape de tri. Cette description correspond à une intégration de la solution développée dans l'actuel état de l'art.

La Figure 2.1 est un organigramme qui schématise les différents traitements que subit le *Zorba*, mélanges de déchets métalliques issus de plusieurs industries.

Au début du processus de tri, le Zorba est tamisé de manière à ne garder que les pièces ayant des dimensions comprises entre 20 et 100 mm. Ensuite, une séparation magnétique permet d'extraire les métaux ferreux de l'ensemble. Après, une séparation à l'aide du principe des courants de Foucault permet de différencier les déchets métalliques des non-métaux qui sont évacués de la chaîne. À ce stade, les déchets métalliques sont principalement composés d'Aluminium et sont différenciés des PLICZ (i.e. Plomb Laiton Inox Cuivre Zinc et Exotique) à l'aide d'une estimation de la masse volumique permise par des rayons X puis d'une éjection par des soufflettes.

C'est lorsqu'il ne reste principalement que des membres de la famille des PLICZ que le dispositif Pick-It intervient pour l'identification et le tri. Le dispositif doit idéalement permettre de séparer chaque famille mais également d'identifier les nuances existantes au sein de ces familles. Puisque le débit de déchets est estimé à 0,8 pièce/seconde et qu'une recirculation à hauteur de 1,4 pièces/seconde est autorisée, l'ensemble des robots doit avoir une capacité de tri de 4,9 déchets/seconde.

Puisque le tri à effectuer par la chaîne Pick-it doit intervenir à la suite d'une série de traitements, il est intéressant d'avoir conscience de ces procédés antérieurs qui font partie du contexte du projet. Cela peut ainsi permettre de comprendre l'origine et anticiper certains problèmes. De cette manière, il est par exemple primordial de prévoir le comportement à adopter si une pièce a des dimensions trop grandes ou trop petites malgré le passage par les cribles.

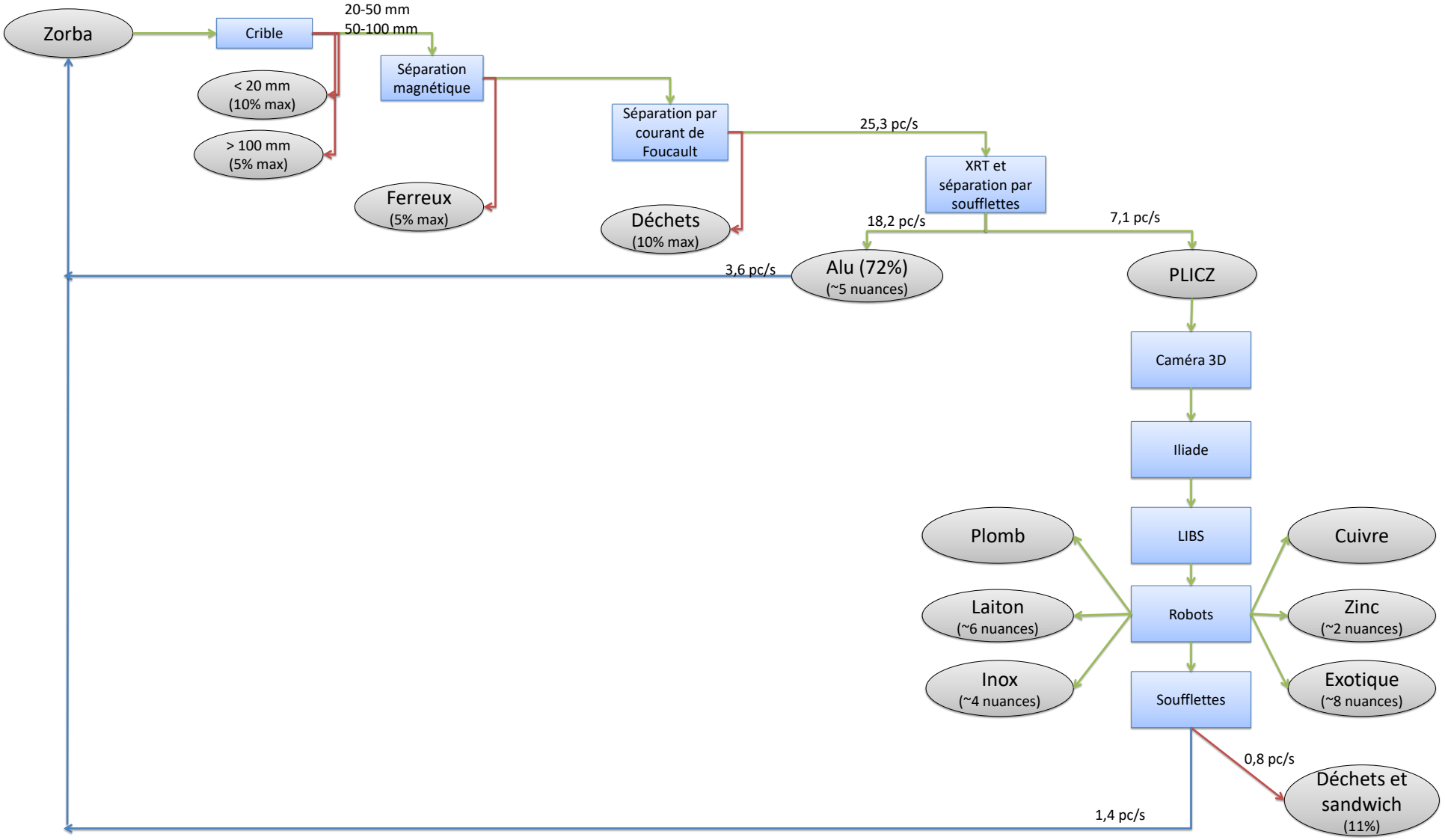


Tableau 2.1 – Flux de matières

2.3.2 Rôle de Citius

Comme expliqué auparavant, le rôle de Citius Engineering est d'apporter son expérience industrielle pour aider à la conception du prototype et du démonstrateur. Puisque les ingénieurs de Citius sont spécialisés dans le développement de machines automatiques sur mesure, les postes suivants leurs ont étés attribués :

- Réalisation des plans électriques et installation,
- Élaboration de la disposition de la ligne, depuis l'alimentation de matières PLICZ jusqu'aux *Big Bags* contenants chaque nuance,
- Développement de l'architecture et du code de l'automate programmable,
- Programmation des robots.

Il est à noter que, comme il s'agit du domaine d'expertise du GeMMe, l'équipe de Citius n'a pas directement travaillé sur les recherches visant à l'identification des nuances à trier.

2.3.3 Contribution personnelle

Lorsque l'objet du stage a été discuté pour la première fois, en mars 2017, le travail à effectuer portait sur le développement de l'architecture et du code de l'automate programmable du démonstrateur. Durant cette rencontre, une réserve avait toutefois été émise quant à l'état d'avancement incertain du projet l'année suivante. Cette prudence s'est avérée justifiée puisque, au commencement du stage en janvier 2018, une dissonance concernant la méthode de convoyage a engendré un retard sur le planning et le développement du démonstrateur. Afin de permettre le développement du code de l'automate tout en conservant un aspect pratique, une réunion entre Godefroid Dislaire, responsable du développement du prototype au GeMMe, Sébastien Themlin, le superviseur industriel de ce TFE et le professeur Olivier Brüls, a servi à réorienter l'objet du stage.

Il a ainsi été décidé que le travail fourni devrait être applicable pour automatiser le prototype du GeMMe tout en restant suffisamment modulaire pour pouvoir être utilisable

avec le démonstrateur moyennant le moins possible de modifications. Après consultation des coutumes industrielles, il est apparu que les technologies couramment utilisées se prêtaient assez mal à la flexibilité requise. Or, les opportunités offertes par les nouvelles technologies liées aux automates permettent justement une plus grande modularité à condition d'un changement de paradigme de programmation. C'est ainsi qu'est née l'idée du développement d'un framework réutilisable entre différents projets d'automates et capable de permettre d'une part un gain de temps et d'autre part de gagner en sécurité au travers la rigueur du framework. Il est également à noter que, puisque l'expérience des automaticiens de Citius Engineering provient de différentes entreprises, la nécessité d'harmoniser les développements des programmes PLC s'était déjà faite ressentir et ce travail s'inscrit donc également dans ce cadre.

Chapitre 3

Prototype du GeMMe

Le présent chapitre porte sur le prototype développé dans le laboratoire du GeMMe, sous la supervision de Godefroid Dislaire.

Comme expliqué préalablement au chapitre 2, ce prototype a été développé afin de valider les solutions techniques en prévision du démonstrateur industriel. Fin de l'année 2016, il était déjà capable d'identifier cinq différentes classes de matériaux et de les trier à l'aide de robots et de soufflettes[13]. Depuis cette présentation, la solution technique du tri par soufflerie n'y est plus expérimentée au même titre que le tri par glissage, mais de nouvelles technologies comme le lancer de déchets[19] y sont étudiées.

3.1 Description

Le prototype mesure environ 10 m de long pour 2m de large. Dans sa version actuelle, les éléments principaux qui le composent sont une alimentation vibrante, un convoyeur, une série de capteurs pour la caractérisation puis des robots qui effectuent le tri. De plus, des alimentations et armoires électriques sont disposées en dehors du chemin du convoyeur pour assurer le bon fonctionnement des éléments principaux. Le rôle de chaque composant

est plus exhaustivement décrit à la section 3.2.

La Figure 3.1 présente la chaîne pour donner une idée de la disposition réelle des différents éléments en fonction de leur fonctionnalité.

FIGURE 3.1 – Disposition du prototype Pick-It. (a) Éléments de caractérisation (b) Éléments de contrôle (c) Éléments de tri.

Le flux de matière au travers du prototype est décrit par l'organigramme de la Figure 3.2. Ce flux de matière est légèrement différent de celui présenté au chapitre 2 qui est la version finale attendue par Comet Traitements. Dans la version actuelle du prototype, les pièces sont ainsi d'abord dispersées par l'alimentation vibrante sur la largeur de la bande convoyeuse. Les déchets sont ainsi acheminés au travers de la caméra 3D qui reconstruit la forme de la pièce qui servira à estimer la masse et à calculer les points de préhension. Les déchets passent ensuite devant la caméra spectrale qui capte le spectre, invisible à l'oeil nu, puis au travers du système à rayons X et enfin au travers du système laser de type LIBS. L'ensemble des données recueillies par les capteurs permet de classifier, suite à un apprentissage statistique préalable, la famille et la nuance à laquelle le déchet appartient. Les robots vont alors prendre puis déposer les déchets dans une des goulottes, également appelées chutes, selon la classe trouvée.

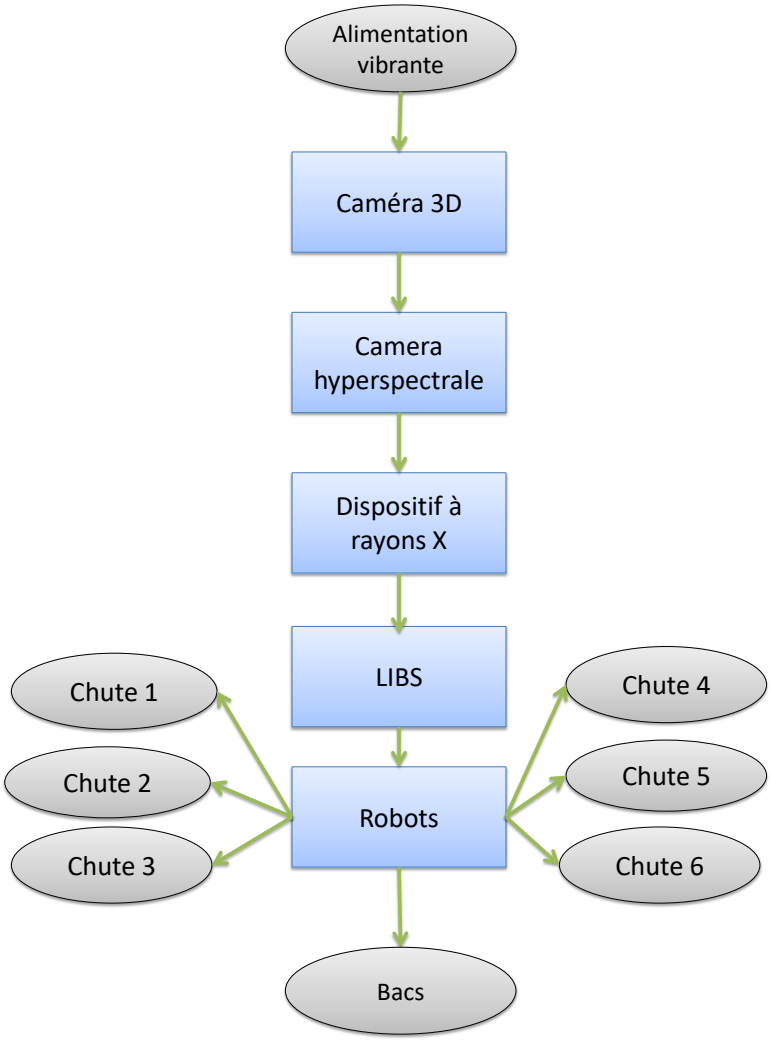


FIGURE 3.2 – Flux de matière du prototype Pick-It.

3.2 Architecture matérielle

La description exhaustive du matérielle est d'une part utile à la compréhension de la ligne mais aussi d'autre part indispensable pour assurer le contrôle de chaque élément. En prévision de cette automatisation, des fiches matérielles ont été rédigées et contiennent des informations comme les voies de communication possibles ainsi que le comportement à adopter à chaque état de la ligne. La description faite ci-dessous en reprend les éléments les plus importants.

3.2.1 Éléments de caractérisation

3.2.1.1 Caméra 3D

Le rôle de cet élément est de déplacer de caractériser le volume et la forme de chaque déchet. Ces informations sont ensuite utilisées pour définir les points de préhension des robots ainsi que la masse de l'objet pour une masse volumique donnée.

La caméra utilisée est le modèle une Ruler E de la marque Sick [2]. Comme le montre la Figure 3.3, la caméra utilise une bande laser qui scanne les éléments qui traversent les rayons. Avec l'aide de l'encodeur du convoyeur, il est ensuite possible de reconstituer le volume d'une pièce à partir des tranches mesurées. Une attention particulière doit être portée afin d'occulter au mieux la caméra puisque même la lumière naturelle a tendance à fausser les mesures.

Le contrôle et l'acquisition des données de la caméra sont effectués par connexion TCP/IP avec un ordinateur et seule la mise sous tension peut éventuellement être à contrôler par l'automate.

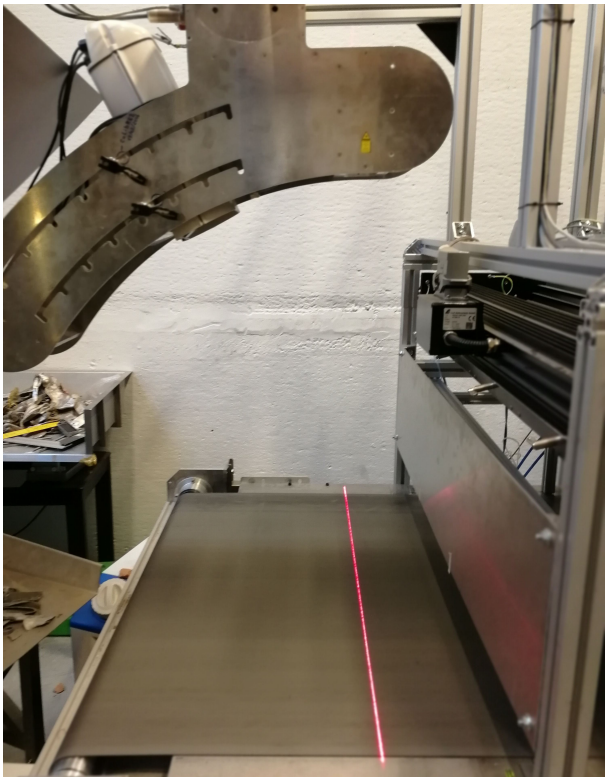


FIGURE 3.3 – Intégration de la caméra 3D dans le prototype.

3.2.1.2 Caméra hyperspectrale

Le rôle de cet élément est mesurer le spectre caractéristique de chaque déchet métallique puis de le comparer avec des valeurs connues afin d'en déduire la famille de métaux auquel le déchet appartient. Un papier a notamment été rédigé à propos du fonctionnement de cette caméra de type VNIR, Visible to near-infrared, par les chercheurs du GeMMe [4].

L'émission lumineuse est amplifiée à l'aide des lampes, de types halogènes ou LED, qu'il convient de contrôler avec l'automate, car leur fonctionnement peut entraîner des dégâts sur la bande si elle est à l'arrêt. De plus, l'échauffement de ces dispositifs d'émissions nécessite l'usage d'un système de refroidissement externe.

Le contrôle et l'acquisition des données de la caméra hyperspectrale VNIR sont effectués par connexion TCP/IP avec un ordinateur et seule la mise sous tension peut éventuellement être à contrôler par l'automate.

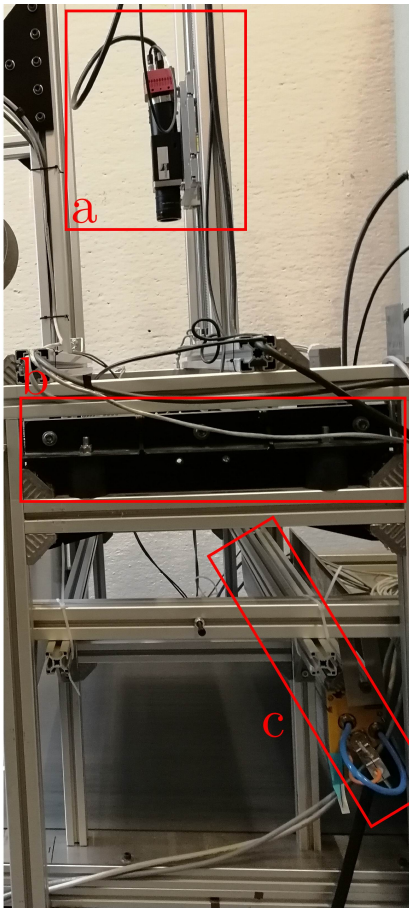


FIGURE 3.4 – Disposition des composants du système d'acquisition par caméra hyperspectrale. (a) Caméra hyperspectrale VNIR. (b) Éclairage halogène. (c) Éclairage LED

3.2.1.3 Système à rayon X

Le rôle de cet élément est d'émettre des rayons X sur les déchets puis de mesurer le spectre réfléchi et d'en déduire des informations sur notamment la densité du matériau.

L'émetteur des rayons X s'effectue à l'aide de l'appareil IXS Series P198 de la marque VJ Technologies [24] et est alimenté par une source de puissance Acopian Modèle S17480-230. Le contrôle de l'émission se fait à l'aide d'un application Windows dédiée qui permet d'une part de gérer la procédure de démarrage des rayons et, d'autre part, de s'assurer que les différentes règles de sécurités sont bien respectées. Ces règles de sécurité sont très strictes, car l'exposition non contrôlée de rayons X sur un humain peut avoir des conséquences sanitaires importantes. Le contrôle et la communication entre l'émetteur et l'ordinateur a lieu au moyen d'un port série de type RS232.

Le capteur eValuator - 3500 de la marque Kromek permet d'extraire des informations sur la densité des déchets suite à la réflexion des rayons envoyés par l'émetteur. L'ensemble est composé d'un contrôleur, une alimentation et du capteur en lui-même. Le contrôleur collecte les données des capteurs via un port série (DB9) et les transmet au PC d'acquisition un port Ethernet (RJ45).

FIGURE 3.5 – Caisson à l'intérieur duquel s'effectue l'analyse par rayons X.

L'ensemble de la caractérisation à rayon X se déroule dans un caisson blindé, présenté à

la Figure 3.5 qui empêche les rayons X de se propager en dehors de l'enceinte. Toutefois, un contrôle sécuritaire additionnel est nécessaire pour diminuer le risque d'accident.

3.2.1.4 Système LIBS

Le principe d'un LIBS, pour *laser-induced breakdown spectroscopy*, est de permettre une analyse de la composition détaillée des déchets en analysant les émissions produites lorsqu'une partie du matériau passe à l'état plasma[12]. Pour un dispositif multidirectionnel comme celui installé sur le prototype, le système est capable d'orienter un faisceau laser et de tirer de manière à sublimer puis ioniser la matière localement.

En ce qui concerne l'intégration, les résultats de l'analyse spectroscopique sont transmis au PC par connexion TCP/IP. Les opportunités de contrôle à distance de l'appareil ne sont pas bien définies, car, au vu du caractère très expérimental de la technologie et de la jeunesse du fabricant, il existe peu de documentation à ce sujet. Le système est relativement indépendant puisqu'il possède sa propre armoire électrique sur laquelle est placé un HMI qui sert à afficher le statut et les erreurs de la machine. Il est toutefois à noter que l'appareil a besoin d'une source de refroidissement externe et ne s'active pas si elle ne lui est pas fournie.

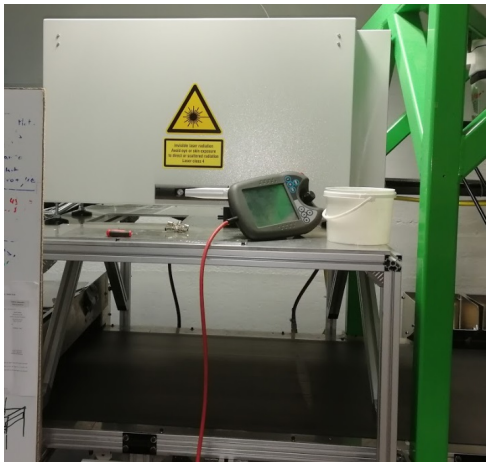


FIGURE 3.6 – Appareil à spectroscopie sur plasma induit par laser.

La Figure 3.6 présente l'intégration du LIBS sous sa forme actuelle dans le prototype. Au

vu des risques liés à la réflexion d'un rayon laser à partir de la surface réfléchissante des déchets, l'ajout d'une enceinte spéciale est à prévoir.

3.2.2 Éléments de tri

3.2.2.1 Robots

Les robots utilisés pour effectuer le tri sont des robots IRB340 Flexpicker de la marque ABB qui sont présentés à la Figure 3.7. Ce modèle de robots *Deltas*, ou parfois appelés *araignées* a la spécificité d'être très rapide, car la position des moteurs et les matériaux qui le composent donnent lieu à une très faible inertie de l'effecteur. De plus, la versatilité et le grand espace de travail que le flexpicker autorise le rendent plus approprié qu'un robot de type SCARA sur ce genre de ligne de tri. Le principal défaut de ce modèle est la charge manipulable qui limite donc le poids des objets à trier.

Après avoir configuré les actions possibles sur leur propre contrôleur, les instructions des robots sont envoyées par le PC pour chaque déchet à trier. Comme ces robots se déplacent très rapidement et qu'il n'y a pas d'enceinte fermée pour les contenir, il faut idéalement pouvoir contrôler leur mise en marche et arrêt à l'aide de l'automate.

3.2.2.2 Soufflettes

N'étant plus utilisées sur l'actuelle version du prototype, les soufflettes n'exercent plus de rôle sur le tri. À titre informatif, celles-ci étaient placées à la fin de la bande convoyeuse et propulsaient vers le haut les éléments plats et fins comme les circuits imprimés. Ces soufflettes étaient contrôlées par leur propre automate qui recevait les positions des pièces de la part du PC.

FIGURE 3.7 – Robots utilisés pour le tri des déchets. À l'avant-plan, le robot déplaceur et à l'arrière-plan le robot préhenseur.

3.2.3 Éléments de contrôle

3.2.3.1 Armoire électrique

Bien que l'armoire électrique soit elle-même composée d'un grand nombre d'éléments, elle peut être considérée comme un élément à l'échelle macroscopique. La Figure 3.8 permet de scinder les zones de l'armoire selon le rôle des composants. On distingue ainsi :

- Les disjoncteurs qui permettent de limiter le courant apporté à chaque composant (a),
- Les modules de relais de sécurités qui sont connectés à l'arrêt d'urgence (b),
- L'automate Beckhoff C6930-0040 (c),
- Les sources de courant continu (d),
- Les diviseurs de pulses de l'encodeur qui permettent de diminuer la fréquence des pulses pour qu'ils soient utilisables avec chaque appareil (e),
- Les cartes d'entrées/sorties digitales et analogiques ainsi que le coupleur de bus

connecté au PLC par EtherCAT (f),

— Le switch réseau qui permet aux appareils de communiquer entre eux via Ethernet (g),

— La zone de report des entrées sorties des entrées/sorties de (f) dont l'objectif est d'alléger les branchements électriques (h),

— Le HMI formé de boutons et de voyants physiques, mais aussi d'un écran tactile.

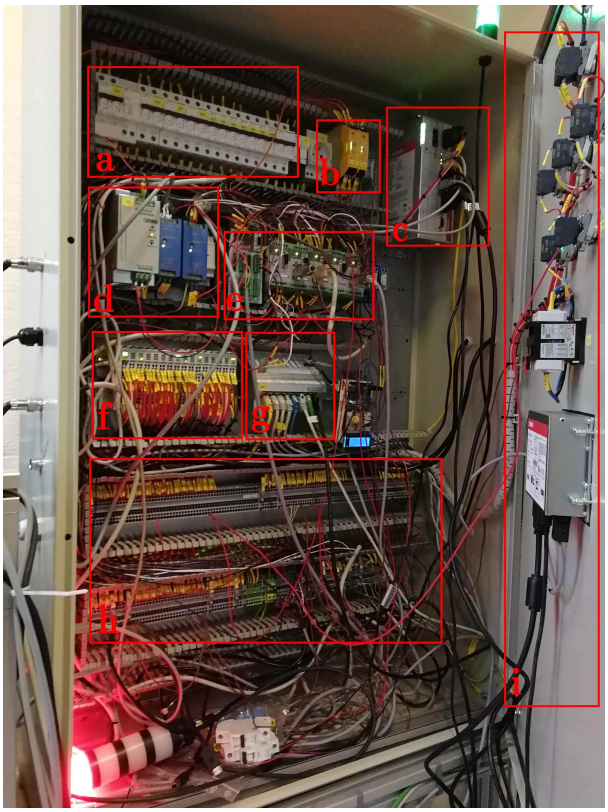


FIGURE 3.8 – Armoire électrique du prototype Pick-It. (a) Disjoncteurs. (b) Relais de sécurité. (c) Automate Beckhoff. (d) Source de tension DC. (e) Diviseurs des pulses de l'encodeur. (f) Module et cartes d'entrées/sorties de l'automate. (g) Switch Ethernet. (h) Zone de connexion avec les entrées/sorties de l'automate.

Comme il serait trop inutilement fastidieux d'énoncer chacun des éléments et d'écrire en français leur rôle, le lecteur qui souhaiterait approfondir le sujet est invité à consulter les plans électriques et la liste des composants électriques avec les références .0.6.

3.2.3.2 Ordinateur d'acquisition

L'ordinateur d'acquisition est un PC opérant sous Windows 10 qui a pour objectif de traiter les informations reçues par les différentes caméras et capteurs. L'application développée par l'équipe du GeMMe est ainsi capable de rassembler les informations correspondantes à une même pièce puis calculer la classe d'appartenance de ces pièces et enfin d'envoyer l'action à effectuer aux robots. Au vu de l'importance du contrôle de cet élément, il est primordial d'établir une communication avec le PC.

3.2.4 Éléments divers

3.2.4.1 Alimentation vibrante

Le rôle de l'alimentation vibrante est d'alimenter le convoyeur en débris métallique. Il s'agit d'un modèle standard de la marque Tuxel et est composé d'un entonnoir et d'un plateau posé sur une embase électromagnétique vibrante. À mesure que la fréquence de vibration de l'embase augmente, le nombre de déchets déposés sur le convoyeur augmente également.

Le contrôle de cette alimentation s'effectue au moyen d'une commande composée d'un potentiomètre pour le réglage de la fréquence et d'un bouton de marche/arrêt. Le modèle du prototype ne possède pas d'entrée ni de sortie pour un permettre un contrôle à distance via un automate, mais il suffirait de changer de commande pour rendre cette fonction disponible comme un contrôleur R3FSC [20].

3.2.4.2 Bande transporteuse

Le rôle de cet élément est de déplacer les débris au travers des différents appareils d'identification puis de les présenter aux robots qui effectuent le tri. Le choix des dimensions de

FIGURE 3.9 – Alimentation vibrante [19].

la bande a dépendu des rayons d'action des robots et des capteurs et une largeur de 1m pour une longueur de 8m ont finalement étés retenus. L'ensemble composé de la bande, du moteur et de son contrôleur est de la marque Lenze.

Le contrôle du moteur est effectué à l'aide d'un Inverter Lenze 8400 StateLine C[22]. À son tour, ce drive peut être contrôlé par une télécommande dont la référence est Diagnosis terminal X400 ou à l'aide d'une connexion par un terminal d'entrées/sorties.

3.2.4.3 Refroidisseur

Comme mentionné dans la description des éléments fonctionnels, certaines machines nécessitent d'utiliser un circuit de refroidissement externe afin de fonctionner. Face à ce besoin, un refroidisseur à eau HRS024-AF-20 de la marque SMC [23] a été installé.

Ce composant peut être contrôlé à l'aide d'une série d'entrées/sorties digitales ou encore via un port série (DE-9). De cette manière, il est possible de contrôler la température de décharge du liquide et de signaler si cette température est trop élevée.

3.3 Besoins du GeMMe

Lors de la première visite du prototype, en janvier 2018, il est apparu que le GeMMe avait un besoin concernant l'automatisation du prototype. En effet, malgré la cinquantaine d'entrées et sorties prévues pour le contrôle par automate, seule une dizaine d'entre elles était réellement utilisée. Bien que certaines séquences simples avaient étés programmées, les changements de la disposition du prototype les avaient rendues obsolètes. Au-delà d'une sous utilisation des capacités du matériel, ces problèmes donnaient également lieu à des manquements sécuritaires.

Une discussion avec Pierre Barnabé et Godefroid Dislaire a permis d'identifier et synthétiser les besoins qui sont décrits par les sections ci-dessous.

3.3.1 Analyse fonctionnelle

L'analyse fonctionnelle du système a permis d'établir les fonctions de services établies dans la table 3.1. Puisque le prototype est déjà fonctionnel, il est permis de ne pas se préoccuper de la fonction principale et plutôt de se concentrer sur les fonctions de service secondaires et les fonctions de contraintes.

3.3.2 Délivrables

Comme expliqué au Chapitre 2, il a été décidé de régler certains problèmes liés au PLC du GeMMe tout en travaillant en parallèle sur le développement des solutions liées au démonstrateur. C'est ainsi que se justifie la présence d'éléments supplémentaires non sollicités, car le travail est le même pour le prototype et le démonstrateur. Par exemple, le besoin d'une fonction "Purge" du système n'a pas été formulé par les chercheurs du GeMMe mais figure dans la solution élaborée au chapitre 5 par souci de généralité.

ID	Type	Pond.	Libellé	Critère	Niveau	Flex.	Remarques
FP1	Principale	100	La ligne identifie puis trie les déchets parmi 26 classes et nuances de métaux.	Débit de déchet	4,9 pcs/s ou plus	F0	Le débit de 4,9 pcs/s concerne le démonstrateur.
				Classes identifiées	2 à 26 classes		
FS2	Secondaire	80	Le PC peut synchroniser les appareils de la ligne.	Synchronisation	Synchronisé ou non	F1	
FS3	Secondaire	70	L'utilisateur contrôle les éléments de la ligne	Activation manuel	On/Off	F2	
				Configuration	Fichier config. chargé		
				Vitesse (convoyeur)	[0-5 m/s]		
FS4	Secondaire	70	L'utilisateur supervise le comportement de la chaine	Visibilité et Ergonomie	États de voyants État de la tour lumineuse HMI	F2	
FS5	Secondaire	60	L'utilisateur peut consulter à postériori le comportement de la chaine	Actions précédentes	Affichage de logs et performances	F3	
FC6	Contrainte	100	Le dispositif est sécuritaire	Normes ISO 45001 et ISO 10218-1 :2011	Respect absolu	F0	
FC7	Contrainte	90	Les éléments du dispositif ne s'abîment pas entre eux.	Éclairage du convoyeur à l'arrêt	Max. 1 s	F1	

Tableau 3.1 – Tableau récapitulatif des différentes fonctions de service.

Après avoir présenté les besoins aux experts de chez Citius, certaines demandes ont étés reformulées pour répondre à des règles de bonnes conduites industrielles ou de sécurité. C'est ainsi que, même si les fonctionnalités ont étés implémentées, il est possible qu'elles ne soient pas rendues accessibles dans le contexte initialement demandé.

3.3.2.1 Séquences

Séquence de synchronisation Cette séquence a pour but la synchronisation entre les appareils de la ligne. Cette synchronisation est nécessaire, car le fonctionnement des appareils de mesures et des robots dépend des pulses de l'encodeur du convoyeur, mais ces pulses ne correspondent pas nécessairement à une base commune puisque les machines

ne démarrent pas en même temps.

Concrètement, pour réaliser cette synchronisation, il est nécessaire de bloquer les signaux de l'encodeur pour les faire redémarrer en même temps. Durant le blocage des signaux, un reset des acquisitions doit également être effectué par l'ordinateur ce qui nécessite l'établissement d'une communication entre le PLC et le PC comme stipulé dans le schéma de l'annexe .0.6. Comme la référence du robot dépend du signal de page de l'encodeur, c'est au front montant de ce signal carré, qui vaut 0V durant un demi-mètre puis 24V durant le demi-mètre suivant, que le redémarrage des signaux doit être enclenché.

Séquence de tri et d'acquisition Durant cette séquence, l'ensemble des appareils d'acquisitions sont activés et le tri s'effectue à l'aide des robots. Idéalement, il faudrait aussi être capable de désactiver l'action des robots si une séquence d'acquisition uniquement est désirée.

Séquence d'arrêt La séquence d'arrêt permet aux équipements de s'éteindre dans des bonnes conditions, contrairement à un arrêt d'urgence brutal. Lorsque cette séquence est demandée, une instruction doit notamment être envoyée au robot pour qu'il termine sa tâche. Afin de veiller à ne pas brûler le convoyeur, il est important que l'éclairage de la caméra hyperspectrale s'éteigne avant le convoyeur.

3.3.2.2 Interface homme-machine

Afin de pouvoir contrôler et superviser la ligne, une demande d'un HMI a été formulée par le GeMMe et est présentée à l'annexe .0.6. Selon leurs spécifications, les fonctionnalités à implémenter sont les suivantes :

— Une page Status dans laquelle le statut est affiché et peut être commuté à l'aide d'un bouton,

— Une page Rules dans laquelle les règles de sécurité entre les appareils peuvent être activées et désactivées,
— Une page Tasks dans laquelle les différentes séquences peuvent être initiées,
— Une page Logs dans laquelle les performances et les dernières actions sont reprises.

3.3.2.3 Sécurité

Le danger des appareils utilisés sur la ligne rend nécessaire une intégration plus poussée de l'arrêt d'urgence dans les séquences du PLC. Puisque l'arrêt d'urgence est déjà électriquement connecté aux interlocks des éléments dangereux, le travail à fournir consiste à s'assurer que les machines n'auront pas un comportement inapproprié en cas de relâchement de l'arrêt d'urgence. Seuls certains appareils sont concernés par cette attention particulière et sont les rayons X, le LIBS, le convoyeur et les robots.

Chapitre 4

Automates programmables

L'objectif de cette section est de constituer une synthèse des connaissances nécessaires à l'automatisation d'un procédé via l'utilisation d'*automates programmables*. Dans un premier temps, les concepts fondamentaux liés aux automates programmables seront exposés. Bien que loin d'être exhaustif, ce recueil a d'une part comme ambition de permettre à un ingénieur de s'initier à l'automation par ordinateurs industriels. D'autre part, la description de la norme de CEI 61131-3 a pour objectif d'être assez complète pour permettre à un technicien de comprendre quelles sont ses spécificités et ses intérêts par rapport aux anciens langages. Finalement, le concept de Human Machine Interface sera développé dans un contexte industriel.

Bien que ce chapitre puisse apparaître comme étant fort long et descriptif, il est cependant nécessaire pour distinguer le caractère innovant de ce TFE. Ce n'est en effet qu'en se familiarisant avec les nouvelles normes et technologies de l'automatisation qu'il est possible de relever l'originalité de l'approche suivie durant ce travail.

4.1 Définition

Le site internet automation-sense définit de manière générale un automate comme suit :

> Un *automate programmable industriel* (ou API) est un dispositif similaire à un ordinateur, ayant des entrées et des sorties, utilisées pour automatiser des processus comme la commande des machines sur une ligne de montage dans une usine. [15]

Par la suite, la notion d'automate est également mentionnée sous le nom de PLC de l'anglais *Programmable Logic Controller*.

4.2 Évolution et historique

Au-delà de l'intérêt culturel, il est primordial de connaître les origines des automates programmables ainsi que les évolutions technologiques qui les ont amenés à être ce qu'ils sont aujourd'hui. Les difficultés de former le personnel confèrent au monde de la production industrielle une certaine inertie face aux abondantes nouvelles technologies. C'est ainsi que l'utilisation de certains paradigmes de programmation dépassés s'explique et se rencontre encore dans l'industrie.

L'histoire moderne des systèmes de contrôle automatisés commence avec des régulateurs de vitesse mécaniques utilisés dans les moulins au 17$^{\text{ème}}$ siècle et ensuite avec les machines à vapeur au 18$^{\text{ème}}$ siècle. Au début du 20$^{\text{ème}}$ siècle, la popularisation de l'électricité et des connaissances en traitement de signaux permettent le développement de contrôleurs PID capable contrôler de manière continue le cap des navires en fonction des perturbations extérieures telles que le vent et les vagues.

À la même époque, apparaissent également des systèmes de *logique câblée* ou aussi *logique à relais* puisque ces circuits utilisent des relais électromagnétiques qui sont commutés

pour adapter leurs états et sorties. Ces systèmes purement combinatoires ont ainsi été utilisés dans l'industrie jusqu'à la fin des années soixante. Suite aux évolutions rapides des produits à fabriquer, un ingénieur de chez General Motors a formulé la demande d'une interface virtuelle pour éviter d'avoir à reconnecter manuellement les boutons, relais et sorties à chaque modification de la ligne [3]. C'est ainsi que Bedford Associates proposa le premier PLC baptisé MODICON, pour MOdular DIgital CONtroller, en 1969. Alors que le PLC était initialement conçu pour être programmé à l'aide d'un langage assembleur ou de langages de haut niveau comme le Fortran, c'est pour éviter de devoir former les automaticiens que le langage ladder graphique, adaptation virtuelle de la logique à relais, a finalement été adopté.

À l'heure actuelle, il existe de nombreux constructeurs de PLC qui ont souvent développé leur propre standard matériel et de programmation. Face aux problèmes liés aux différences des langages entre les constructeurs, un standard commun a été établi par la norme CEI 61131-3 pour unifier les normes de développements en définissant des langages à utiliser.

4.3 Utilisation

Les PLC sont largement utilisés dans le monde industriel pour permettre le contrôle et la supervision de processus de production. D'une part, ces appareils permettent la mise en place d'une série de séquence d'actions déclenchées à l'aide des sorties digitales ou analogiques. D'autre part, le comportement d'une chaîne peut être adapté en fonction des informations fournies par les entrées connectées aux capteurs placés sur la ligne pour notamment permettre un contrôle de la sécurité.

Le choix de l'usage d'un contrôleur PLC dépend des besoins de la solution au travers de la facilité d'implémentation, des fonctionnalités nécessaires et du prix. La figure 4.1 montre ainsi les solutions de contrôle généralement envisagées en fonction des fonctionnalités

nécessaires. Un critère additionnel à considérer est la fiabilité puisque certains éléments d'un procédé industriel doivent absolument être pilotés par des contrôleurs respectant des normes strictes. Il est à noter toutefois que certains nouveaux automates industriels utilisent désormais une couche PC qui permet de profiter des fonctionnalités d'un ordinateur classique tout en garantissant la sécurité d'un automate industriel. Les différences entre un PC et un PLC sont toutefois amenées à persister, car l'exécution d'un code automate repose sur des principes fondamentalement différents de ceux de l'exécution : L'intégralité du code PLC doit être exécuté à chaque temps de cycle pour recalculer ses sorties alors que, lorsqu'un PC atteint sa dernière instruction, le code s'arrête.

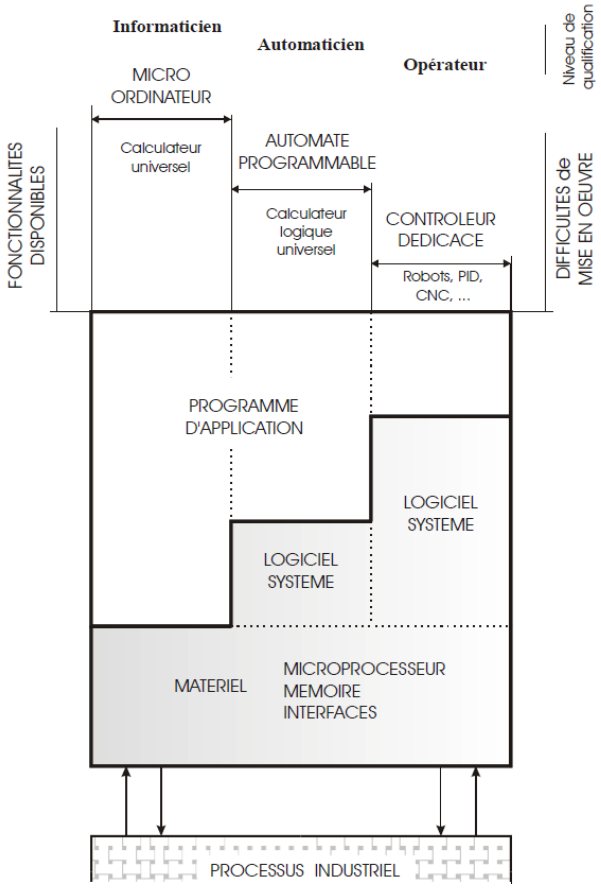


FIGURE 4.1 – Solutions de contrôle adéquates en fonction des fonctionnalités nécessaires[14].

4.4 Matériel

De manière générale, un automate programmable contient une unité logique qui lui permet d'effectuer des calculs appelés CPU à laquelle viennent s'ajouter des modules supplémentaires en fonction des besoins tels que :

- Des cartes d'entrées/sorties soit directement connectées à l'automate, soit au moyen de bus de communication,
- Une interface homme-machine,
- Un automate de sécurité qui permet d'assurer une meilleure fiabilité et réactivité des fonctions d'urgence.

La disponibilité des modules et leur intégration native dépend des constructeurs et fait ainsi partie des considérations prises en compte lors du choix de l'automate.

4.5 Programmation avec la norme CEI 61131-3 [25]

4.5.1 Motivations

La troisième part de la norme CEI 61131, a pour but d'uniformiser les langages utilisés lors de la programmation de PLC. En effet, depuis la création des automates, chaque constructeur a développé ses propres langages de programmation avec des mots-clefs et des fonctionnalités différentes. Ces différences entre constructeurs ont eu ainsi tendance à compliquer l'utilisation d'automates de plusieurs marques par un automaticien. Dans la mesure où les besoins de l'industrie 4.0 imposent aux constructeurs de PLC une interconnectibilité de plus en plus grande, ce genre de standardisation prend toute son importance et permet à un programmeur de profiter de l'ensemble des produits du marché.

Dans cette optique, la Commission électrotechnique internationale (CEI) a défini en 1993, puis mis à jour en 2013, un standard qu'il incombe aux constructeurs de respecter et

concerne les cinq langages de programmation suivants :

Langages graphiques

— Ladder Diagram (LD) ou langage ladder/schéma à contacts en français,

— Function Block Diagram (FBD) ou boites fonctionnelles en français.

Langages textuels

— Structured Text (ST) ou texte structuré en français,

— Instruction List (IL) ou liste d'instructions en français,

— Sequential Function Chart (SFC).

Il est important de noter que les spécifications de la norme laissent aux constructeurs la possibilité d'implémenter des fonctionnalités supplémentaires qui peuvent ainsi expliquer l'existence de différences entre les codes provenant de deux constructeurs différents. De plus, il existe certains cas de figure très particuliers pour lesquelles le comportement du programme n'est pas extensivement spécifié par la norme et qui sont donc à éviter pour ne pas dépendre de l'interprétation du constructeur.

4.5.2 Éléments de base de la norme CEI 61131-3

Afin de pouvoir comprendre entièrement un code écrit à l'aide de la norme, il est nécessaire de se familiariser avec les éléments de base qui la composent. Comme certains de ces éléments sont communs à la plupart des langages de programmation, une attention particulière est fournie aux éléments ayant un caractère innovant. Le lecteur désireux d'approfondir le sujet est invité à consulter les ressources propres à son automate ou un livre de référence[25].

4.5.2.1 Program Organisation Unit (POU)

Les unités d'organisation du programme ou Program Organisation Unit (POU) constituent l'ensemble des blocs qui permettent de construire le programme à exécuter. Ces différents types de blocs reprennent les blocs auparavant définis dans le standard DIN 19239 pour les réduire au nombre de trois : Programme, Fonction et Bloc de fonction dans lesquels la logique du programme va être programmée.

La spécificité première d'un Programme est d'accéder aux variables d'entrées/sorties et d'appeler les éventuels blocs de fonctions et fonctions. Les Fonctions servent à retourner une valeur en fonction des entrées qui lui sont fournies tandis que les Blocs de fonctions sont souvent assimilés à des fonctions possédant une mémoire.

Les blocs de fonctions constituent une réelle innovation de par leur caractère orienté-objet. Pour un technicien ayant des connaissances plus approfondies en informatique moderne, un parallélisme très clair peut être fait avec le concept de Classes pour les raisons suivantes :

- Avant de pouvoir être utilisés, les blocs de fonctions doivent être instanciés,
- Des méthodes peuvent être définies pour notamment manipuler des variables internes au bloc de fonction,
- Il est possible de définir des blocs de fonctions parents puis d'en faire hériter les propriétés à d'autres blocs de fonctions enfants à l'aide du mot-clef EXTENDS,
- Des "Propriétés" peuvent être définies pour générer l'équivalent des accesseurs Get/Set,
- Des "Interfaces" permettent d'imposer des méthodes à définir à l'aide du mot-clef IMPLEMENTS pour permettre une harmonisation du développement des méthodes entre plusieurs classes.

Ces fonctionnalités, mises en avant-plan par la norme, laissent suggérer que les approches orientées objet sont de plus en plus encouragées et que l'objectif du CEI 61131-3 est de combler progressivement les différences qui existent entre les langages de programmation

des automates et ceux des ordinateurs. Contrairement aux autres langages orientés objet, il est à noter que les propriétés listées ci-dessus sont, pour la plupart, également définies pour les blocs de type Programme.

4.5.2.2 Variables, types de données et éléments

La norme IEC 61131-3 permet l'utilisation de différents types de variables qui déterminent comment celles-ci sont interprétées et combien d'espaces de la mémoire leur sont alloués.

Les types classiques parmi lesquels se retrouvent les booléens, les entiers, les réels, les chaînes de caractères, etc., mais également des types moins conventionnels comme des types temporels et des types définis par l'utilisateur tels que des énumérations, des structures, etc.

Lors de la déclaration d'une variable, l'identifiant et le type sont les caractéristiques minimales à spécifier. De plus, l'adresse de la variable peut être définie manuellement ainsi que certains symboles pour indiquer que la variable est liée à une entrée ou une sortie. À titre d'exemple,

myBoolOutput AT%Q* : BOOL := TRUE; et myBoolInput AT%I* : BOOL;

représentent les déclarations respectives d'une entrée et d'une sortie booléenne.

4.5.3 Langages de programmation

Afin de respecter les préférences et les besoins d'un grand nombre de programmeurs, la norme CEI permet l'utilisation de cinq langages de programmation différents parmi lesquels deux langages graphiques et trois langages textuels.

4.5.3.1 Instruction List - IL

C'est un langage historiquement très utilisé et qui s'apparente à du langage assembleur avec une gestion de la mémoire automatisée. L'exécution s'effectue de haut en bas tel que chaque ligne du code corresponde à une action appelée instruction. Pour chaque instruction, l'action à effectuer est décrite au moyen d'un Label, d'un Opérateur et d'une Opérande. Une spécificité de ce langage se situe, comme pour le langage assembleur, dans l'existence du type Current Result qui reproduit le principe physique d'un accumulateur. Ces opérateurs spécifiques font de l'Instruction List un langage très puissant, mais également difficile à aborder pour un non-initié.

4.5.3.2 Structured Text - ST

Le Structured Text est un langage assez similaire au C ou au Pascal. Composé de statements, l'exécution du programme s'effectue ligne par ligne de haut en bas. Il existe des opérateurs spécifiques à ce langage permettant des itérations (while, for, repeat, etc.), de la multisélection (switch), mais pas d'opérateur de saut (jump) contrairement à l'Instruction List. Le Structured Text est donc compact et est un langage de haut niveau, mais ne permet pas de contrôle sur la traduction en code machine et donc potentiellement moins efficace qu'un code équivalent en Instruction List.

4.5.3.3 Function Block Diagram - FBD

Assez similaire au Ladder diagram, un bout de code défini en Function Bloc Diagram possède une partie de déclaration textuelle et une partie de code divisée en réseaux. Les blocs de fonctions caractéristiques à ce langage se présentent comme des boites rectangulaires avec les entrées à gauche et sorties à droite et dont le contenu peut être aussi simple qu'un *ET logique* ou être une fonction mathématique complexe. Contrairement au ladder, les connexions parallèles entre deux branches ne sont pas autorisées. L'ordre

d'exécution du code est moins évident et se base sur le principe de stabilité de l'entrée. D'une part, la sortie d'un bloc de fonction n'est activée que lorsque ses entrées sont stables. D'autre part, et de manière plus conventionnelle, la lecture des réseaux se fait de haut en bas. L'utilisation de variable de feedback est une pratique étonnante pour un non-initié, mais courante dans laquelle une entrée du cycle courant correspond à une sortie du cycle précédent.

Un exemple d'élément de code programmé en FBD est présenté à la Figure 4.2

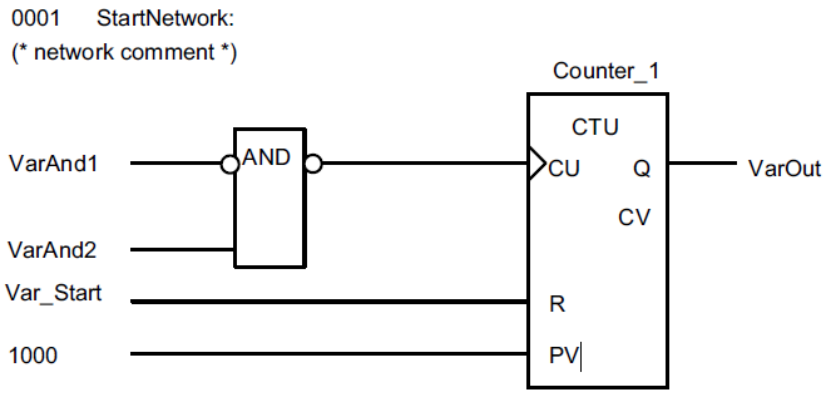


FIGURE 4.2 – Exemple d'un réseau de programme codé en Function Block Diagram[25]. Le bloc de fonction Counter_1 compte le nombre de pulses positifs et set sa sortie à True dès que ce nombre dépasse 1000.

4.5.3.4 Ladder Diagram - LD

Le langage Ladder est un langage historiquement lié aux anciennes séquences à relais mécaniques. Les programmes écrits en Ladder Diagram simulent le comportement d'un circuit électrique pour lesquels des signaux digitaux sont transmis aux contacts et aux sorties par des câbles de connexion. Un programme est composé de réseaux exécutés de haut en bas, pour lesquels l'alimentation en puissance est fournie du coté gauche et dont les sorties du côté droit. Bien qu'un programme ladder traite principalement des signaux, l'activation de fonctions particulières est possible (Timers, EN/ENO) tandis que les fonctionnalités de saut et de feedback sont implicites.

Un exemple d'élément de code programmé en ladder est présenté à la Figure 4.3

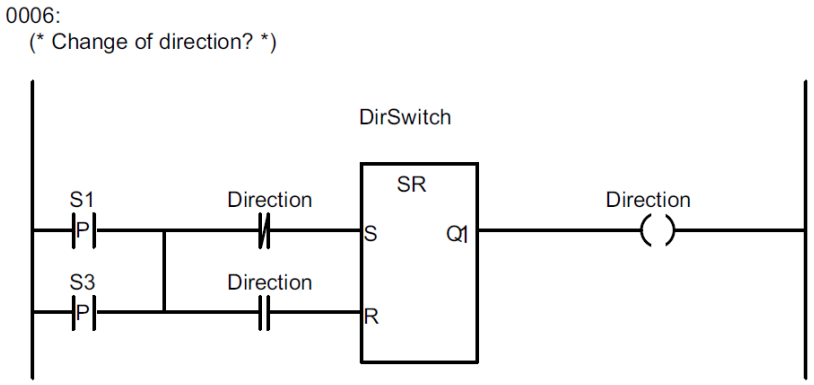


FIGURE 4.3 – Exemple d'un réseau de programme codé en Ladder Diagram[25]. Un bloc de fonction de type Reset est utilisé pour changer le booléen définissant la direction choisie.

Il est à noter que le Ladder Diagram est un langage plus répandu aux États-Unis qu'en Europe et qu'une version locale plus complexe y a donc été développée. Toutefois ses fonctionnalités n'ont pas spécialement étés intégrées dans la norme CEI 61131-3 .

4.5.3.5 Sequential Function Chart - SFC

Le langage SFC permet de découper une tâche en une séquence d'actions. Ce langage est donc idéal pour les tâches séquentielles pour lesquelles les actions s'effectuent à la suite les unes des autres. La représentation graphique est dérivée des réseaux de Petri et du langage descriptif GRAFCET. Le passage d'une étape à l'autre se fait suite à la vérification d'une condition appelée transition. Pour chaque étape active, les actions contenues dans les blocs d'actions sont activées. Lors de la définition de chacune de ces actions, un qualifier est utilisé et définit la durée et le moment d'exécution de l'action. Ces actions peuvent être le set/reset d'une valeur booléenne, mais aussi le lancement d'autres séquences d'action décrites dans d'autres langages de la norme.

Un exemple d'élément de code programmé en SFC est présenté à la Figure 4.4

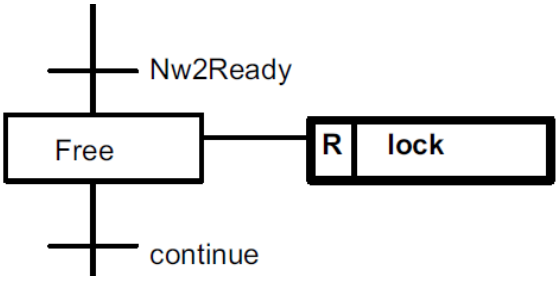


FIGURE 4.4 – Exemple d'élément d'un GRAFCET codé en langage Sequential Function Chart [25]. À l'étape Free, le booléen "Lock" est resetté et le programme reste dans cet étape tant que la transition "continue" vaut False.

4.5.3.6 Comparaison

Au vu du nombre de langages possibles pour développer une application en IEC 61131-3, il convient de les comparer entre eux pour déterminer lequel utiliser. Certains langages sont plus aptes que d'autres pour effectuer certaines opérations. Le parcours d'un tableau est par exemple plus facile à effectuer en texte structuré qu'en langage ladder tandis qu'une condition OU peut paraître plus claire en Ladder qu'en Instruction List. La Figure 4.5 montre une comparaison entre différents langages effectuant la même action. Bien qu'il serait tentant de faire un classement entre les langages, il faut remarquer qu'il n'existe pas spécialement un langage meilleur que les autres et que la force de la norme est justement d'offrir la possibilité de jongler entre ces différents outils au sein d'un même programme. Le choix à faire n'est donc pas unique ni objectif et dépend généralement du type de fonctionnalité à implémenter, des connaissances préliminaires de l'automaticien ou encore des habitudes de l'entreprise dans laquelle il travaille.

4.5.4 Fonctionnalités standardisées

Afin de permettre aux utilisateurs de bénéficier de l'usage de fonctionnalités communes, la norme CEI 61131-3 impose aux constructeurs d'implémenter une série de fonctions et de blocs de fonctions. Quel que soit le constructeur choisi, un utilisateur peut ainsi savoir que ces fonctions sont implémentées.

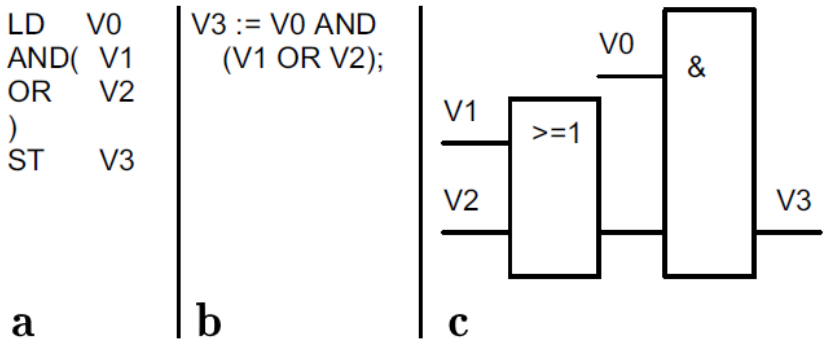


FIGURE 4.5 – Comparaison entre les différents langages de programmation pour l'implémentation de fonction logique. (a) Instruction List (b) Structured text (c) Function Block Diagram [25].

4.5.4.1 Fonctions

La norme nécessite un certain nombre de fonctions usuelles et permet un certain confort lors du développement des applications. Ces fonctions concernent des conversions de types, des calculs numériques fréquents (sqrt, abs, log, etc.) et des opérations sur des chaînes de caractères.

Lorsqu'une même fonction est définie pour un nombre d'arguments variable, la fonction est alors qualifiée d'extensible. Si par ailleurs, la fonction accepte plusieurs types d'entrées, elle est qualifiée de surchargée.

4.5.4.2 Blocs de fonctions

Les blocs de fonctions sont des éléments souvent présentés comme étant des fonctions ayant une mémoire. En effet, tout comme les fonctions, un bloc de fonction retourne une ou plusieurs sorties qui dépendent des entrées fournies à un moment donné, mais peuvent aussi dépendre des entrées des temps précédents. Un exemple courant est le bloc de fonctions R_TRIG, standardisé par la norme, qui permet une détection de bord, i.e. de détecter lorsqu'un signal passe de 0 à 1. Les timers sont également des blocs de fonctions communs pour lesquels le temps écoulé est la valeur sauvegardée en mémoire.

4.5.5 Configuration

Bien que cet aspect ne concerne pas directement les langages de programmation, il est intéressant de noter que la norme standardise également la configuration des automates. C'est un aspect assez audacieux, car les configurations sont souvent liées au matériel et donc différentes d'un constructeur à l'autre. Cette standardisation est toutefois un besoin face aux nouvelles technologies comme la multiplicité des processeurs, la modularité, le run-time adapté et les nombreuses voies de communication. La norme spécifie ainsi le moyen d'attribution des ressources (fréquence d'exécution, priorité, etc.), l'accessibilité des variables et le partage des données.

4.5.6 Types de PLC

On distingue ici l'ensemble des systèmes d'exécution autorisés par la norme IEC 61131-3. À noter que comme la norme ne fait plus de différence entre PLC, Ordinateur de Process et PC, la norme permet une très grande liberté et les déclinaisons suivantes sont possibles :

- PLC compacte avec modules d'entrées/sorties intégrés,
- PLC modulaire avec possibilité d'ajout des cartes d'entrées/sorties,
- Contrôleur embarqué,
- PC embarqué avec matériel typique d'un PC et Windows CE/Realtime Linux,
- PC standard avec cartes d'entrées/sorties ajoutées.

4.6 Human Machine Interface

Un HMI, de l'anglais "Human Machine Interface", est un dispositif qui permet l'interaction entre un opérateur humain et le programme de la machine. Cette interaction s'effectue généralement au travers de boutons et voyants physiques, mais aussi de leurs

équivalents virtuels. Dans la suite de ce TFE, ces deux types de solutions seront désignées respectivement en tant que HMI hardware et HMI software.

L'intérêt d'une interface matérielle se situe principalement au niveau de la fiabilité et la simplicité. Il ne serait en effet pas prudent que, en cas de blocage du programme PLC, le bouton d'arrêt d'urgence ne soit plus fonctionnel et n'engendre plus l'arrêt des machines dans une position sécuritaire. L'utilisation d'un HMI Software est quant à elle nécessaire dans le cas de grandes applications pour lesquelles un grand nombre d'actions est à lancer et un grand nombre de statuts à surveiller. Dans une telle situation, le développement d'un HMI hardware n'est pas envisageable, car utiliser beaucoup de boutons et voyants réduirait l'intelligibilité de l'interface au détriment de son utilisateur. L'utilisation conjointe de ces solutions se justifie donc par leur complémentarité bien qu'il soit courant d'avoir recours uniquement à un HMI hardware pour de petites ou simples applications.

L'avènement de l'industrie 4.0 a donné lieu au développement de solutions de HMI très innovantes et redéfinissant certains concepts propres aux HMI Software traditionnels. Cette section n'aborde pas le sujet plus en détail puis qu'il fait l'objet du choix de la solution technique au chapitre 5.

Chapitre 5

Outils de contrôle générique - Framework

Bien que le service offert par Citius Engineering soit de réaliser des machines sur mesures en fonction du client, les solutions développées se veulent les plus modulaires possibles afin de pouvoir en réutiliser certains blocs d'un projet à l'autre. Ce réemploi de module permet d'une part d'économiser des coûts de développement et d'autres parts d'assurer une qualité constante au fil du temps en plus de la standardisation via la norme ISO 9001 et la méthode du 5S. C'est donc d'une part pour ces raisons et d'autre part pour répondre aux besoins de modularité énoncés dans le chapitre 3 que les solutions développées dans le cadre de ce TFE sont les plus génériques possible.

5.1 Programme automate

5.1.1 Origines

Lors de l'automatisation d'un procédé, il peut être tentant de définir le programme automate comme une série de règles à respecter. Si les comportements à automatiser sont assez

simples, ces besoins peuvent souvent être traduits directement en code pour répondre au besoin.

“Si tel évènement se produit, alors tel réaction doit avoir lieu.”

Le problème est que, à mesure que la taille de l’application évolue, cette liste de règles devient de plus en plus difficile à utiliser : Les liens de cause et d’effet entre les éléments finissent par se confondre, la moindre modification entraîne des changements dans tout le code et il devient finalement difficile de vérifier les règles de sécurité.

Face aux problèmes liés à cette approche naïve, il est nécessaire d’établir un cadre de résolution ou *framework* dans lequel les applications peuvent s’inscrire. Puisque tous les projets d’automatisation se ressemblent sur certains points, l’idée est de construire une base commune et abstraite réutilisable à chaque projet. Par définition, cette approche s’éloigne de la méthode naïve et simple donc l’utilisation de ces principes de programmation pourrait sembler complexifier le développement. Toutefois, après familiarisation avec le framework, cette standardisation devrait permettre de simplifier et d’accélérer la création d’applications grandes et complexes. Un autre avantage important est qu’un framework bien défini garanti une meilleure sécurité. Il est en effet à prévoir que la redéfinition systématique des éléments de sécurité donne lieu à une moins bonne fiabilité.

La création d’un framework nécessite l’utilisation de concepts abstraits de programmation. Ces concepts sont rendus disponibles par la norme CEI 1131-3, mais peinent à être adopté par les automaticiens expérimentés à cause de leur côté abstrait et de la difficulté d’accès aux formations. L’évolution des formations laisse toutefois prévoir l’adoption de ces concepts dans un futur proche : Alors que l’ancienne formation d’automaticien était proche de celle d’un électricien, la nouvelle formation est amenée à intégrer de plus en plus d’éléments de celle d’un informaticien.

5.1.2 Caractéristiques d'un projet

Il est nécessaire de lister les éléments et fonctionnalités qui reviennent fréquemment dans les projets d'automatisation afin de les intégrer à la base commune idéale. Après discussion avec les ingénieurs de Citius Engineering, les points suivants se sont distingués :

- Le framework doit permettre le déclenchement de séquences courantes comme la séquence de mise sous tension, la séquence de mise en puissance, la séquence de production, la séquence d'arrêt, la séquence de calibration et la séquence de reset (suite à un arrêt d'urgence),
- Le framework doit standardiser certaines fonctions fréquentes comme le clignotement des lampes d'une tour lumineuse et la fréquence de retentissement d'un buzzer,
- L'intégration des fonctions de communication avec des appareils extérieures doit être faite via une interface claire.

5.1.3 Choix des langages

Avant de parler de la structure du code, il convient de choisir quels langages utiliser pour développer le programme. Comme expliqué dans le chapitre 4, le choix des langages à utiliser n'est pas absolu et dépend des applications et des préférences du technicien. Afin de tirer profit au maximum des capacités de la norme IEC 61131-3, une combinaison de langages a ainsi été choisie dans laquelle le langage utilisé va dépendre de l'action à effectuer. Dans le cadre de ce TFE, ces choix se justifient également, car les langages présentés ci-dessus sont couramment employés par les automaticiens de chez Citius.

Les langages employés dans le framework sont :

- Le Sequential Function Chart, SFC, pour les séquences principales, car il permet la mise en place de tâches séquentielles facilement et une bonne visualisation de l'exécution du code,

— Le Function Block Diagram, FBD, pour les conditions de transition, car il permet une visualisation claire même pour de longues expressions logiques. Ce langage reprend ainsi les avantages du ladder sans ses inconvénients.
— Le Structured Text, ST, pour l'implémentation de fonctions complexes lorsqu'il est nécessaire comme avec les connexions TCP/IP ou le traitement de chaînes de caractères.

5.1.4 Paradigme utilisé

En informatique, plus un problème est grand et complexe, plus il existe de méthodes et d'approches de résolution possible. Ces différentes approches se différencient généralement par leur performance, leur simplicité ou encore leur modularité. Afin d'atteindre l'objectif fixé, les solutions exposées ci-dessous ont ainsi été considérées puis comparées.

Itération 1 - Approche traditionnelle

L'approche qui a été suivie par les automaticiens de Citius consiste à définir plusieurs POUs4 de type Programme dont :

— Un programme LineModes qui représente l'état de la chaîne. Parmi les états de son GRAFCET, se trouvent donc l'état d'arrêt d'urgence, l'état BzrRun durant lequel un buzzer retentit et l'état de production,
— Chaque machine est représentée par son propre programme SFC qui contient la séquence d'actions propres à son poste, mais qui s'effectue en venant consulter les états du LineModes,
— Un programme MAIN dans lequel chaque machine peut être instanciée en fonction de certains flags comme celui de l'activation du mode manuel.

L'intérêt de cette méthode est que le programme LineModes peut être directement réutilisé dans un autre projet et qu'il est donc un support très utile autour duquel articuler

le développement du projet.

Par contre, cette méthode présente des inconvénients au niveau des relations entre le LineModes et les machines et des problèmes d'encapsulation entre les éléments. Dans le cas où un bouton est appuyé afin de déclencher le mode production, il semble naturel de penser que le LineModes change d'état puis impose ce changement aux machines de la chaîne. Or, avec la solution traditionnelle, c'est aux machines individuelles qu'il advient de consulter l'état du LineModes pour décider ou non de faire évoluer leur propre GRAFCET. Idéalement, il faudrait renverser ce rapport et permettre au LineModes d'être le maître qui envoie les requêtes aux machines. Le problème lié à l'absence d'encapsulation provient du faite qu'il n'existe pas de séparation claire entre les variables internes et externes de chaque machine. Il est ainsi possible qu'une machine dépende du fonctionnement d'une autre voire l'influence sans son consentement et qu'il soit nécessaire d'analyser tout le code pour s'en rendre compte. Dans le cas d'un projet réalisé par plusieurs personnes ce genre de comportement peut rapidement devenir problématique.

Itération 2 - Approche orientée-objet

Au vu des inconvénients de la solution traditionnelle et, puisque la norme encourage ce paradigme de programmation, une solution orientée-objet a été étudiée et formulée comme suit :

- Un programme LineModes qui représente l'état de la chaîne. Il envoie les requêtes de changement d'état aux machines et change son propre état que lorsqu'il a reçu la confirmation que les états de machines ont bien étés atteints,
- Un bloc de fonction (ou classe) abstrait Machine est défini ainsi que des variables témoignant de l'état actuel et requis ainsi que des méthodes comme IsTurnedOn et ManualTurnOn,
- Pour chaque machine, un bloc de fonction particulier est redéfini tout en héritant

des fonctionnalités du bloc de fonction Machine. Les différentes méthodes sont alors redéfinies en fonction des spécificités de chaque machine. De plus, chacune d'elles gère l'accès à ses propres entrées et sorties à l'aide du mot-clef EXTERNAL qui permet l'accès aux variables globales.

Cette solution est plus abstraite, mais permet l'encapsulation des variables qui ne sont dès lors plus d'accessibles qu'à l'aide de méthode Get/Set. Par exemple, pour savoir si un voyant est activé, il ne faudra plus consulter la valeur booléenne de la sortie O_Voyant mais plutôt utiliser la commande Voyant.IsTurnedOn(). Avec cette approche, le passage d'un état à l'autre peut par ailleurs être piloté par le LineModes, car les états de chaque machine n'évoluent pas tant qu'une requête n'a pas été reçue. Comme, à chaque changement d'état du LineModes, une requête doit être envoyée puis une confirmation reçue, la gestion des machines se fait à l'aide d'une liste de pointers sur machine sur laquelle il suffit de boucler pour mettre à jour les états requis par la ligne. Ainsi, l'ajout d'une ou de dix machines supplémentaires ne change en rien les conditions de transition du LineModes.

Toutefois, une approche purement orientée-objet impose que les machines n'interagissent entre elles qu'au moyen du programme maître, le LineModes. C'est donc à lui qu'il incombe de gérer les règles de sécurités entre les machines comme "Ne pas allumer l'éclairage tant que le convoyeur ne fonctionne pas". Or, une gestion absolue de la logique entre les appareils peut être fastidieuse à mettre en place et difficile à relire.

Itération 3 - Approche mixte

Tirant profit d'une part des opportunités de l'approche orientée-objet et d'autres parts du sens pratique de l'approche traditionnelle, une approche mixte a ainsi été développée pour contrôler les machines. Cette approche mixte implémente les mêmes éléments que pour l'approche orientée-objet à la différence que les interactions entre les machines

peuvent s'effectuer localement. L'accès aux méthodes des machines peut donc se faire soit au travers de liste de pointeurs sur machine, soit au travers de l'identifiant des éléments référencés dans cette même ligne. Des commandes comme setAllReqStates(Start, Machines.ListPtMachines) et Machines.Conveyor.setReqState(Start) sont ainsi toutes les deux valides et permettent respectivement de changer les états requis de toute la ligne à "Start" ou juste celui du convoyeur.

Une modification encore possible de cette approche serait le passage en argument de toutes les entrées et sorties et permettrait donc d'instancier un grand nombre de fois une même machine. Le problème de cette modification est qu'elle délocaliserait l'endroit dans le code où les variables d'entrées et sorties sont définies. Or les méthodes actuelles permettent une grande lisibilité donc le gain en termes de mémoire ne justifie pas toujours la modification.

5.1.5 Structure du framework

La Figure 5.1 montre la structure suivant laquelle les différents fichiers d'une application peuvent être répartis. Au sommet de l'arborescence, se trouvent notamment les dossiers

- Data Types qui contient tous les types définis par l'utilisateur comme les énumérations et les structures,
- Global Variables qui contient les listes de variables globales qui pourront être accédées dans tout le programme.
- POUs qui contient toutes les unités organisationnelles de programmation.

Le dossier POUs est à son tour divisé entre les composants Communs, i.e. qui sont semblables d'un projet à l'autre, les composants HMI liés à l'interaction homme-machine, et les Machines qui sont définies spécifiquement en fonction du projet.

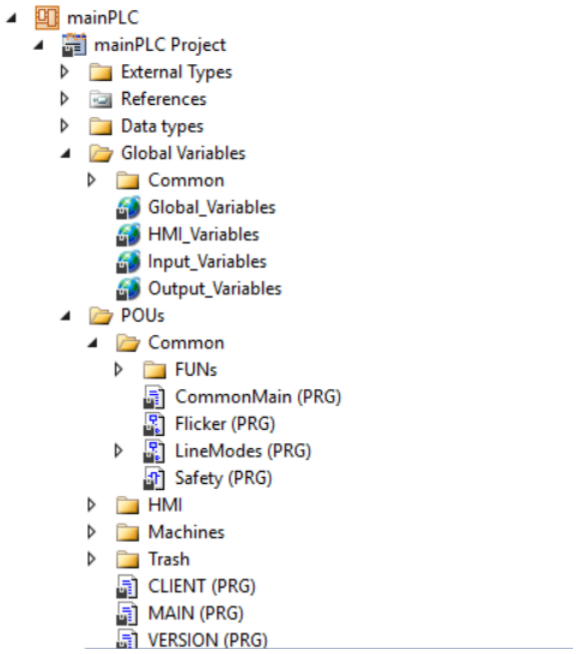


FIGURE 5.1 – Structure du framework pour une application en TwinCAT3.

5.1.5.1 Composants principaux du framework

LineModes Comme expliqué dans la section sur le choix du paradigme, le programme LineModes représente l'état général de la ligne. C'est ce composant qui ordonne les changements d'état en fonction notamment des requêtes du HMI. La Figure 5.2 montre que le programme commence dans l'état d'arrêt d'urgence puis arrive dans le mode Waiting d'où il peut transiter vers les différentes séquences. En plus des requêtes du HMI, les transitions nécessitent parfois des flags liés aux actions précédentes chaîne. De cette manière, il est possible d'implémenter le Guide d'Étude des Modes de Marche et d'Arrêt de Citius [1] même si le GRAFCET présente les séquences sur un même niveau. Un exemple de conditions de transition est présenté à la Figure 5.3.

Machine Le bloc de fonction Machine peut être assimilable à une classe abstraite dans la mesure où il n'est jamais instancié directement. La Figure 5.4 présente quelques lignes du bloc Machine. Il y est possible d'y voir la déclaration des états requis et actuel ainsi que

1. Présenté en annexe

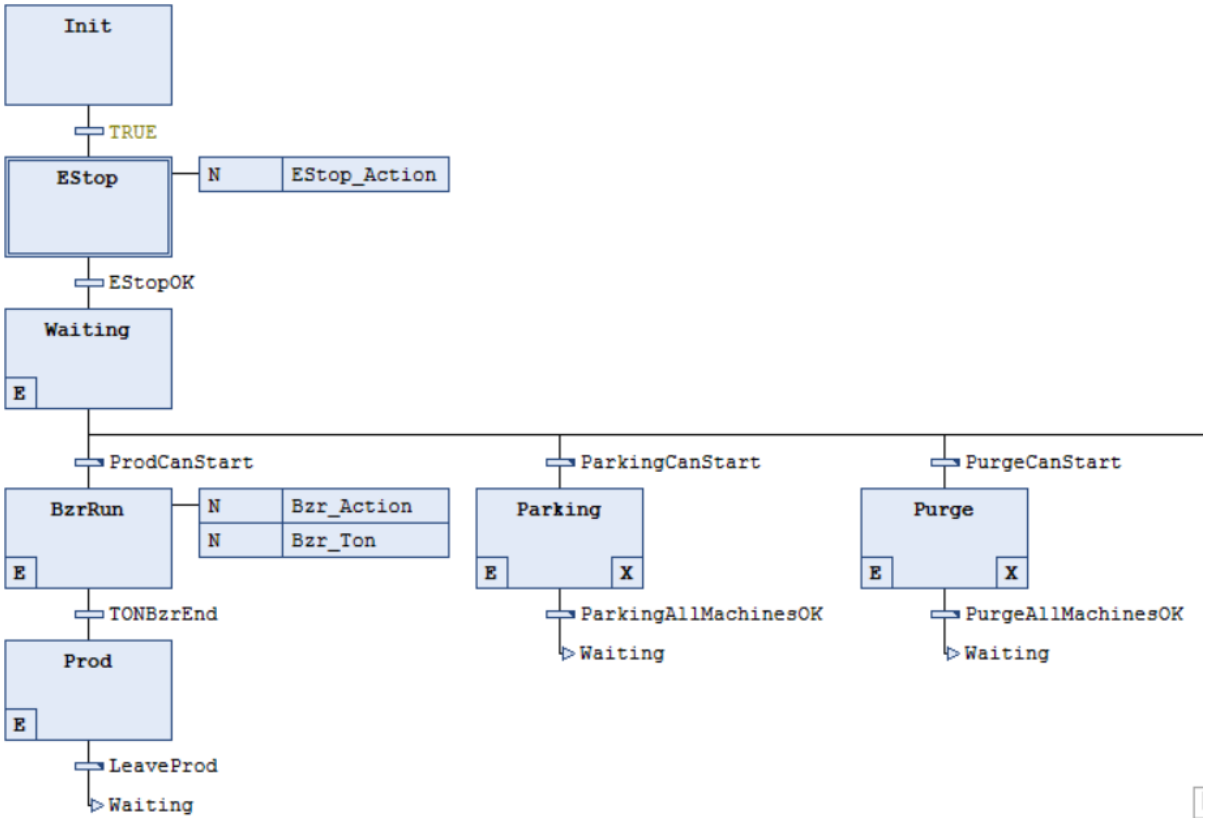


FIGURE 5.2 – Représentation partielle du programme LineModes écrit en Sequential Function Chart avec TwinCAT3.

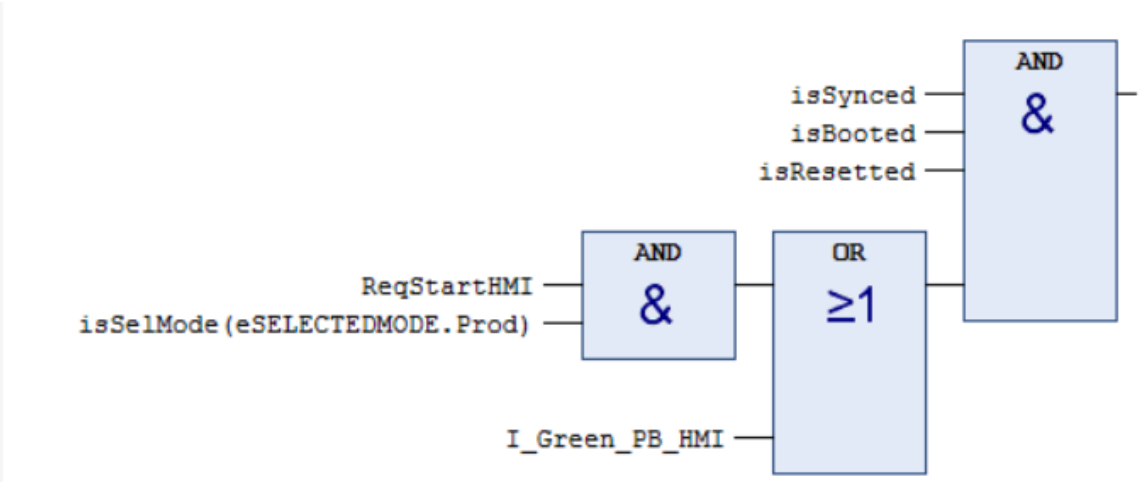


FIGURE 5.3 – Condition de transition pour le démarrage de la production écrit en Function Bloc Diagram avec TwinCAT3.

les flags témoignant des précédentes actions effectuées. Même si certaines machines n'ont sans doute pas besoin d'autant de flags, car une caméra n'a pas besoin d'action spécifique pour le reset d'un arrêt d'urgence, il a été décidé d'implémenter systématiquement toutes les branches. Afin de permettre un développement clair et systématique, il est ainsi préféré de définir des états inutiles plutôt que de particulariser le code pour chaque machine.

```
FUNCTION_BLOCK Machine
VAR_INPUT
    typeMachine     : eTYPEMACHINE;
    ID              : DINT;

END_VAR
VAR_OUTPUT
    stateDesc       : STRING;
    I_EStopOK       : BOOL := TRUE;
END_VAR
VAR
    reqState        : eSTATEMACHINE := eSTATEMACHINE.Init;
    currState       : eSTATEMACHINE := eSTATEMACHINE.Init;

    (* Variable related to state transitions *)
    isSetPwrOK                  : BOOL := FALSE;
    isStartingOK                : BOOL := FALSE;
    isPurgeOK                   : BOOL := FALSE;
    isResetProdOK               : BOOL := FALSE;
    isResetEndCycleOK           : BOOL := FALSE;
    isResetOK                   : BOOL := FALSE;
    isParkingOK                 : BOOL := FALSE;
```

FIGURE 5.4 – Représentation partielle du bloc Machine en Structured Text avec TwinCAT3.

Machines Le programme Machines est la partie de l'application dans laquelle les machines sont instanciées. C'est au niveau de cette l'instanciation que le reset du GRAFCET des machines peut être enclenché afin de mettre une machine dans son état d'arrêt d'urgence comme le montre la Figure 5.5. Ensuite, après calcul des adresses, la référence de chacune de ces machines est alors placée dans une liste de pointeurs sur machine.

5.1.6 Gestion des erreurs et sécurité

Il existe deux manières d'allumer une sortie de manière sécuritaire.

La première est d'élaborer une grande fonction qui calcule la valeur à imposer à la sortie

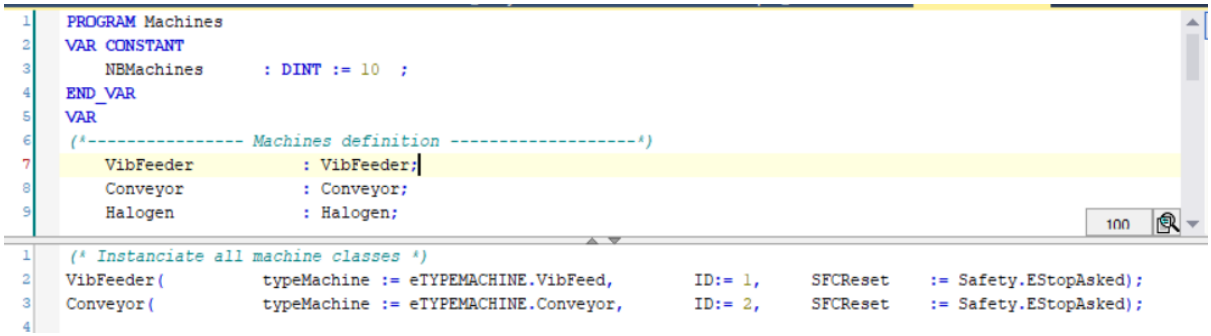


FIGURE 5.5 – Représentation partielle du programme Machines en Structured Text avec TwinCAT3.

à chaque instant. Pour se faire, cette grande fonction doit contenir les conditions traitants tous les cas de figure possibles et imaginables. La mise au point de toutes ces conditions d'un seul coup peut être difficile et des configurations oubliées peuvent survenir.

La seconde méthode consiste à changer la valeur des sorties à des moments ponctuels pour lesquels les conditions de sécurités doivent être respectées. Entre deux actions de set ou de reset, les règles de sécurités ne sont pas vérifiées et la valeur de la sortie ne change normalement pas. Comme des évènements imprévus peuvent se produire et présenter un danger pour le personnel ou le matériel, il est toutefois nécessaire de s'assurer du respect des conditions de sécurité. Pour ce faire, un programme peut être dédié à la sécurité et surveiller l'état de ces conditions à chaque cycle. C'est deuxième cette méthode qui est implémentée dans le framework, car elle convient mieux au paradigme développé dans ce projet.

Concrètement, un programme Safety a ainsi été développé et définit des flags qui sont recalculés à chaque cycle. Si un de ces flags devient vrai, il est alors possible de mettre en état d'arrêt d'urgence les machines qui posent problème. Ce reset du GRAFCET est une fonctionnalité offerte par le langage SFC et s'utilise à l'instanciation de la machine comme le montre la Figure 5.5. Les conditions de sécurité au set ou au reset de la sortie sont alors très simples et directement traduites du besoin. La Figure 5.6 montre comment une condition de sécurité, visant à s'assurer qu'un convoyeur ne soit pas brûlé par une lampe, peut être prise en compte dans l'application.

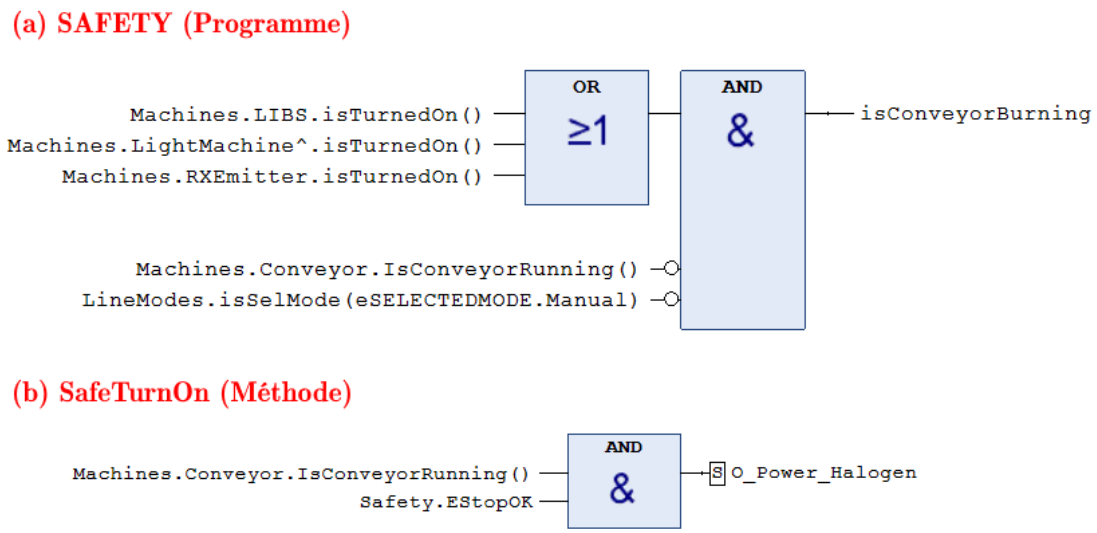


FIGURE 5.6 – Conditions de sécurité vérifiées (a) à chaque cycle via le programme de Safety et (b) à l'allumage d'une sortie via la condition ponctuelle.

5.2 HMI

Dans le chapitre 4 sur les différents composants d'un automate, l'objectif d'un Human Machine Interface est rappelé : comme son nom le laisse suggérer, il est de permettre à un opérateur d'interagir et influencer le déroulement d'un programme machine. Pour se faire, l'automaticien peut décider d'implémenter une interface matérielle -avec des boutons et des voyants-, une interface virtuelle -au moyen d'un écran tactile-, ou une combinaison des deux. Puisque la complexité du HMI hardware est limitée par le nombre d'entrées sorties et l'ergonomie d'utilisation, son développement est généralement assez similaire d'un projet à l'autre et ne présente pas de défi spécifique. Par contre, à l'ère d'une industrie 4.0 connectée et de l'uniformisation des langages automates, le développement d'un HMI software simple, multi-plateforme et réutilisable représente un enjeu majeur.

C'est afin de répondre à ces besoins que le deuxième aspect du Framework concerne le développement d'un HMI software.

5.2.1 Solutions techniques

5.2.1.1 HMI Propriétaire

La solution la plus directe pour mettre au point un HMI est bien souvent de passer par le module HMI du fabricant. Les fabricants intègrent ainsi directement l'outil de création du HMI dans l'interface de développement et permettent de tirer pleinement profit des fonctionnalités implémentées dans l'IDE. Dans le cas de TwinCAT, l'interface natif avec le programme peut s'effectuer soit via l'outil Visualisation soit via le produit TwinCAT HMI - TE2000 & TF200.

Visualization Les fonctionnalités disponibles dans l'onglet Visualisation servent principalement à l'ingénieur lors du développement de l'application automate. Parmi ces fonctionnalités, un système simple de *Glisser-Déposer* permet de positionner un ensemble de boutons et voyants puis de les lier aux variables du programme pour permettre de simuler des entrées et déclencher des séquences. Chaque élément possède des propriétés qui sont définies à partir des variables du programme comme le représente la fenêtre Propriétés de la figure 5.7. Le changement d'état d'une variable booléenne peut ainsi changer l'état d'une lampe virtuelle. Contrairement aux autres solutions qui utilisent des handles ou tags, la Visualisation n'a pas besoin de la mise en place d'un protocole particulier pour lire et écrire des variables du programme, car la communication est prise en charge par le programme automatiquement. Il existe toutefois des inconvénients à ces solutions comme le fait d'être accessible principalement dans l'interface de développement ou le manque de fonctionnalités causé par la simplicité de l'outil. Ces raisons justifient l'utilisation d'une solution moins immédiate, mais plus aboutie.

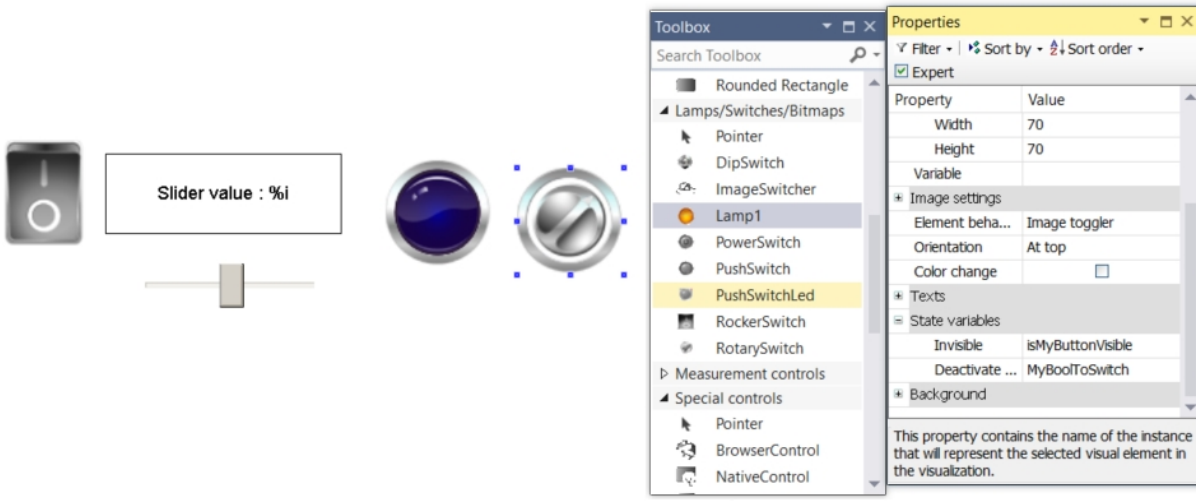


FIGURE 5.7 – Interface de développement d'une Visualization en TwinCAT3 à l'aide de Visual Studio 2013.

TwinCAT HMI L'offre de HMIs professionnels proposée par TwinCAT consiste en une application web, un serveur dédié et une implémentation facilitée par la plateforme Visual Studio. L'intérêt principal d'une solution inspirée du web est d'être multiplateforme. Cette solution s'inscrit parfaitement dans le cadre de l'industrie 4.0 puisque Beckhoff permet ainsi à ses clients de superviser leurs lignes de production au moyen d'une tablette ou d'un ordinateur. Les technologies utilisées sont le JavaScript et le HTML5 ce qui, bien que ces deux langages soient très utilisés sur le web, pourrait constituer une difficulté pour un automaticien qui réaliserait son HMI lui-même et n'aurait pas de connaissance préliminaire dans le développement web. Toutefois, l'outil de développement de HMI, au moyen de Visual Studio, permet un développement aussi simple qu'avec les Visualizations. Le désavantage principal de cette solution est qu'elle est payante au travers de licences qu'il convient d'activer dans TwinCAT3. En plus du prix, le problème de ce genre d'une interface propriétaire qui emploie des fonctionnalités propres au fabricant est qu'elle ne fonctionne qu'avec du matériel compatible. Le HMI n'est donc pas réutilisable pour un projet dans lequel une autre marque d'ordinateur industriel est utilisée.

5.2.1.2 HMI Universel OPC [17]

La standardisation des protocoles de communications a permis à de nouvelles marques de s'installer dans le secteur en se distinguant par le développement de HMI universel. Si un même protocole de communication est implémenté chez deux constructeurs différents, un même HMI peut alors être réutilisé d'un projet à l'autre. Cela représente un gain de temps considérable pour l'automaticien et permet également une certaine cohérence fonctionnelle et esthétique entre plusieurs projets d'un même groupe d'ingénieur. Concrètement, ces solutions universelles fonctionnent à partir d'un serveur OPC auquel les applications viennent se connecter. La version améliorée et plus récente de l'OPC s'appelle OPC-UA et permet une grande flexibilité au niveau des appareils utilisés afin de s'intégrer dans un contexte d'industrie 4.0. Comme les HMI universels ne sont pas développés par les principales marques d'automates, l'implémentation se fait au moyen de logiciels dédiés qui reprennent les fonctionnalités les plus fréquentes.

L'inconvénient principal de cette solution est son prix. D'une part, les fabricants d'automates demandent souvent une licence pour pouvoir utiliser le protocole universel. D'autre part, la société ayant développé le HMI universel facture généralement la licence pour chaque appareil et le prix augmente en fonction du nombre de tags ou variables partagées. De plus, puisque les technologies sont implémentées via des outils développés par des entreprises tierces, la maintabilité des produits n'est pas spécialement assurée. Autre point négatif, au vu de la nouveauté de la technologie, une intégration dans une application déjà existante est difficile à cause du manque de littérature sur le sujet.

À titre informatif, un contact a été établi avec un représentant de chez Open Automation Software qui propose un essai d'un mois de leur plateforme de développement et un tarif évolutif en fonction du nombre de tags synchronisés. Au vu des inconvénients comparativement aux autres solutions, de la limite de temps et d'argent, cet essai n'a pas été consumé. Il reste intéressant de mentionner ces facilités d'accès au cas où il serait

nécessaire d'explorer cette solution plus en profondeur à l'avenir.

5.2.1.3 HMI sur mesure en .NET

Une solution commune, mais laborieuse pour interagir avec le programme automate consiste à développer une solution sur mesure en fonction des librairies rendues disponibles par les fabricants. Comme la plupart des constructeurs d'automates fournissent des librairies en C# et que ce langage est connu des automaticiens de chez Citius, c'est l'utilisation du framework .NET qui est considérée par la suite. [2] Le framework de Windows rassemble en effet un grand nombre de technologies pour faciliter le développement des différents niveaux d'une application tournant sous Windows. Parmi les outils qui utilisent ce cadriciel, les projets de type Windows Form Application et Windows Presentation Foundation représentent deux paradigmes de programmation très utilisés et qui sont donc expliqués ci-dessous.

Windows Form Application - WFA Les applications Windows Form se basent sur l'interface graphique -GUI-, WinForms du framework .NET. Contrairement à un programme de type processus, une application WFA n'exécute pas continuellement des instructions, mais plutôt réagit aux actions de l'utilisateur appelées évènements.

Concrètement, Visual Studio permet l'accès à un grand nombre d'éléments visuels qu'il suffit de glisser et déposer pour les ajouter dans l'application. Ces éléments sont ensuite configurés au moyen de propriétés qui correspondent à des variables présentes dans un programme sous-jacent en C#. Les définitions implicites effectuées par Visual Studio font la spécificité des Windows Form Application. D'un côté, elles permettent simplifier grandement le développement et font gagner du temps au développeur D'un autre coté, ce sont également ces mécanismes cachés qui font que les WFA ne se prêtent pas aux projets de grande envergure. Par exemple, puisque certains paramètres d'un bouton ont

2. Il est à noter que des librairies en C++ sont souvent développées également.

étés remplis automatiquement à leur création, il est souvent nécessaire de repartir de zéro pour dupliquer un bouton. La complexité d'une application moderne peut dès lors augmenter rapidement si elle utilise plusieurs pages contenant beaucoup de boutons et voyants.

Un exemple d'implémentation est donné à la section 6.

Windows Presentation Foundation Suite aux difficultés énoncées lors du développement des applications WinForms, une nouvelle interface a été introduite avec la sortie du Microsoft .NET 3.0., la Windows Presentation Foundation.

Dans cette norme, la gestion des éléments visuels ne s'effectue plus au moyen d'un système de glisser-déposer, mais plutôt avec un langage descriptif appelé XAML. Ce langage, assez comparable au HTML, permet d'instancier des objets graphiques puis de leur attribuer certaines propriétés au moyen de tags. Ces déclarations explicites sont certes plus difficiles à prendre en main, mais offrent par contre bien plus de fonctionnalités et souplesses. En ce qui concerne les performances et, contrairement aux applications WFA, l'interface visuelle est vectorielle et permet donc à l'affichage des éléments graphiques de s'adapter en fonction de la taille de l'écran sans se pixelliser.

Un autre avantage de cette solution est la possibilité de décomposer les différents aspects du développement entre le côté graphique et fonctionnel. L'architecture utilisée s'appelle alors le Modèle-Vue-Vue Modèle (MVVM). Le langage XAML permet le DataBinding qui lie des variables de la Vue au Vue Modèle. Les variables du Vue Modèle sont quant à elles fournies en C# par le Modèle qui s'occupe notamment des connexions à l'automate.

Bien que ce paradigme soit moins accessible aux premiers abords, il est bien plus adapté aux applications modernes et complexes. Il permet en effet de répartir les différents postes du développement avec d'une part la logique et d'autre part l'interface visuelle. Cette séparation du code permet ainsi une grande réusabilité du code puisque, pour permettre la

compatibilité du HMI avec un nouvel automate, il suffit d'adapter le fichier du modèle chargé de la connexion. Si un grand nombre d'applications similaire doit être produit, un *thème* peut alors être développé une seule fois et être réemployé entre chaque projet pour assurer une certaine harmonie et cohérence visuelle. Malgré la maturité du ce paradigme, de nombreux modules sont encore en cours de développement et permettent notamment d'augmenter la compatibilité des applications au moyen d'outils comme Xamarin ou CSHTML5.

Un exemple d'implémentation suivant cette architecture est donné à la section 6.

5.2.1.4 Choix de la solution

Les principales caractéristiques qui différencient les différentes solutions techniques sont leur prix, leur modularité, leur facilité de développement et leur maturité industrielle. La table 5.1 reprend les différentes technologies explorées et y attribue une appréciation pour chaque critère. Il y est montré que chaque technologie possède ses propres avantages et inconvénient et ainsi qu'il est difficile de la comparer de manière absolue : la solution idéale dépend de l'application.

Tableau 5.1 – Comparaison des solutions techniques envisagées pour la réalisation d'un Human Machine Interface.

Nom de la solution	Coût faible	Modularité	Facilité	Fiabilité	Fonctionnalité
Visualization (Propriétaire Beckhoff)	+++	-	+++	-	-
TwinCAT HMI (Propriétaire Beckhoff)	—	-	+++	+++	++
HMI Universel (OPC-UA)	-	++	+++	++	++
WFA (.NET)	+++	-	+++	+	-
WPF (.NET)	+++	+++	–	+++	+++

Dans le cadre de ce projet, puisque toutes ces solutions permettent d'atteindre l'objectif principal, i.e. interagir avec le programme, il convient de les comparer selon ces critères tout en tenant compte des besoins secondaires du client explicité au chapitre 2. Puisqu'il est très important que le travail effectué pour ce projet puisse être réutilisé facilement et que le coût est à minimiser, il convient de réaliser un **HMI avec la technologie WPF** tout en s'appuyant sur les HMI déjà implémentés chez Citius en WFA pour en faciliter l'intégration. Au vu de ses caractéristiques prometteuses, il est à prévoir que les HMI universels gagnent en maturité industrielle et s'imposent de plus en plus sur le marché.

Chapitre 6

Portage du Framework

Ce chapitre est la convergence des précédents puisqu'il a pour objectif d'appliquer le framework défini au Chapitre 5 à l'aide des subtilités de la norme décrite au Chapitre 4 pour répondre aux besoins formulés au Chapitre 3. Le portage du framework consiste alors à appliquer les concepts abstraits pour les confronter à une situation réelle.

6.1 Code automate

6.1.1 Machines

La première étape pour l'adaptation du framework consiste à créer et connecter au programme toutes les machines de la ligne. Comme expliqué dans le chapitre précédent, il est nécessaire que chaque machine créée hérite des fonctionnalités du bloc de fonction Machine. Puisqu'il n'existe par contre pas de mécanisme d'héritage direct pour le code en SFC, une première machine, la plus complète possible est définie puis le code est réutilisé autant de fois que nécessaire.

À titre d'exemple, le code SFC du robot est présenté à la Figure 6.1. Ce code est as-

sez similaire à celui du GRAFCET du programme LineModes à l'exception des états Nom_de_l_etat_OK qui se justifie par la section ci-dessous.

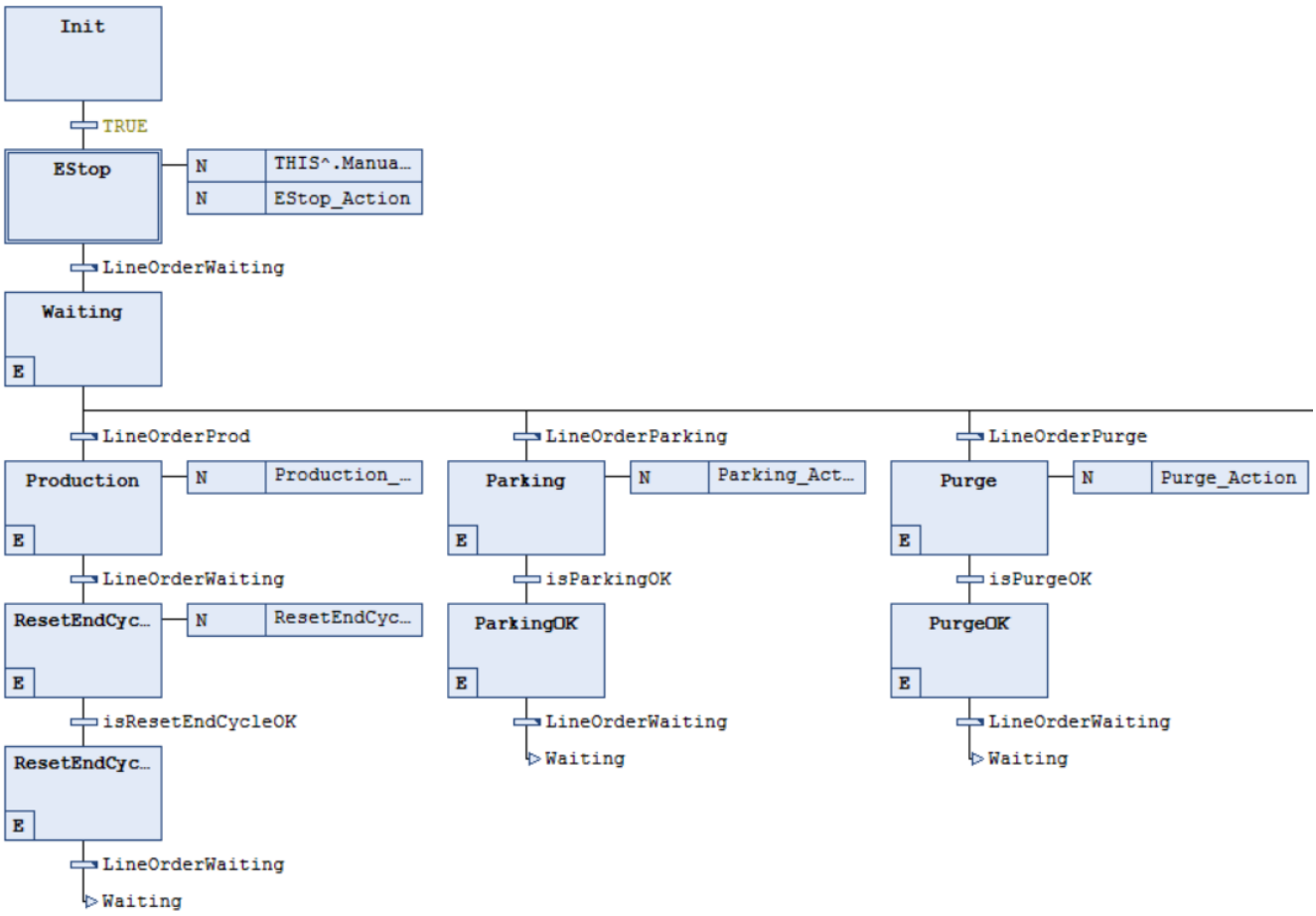


FIGURE 6.1 – Représentation partielle du bloc de fonction LineModes écrit en Sequential Function Chart avec TwinCAT3.

6.1.1.1 Communication avec le GRAFCET de ligne

Comme le déroulement du code d'une machine est intimement lié aux ordres communiqués par le programme LineModes, la Figure 6.2 en explicite le fonctionnement pour l'étape de démarrage :

1. La séquence commence lorsque la condition de transition BootCanStart est vérifiée.
2. L'action d'entrée de l'étape Starting consiste à changer tous les états requis à la valeur Starting et à resetter la valeur du flag ReqTreated, car la requête n'a pas

encore été traitée à ce stade de l'exécution [1].

3. Lorsque la condition de transition LineOrderStarting est vraie, i.e. que l'état requis vaut Starting, l'halogène rentre dans l'étape Starting.
4. La machine change son état actuel à Starting
5. Tant que la machine est dans l'état Starting, sa méthode SafeTurnOn est appelée et a pour effet d'allumer de manière sécuritaire le système d'éclairage. L'éclairage ne sera ainsi pas allumé tant que le convoyeur ne fonctionne pas. De plus, puisque le niveau de pleine puissance de l'éclairage n'est pas atteint immédiatement, l'action Starting_Action contient un timer qui laisse à l'éclairage le temps de chauffer.
6. Lorsque l'éclairage est allumé et depuis assez longtemps, la Starting_Action set la condition de transition isStartingOK à TRUE et la machine peut passer à l'étape StartingOK
7. En entrant dans l'étape StartingOK, la machine change son état actuel à Ready et informe la ligne que la requête a bien été traitée.
8. Si la valeur de ReqTreated vaut True et que toutes les machines sont en mode Ready, la condition est vérifiée et la ligne peut quitter l'étape Starting avec succès.

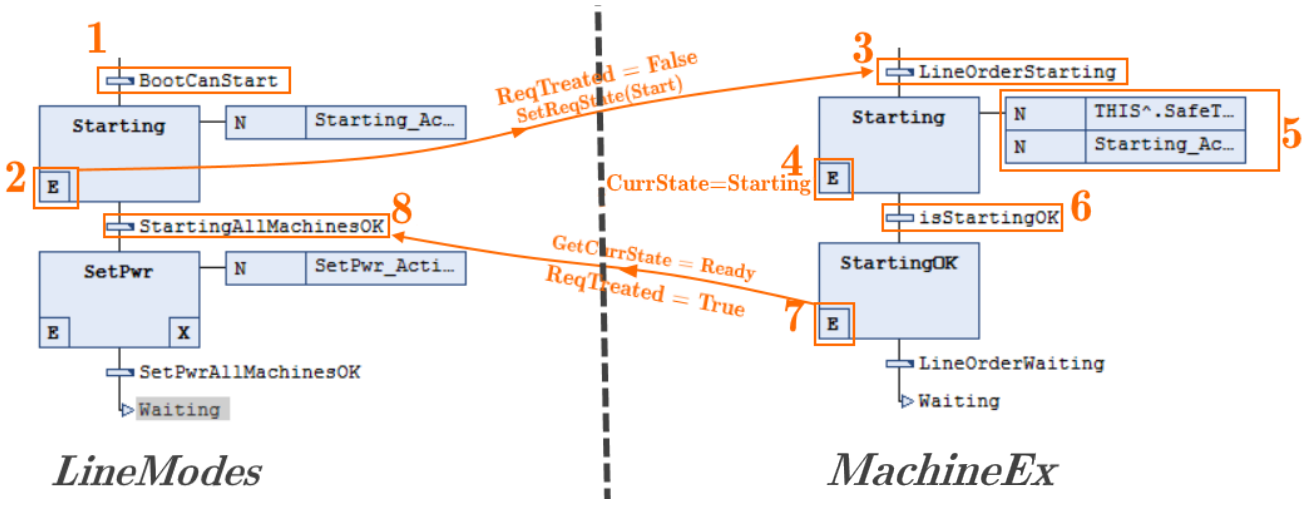


FIGURE 6.2 – Séquence de communication entre les GRAFCET d'une machine et de LineModes.

1. Il s'agit d'un mécanisme de sécurité pour éviter que la ligne ne continue son exécution sans s'arrêter en Starting si les machines étaient déjà dans l'état Ready, qui revient plusieurs fois sur le programme.

6.1.1.2 Comportements spécifiques

Maintenant que le mode d'action et le principe d'incorporation des machines ont étés clarifiés, les particularités implémentées peuvent être décrites sans nécessairement entrer dans les détails du code pour chaque machine :

— L'alimentation vibrante n'est activée que dans la phase de production.

— Le convoyeur doit être activé dans toutes les situations où l'éclairage est allumé.

— L'éclairage ne peut pas être allumé sans le convoyeur. Il a de plus un temps de chauffe non nul.

— La caméra spectrale, la caméra 3D, le système LIBS, le système à rayons X et le système de refroidissement ont un temps d'allumage non nul, mais qui n'est pas contrôlé par le PLC.

— Avant d'utiliser les robots, il est nécessaire d'activer les moteurs à l'aide d'une commande RAPID [2]. De même des commandes RAPID peuvent être initiées pour arrêter le robot, lancer le programme du robot et lancer la séquence de reset du robot.

6.1.2 Synchronisation des pulses

Les actions du PLC lors de la séquence de synchronisation consistent à attendre un message du PC d'acquisition puis à relancer le passage des pulses de l'encodeur pour que toutes les machines travaillent avec une même base.

6.1.2.1 Connexion TCP/IP

Puisqu'il a été décidé que les communications entre le PC et le PLC s'effectueraient au moyen d'une connexion TCP/IP, la mise en place d'un client fonctionnant de pair avec

2. Le robot peut être configuré pour déclencher certaines actions en fonction des entrées de l'automate du robot connectées aux sorties du PLC Bekchoff

le serveur du PC est nécessaire. Les connexions TCP/IP sont un type de connexion pour lesquelles les participants doivent accepter la connexion, au travers d'une séquence appelée handshaking, avant d'échanger des données. Ces connexions ont l'avantage de notifier les utilisateurs lorsqu'une perte de paquets a lieu et de respecter l'ordre d'émission. Ce type de connexion est très courant au vu de ses performances, mais n'est pas simple à mettre en place. Heureusement, Beckhoff, a mis au point une série de blocs de fonction communs et utilisables dans TwinCAT3 en ajoutant le module TS6310. [9]

6.1.2.2 Protocole de communication

Afin de pouvoir interpréter les messages échangés, il est nécessaire de travailler sur une base commune. Un protocole a ainsi été établi puis proposé aux informaticiens du GeMMe. Celui-ci est disponible à l'annexe.0.6 et ses traits principaux sont :

1. Les messages envoyés entre le PC et le PLC sont des chaînes de caractères dont les arguments sont repris entre deux tirets.
2. Chaque chaîne de caractère commence par le caractère @ et se termine par la représentation hexadécimale du chiffre 0.
3. Les informations fournies sont, par ordre d'apparition, l'ID de l'envoyeur, l'ID de destinataire, l'ID du message, la séquence auquel le message correspond, l'action à effectuer, ainsi que trois arguments réservés pour une utilité future.
4. Les messages correspondants à des actions et séquences reconnues sont repris dans la Table 3 de l'annexe .0.6.

Un exemple de message envoyé par le PC pour arrêter les pulses dans la séquence de synchronisation : **@1-10-103-sync-stoptrig- - - - -**
x00. Un inconvénient de ce protocole est d'employer des chaînes de caractère ce qui n'est pas optimal, car la taille des messages pourrait être réduite avec des codes binaires. Toutefois, au vu de la fréquence, de la taille des messages et de la puissance de traitement des appareils utilisés, le gain en efficacité ne justifie pas la perte en compréhensibilité.

6.1.3 Gestion des messages

Puisque, pour avoir un client fiable, il est nécessaire de traverser les étapes de création du handle puis de gérer l'apparition d'erreurs comme les reconnections, il a été décidé d'adapter un code déjà fonctionnel aux besoins du problème. La configuration des données relatives à la connexion (port et adresse ip) se renseigne dans le code `FB_Client` extrait des exemples de Beckhoff.

L'analyse des messages reçus se fait dans le bloc `FB_ClientConnection` qui extrait les chaînes de caractères se terminant par 0 puis les ajoute dans une file. Les modifications apportées dans le cadre de ce projet concernent le traitement des messages de cette file assuré dans `FB_Application`. Dans cette application, un parseur de message a été développé et permet d'extraire les informations de la chaîne de caractère pour les charger dans une structure de type stFRAMETCP. Lorsqu'un nouveau frame a été reçu, la phase à laquelle il fait partie est d'abord lue puis en fonction de l'action requise un flag est levé dans le programme `LineModes` qui pourra influencer le cours du programme comme le montre la Figure 6.3

```
IF newRcvFrame.receiverID = PLC_ID THEN (* the message is addresed t
    CASE newRcvFrame.senderID OF

        1: (* message coming from the PC *)
            IF newRcvFrame.phase = 'sync' THEN
                (* Actions related to the synchronization phase *)
                IF newRcvFrame.reqAction = 'stoptrig' THEN
                    ConfigLine.selectedMode := eSELECTEDMODE.Sync;
                    ReqStartFromPC := TRUE;
                ELSIF newRcvFrame.reqAction = 'enabletrig' THEN
                    LineModes.PCReadySync := TRUE;
                END_IF
```

FIGURE 6.3 – Traitement d'un nouveau FrameTCP reçu dans le code de `FB_Application`.

L'envoi de messages du PLC vers le PC s'effectue à l'aide des mêmes blocs de fonctions que pour la réception des messages. Lorsque le `LineModes` se trouve dans un état pour lequel il doit communiquer avec le PC, il remplit les champs du FrameTCP à envoyer puis

lève un flag newFrameToSend comme le montre la Figure 6.4. Comme l'état de ce flag est surveillé par `FB_Application`, le contenu du FrameTCP va alors être converti en chaîne de caractères structurée puis être envoyé par le bloc de fonction `FB_ClientConnection`.

```
IF newFrameToSend = FALSE AND requestSent = FALSE THEN  // me
    stFrameToSendTCPIP.receiverID       := 1; // the pc
    stFrameToSendTCPIP.phase            := 'sync';
    stFrameToSendTCPIP.reqAction        := 'enabletrigok';
    stFrameToSendTCPIP.argument1        := '';
    stFrameToSendTCPIP.argument2        := '';
    stFrameToSendTCPIP.argument3        := '';
    newFrameToSend := TRUE;
    requestSent := TRUE;
END_IF
```

FIGURE 6.4 – Chargement d'un nouveau FrameTCP à envoyer dans le code de `LineModes`.

6.1.4 Séquence

Après la mise en place des outils décrits ci-dessus, l'implémentation de la séquence de synchronisation est simple et traduite directement des besoins du client expliqués dans le Chapitre 3.

La séquence est initiée par le PC. Le PLC coupe alors les sorties de l'encodeur à l'aide d'une sortie reliée à un relais puis en notifie le PC. Dès que le PC a correctement réinitialisé ses équipements, il avertit le PLC qu'il peut redémarrer les pulses de l'encodeur. Ce redémarrage doit avoir lieu lorsque le front montant du pulse de page est détecté au moyen d'un bloc de fonction prévu à cet effet `R_TRIG`. Dès que l'encodeur a été allumé avec succès, le PLC notifie l'action au PC puis quitte la séquence de synchronisation. La Figure 6.5 représente un schéma de la séquence PLC avec les communications.

Au vu de l'importance de l'étape de synchronisation, un serveur TCP/IP en Python a été simulé sur un ordinateur distant. L'ordinateur a ensuite été connecté à la version émulée

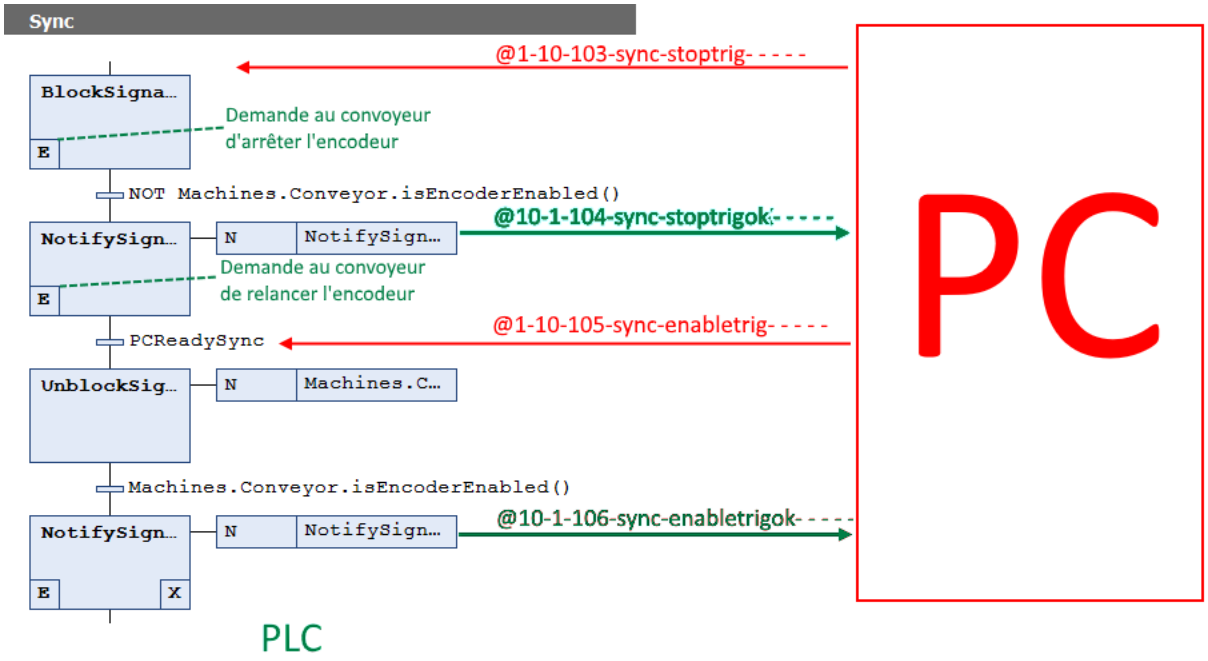


FIGURE 6.5 – Séquence de la macro-étape Synchronization dans le code du bloc de fonction `LineModes`.

du PLC au travers d'un switch réseau. Ce test a ainsi permis de valider le fonctionnement des échanges avec le serveur TCP/IP et du déroulement de la séquence sans avoir à tester sur le matériel du client.

6.2 Sécurité

À ce stade du développement du prototype, la seule requête de sécurité concerne l'éclairage du tapis roulant qui a déjà été présenté au chapitre précédent comme exemple. Lorsque le besoin d'autres règles se fera ressentir, il suffira d'ajouter des lignes de réseaux au programme Safety et de les référencer au moment de l'instanciation des machines concernées dans le programme Machines.

La gestion de l'arrêt d'urgence se gère de la même manière que les règles particulières à la différence de s'appliquer à l'ensemble des machines dangereuses. L'absence de signal de la part du bouton EStop a donc pour conséquence de resetter le GRAFCET de ces

machines à leur étape initiale, l'arrêt d'urgence. L'exemple du code de machine montré à la Figure 6.1 permet de remarquer que dans l'état EStop, l'action ManualTurnOff() y est appelée. Dans le cas du robot, cela correspond à l'allumage d'une sortie lue par le code RAPID du robot et provoquant l'arrêt de l'instruction en cours.

6.3 HMI

Lors du stage effectué chez Citius Engineering, la réusabilité des solutions développées a toujours été une des finalités. C'est dans cette optique que, avant de considérer des solutions plus exotiques, il a d'abord été décidé d'étudier en profondeur l'implémentation des solutions actuelles. De cette manière, les améliorations proposées peuvent ainsi facilement enrichir les solutions existantes et ne nécessitent pas de nouvelles formations du personnel. Un exemple simple a donc été développé pour se familiariser avec la technologie WFA et la connexion à l'automate puis un programme plus complexe avec l'approche WPF pour la solution définitive.

6.3.1 Connexion à l'automate via le protocole ADS

La seule solution pour se connecter à l'automate sans utiliser de licence supplémentaire consiste à utiliser le protocole créé par Beckhoff et appelé Automation Device Specification ou ADS. Ce protocole est présent à différents niveaux de l'architecture d'un automate Beckhoff comme en témoigne la Figure 6.6 et permet une communication directe ou basée sur des socket IP au moyen d'un routeur de messages. C'est une couche supérieure au TCP/IP et au UDP/IP et dont le contenu des messages contient les éléments suivants[6] :

— Le AMSNetID qui référence l'adresse de la machine à laquelle est destiné le message.

— Le numéro de port qui distingue les différents éléments connectés sur une même

machine.

— L'indice du groupe qui différencie les données dans un même port.

— Le décalage de l'indice qui indique à quel endroit commencer la lecture ou l'écriture.

— La longueur des données.

— Le numéro de port TCP qui vaut toujours 48898 ou 0xBF02 pour l'ADS.

FIGURE 6.6 – Représentation du rôle du protocole ADS dans les communications d'une application Beckhoff fictive[6].

Afin de permettre les connexions et simplifier au mieux l'utilisation du protocole avec le framework .NET, Beckhoff fournit une bibliothèque compatible avec le C# appelée `TwinCAT.Ads.dll`. Cette bibliothèque implémente des fonctions qui gèrent notamment la connexion/déconnexion au serveur, l'écriture et la lecture des variables comme expliqué dans la documentation en ligne[8].

6.3.2 Implémentation d'une interface simple en WFA

Puisque les documentations concernant l'intégration d'un HMI avec le protocole ADS ne concernent que les applications WinForms, il est normal de commencer par cette solution. Après création d'un projet Windows Form, un exemple d'application peut être défini comme présenté à la Figure 6.7. Cet exemple est structuré comme suit :

— Le rendu final de l'application est défini en faisant glisser-déposer des boutons et cadres de texte sur le fichier Form1.cs. Les fichiers Form1.Designer.cs et Form1.resx se génèrent alors automatiquement.

— Le fichier CommunicationManager.cs contient l'interface de connexion au PLC et sert aussi à déclencher les évènements comme le rafraîchissement des données affichées ou l'action d'un bouton.

— L'onglet Références dans lequel est ajoutée la libraire TwinCAT.Ads

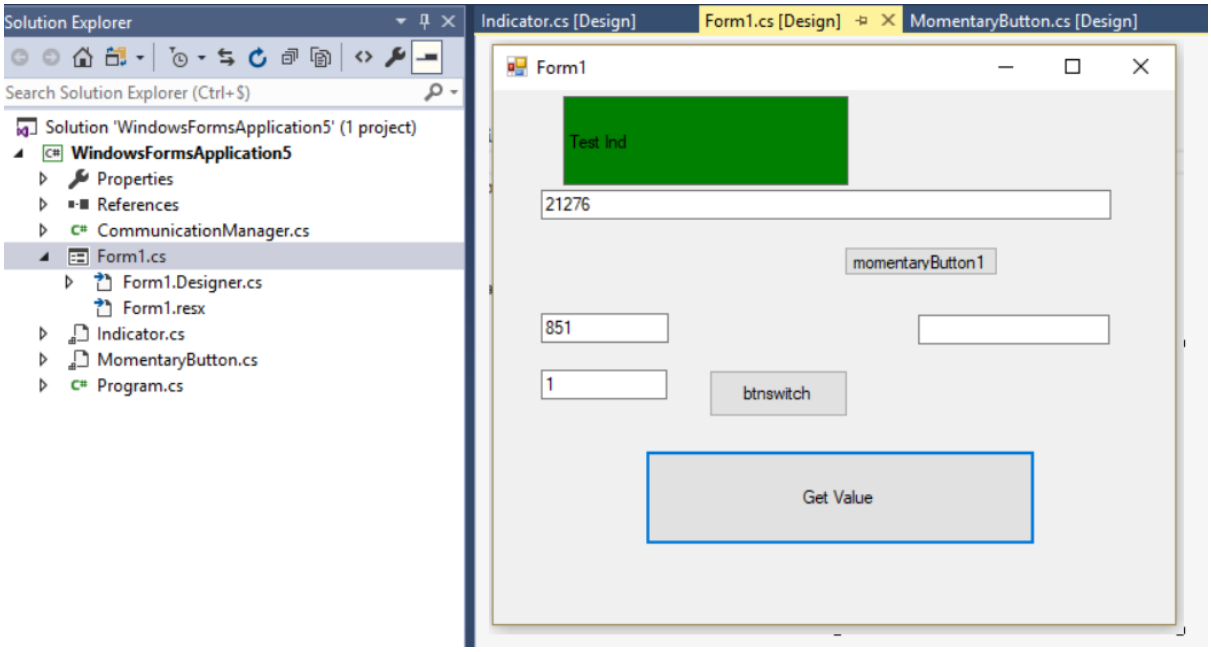


FIGURE 6.7 – Structure et rendu d'une application WFA.

La simplicité du WFA est son principal intérêt, mais aussi son défaut, car ses limites sont très rapidement atteintes. Cet exemple a pour but de tester le fonctionnement de chacune des fonctionnalités à implémenter pour avoir un HMI complet. Cette application implémente ainsi les connexions et reconnexions avec l'automate, l'écriture et la lecture des variables de l'automate (booléennes et réelles) et l'adaptation du rendu visuel en fonction de l'état de certaines valeurs. Il est toutefois apparu que la complexité de l'application exploserait avec la taille de l'application. Le site internet Contact and Coil propose une solution[26] plus complexe pour régler certains des problèmes apparus. Cependant, comme le WFA n'est pas adapté pour soutenir des structures complexes, cette solution ne sera jamais idéale.

6.3.3 Connexion au PLC via une application WPF

Comme expliqué au Chapitre 5, le développement d'une application en WPF est moins immédiat, mais permet une meilleure répartition des tâches grâce à la décomposition en Modèle/ Vue/ Vue-Modèle. Pour le HMI de la cellule Pick-it, l'application est développée

autour d'un code existant[5] et très simple en WPF. L'intérêt de ce code est qu'il implémente déjà tous les mécanismes propres au paradigme Modèle/Vue/Vue-Modèle qui constituent justement un obstacle à l'adoption du WPF.

La structure de la solution finale est représentée à la Figure 6.8 dont les éléments sont répartis en fonction du modèle MVVM.

Le rôle de chaque élément consiste :

- Pour Sharp7Library, à contenir les différentes librairies permettant la connexion avec les automates.
- Pour le dossier Converters, à convertir des valeurs en vue de leur affichage. C'est ainsi que la couleur d'un voyant peut être liée à un booléen.
- Pour le dossier Designer, à définir des valeurs factices pour peupler les vues lors du design et donc faciliter le travail du designer.
- Pour le dossier Models, à définir les classes utilisées pour manipuler les données dans le programme.
- Pour le dossier ViewModels, à servir d'interface entre les vues, codées en langage XAML, et le service de connexion au PLC. À chaque vue, correspond son propre Vue-Modèle dans lequel sont définies les propriétés à lier par DataBinding[11].
- Pour le dossier Views, à définir les vues ou rendus graphiques de l'application. Puisque l'application est multipage, certaines vues sont constamment affichées comme le menu ou la bar de statut tandis que d'autres, les pages, dépendent du choix du menu. Cet arrangement entre les vues est assuré par la vue principale, MainWindow.
- Pour le fichier Bootstrapper.cs, à définir un contexte commun entre les pages et ainsi permettre le partage de données.
- Pour SimpleHMI.PlcService, à reprendre les classes implémentant les connexions vers les automates.

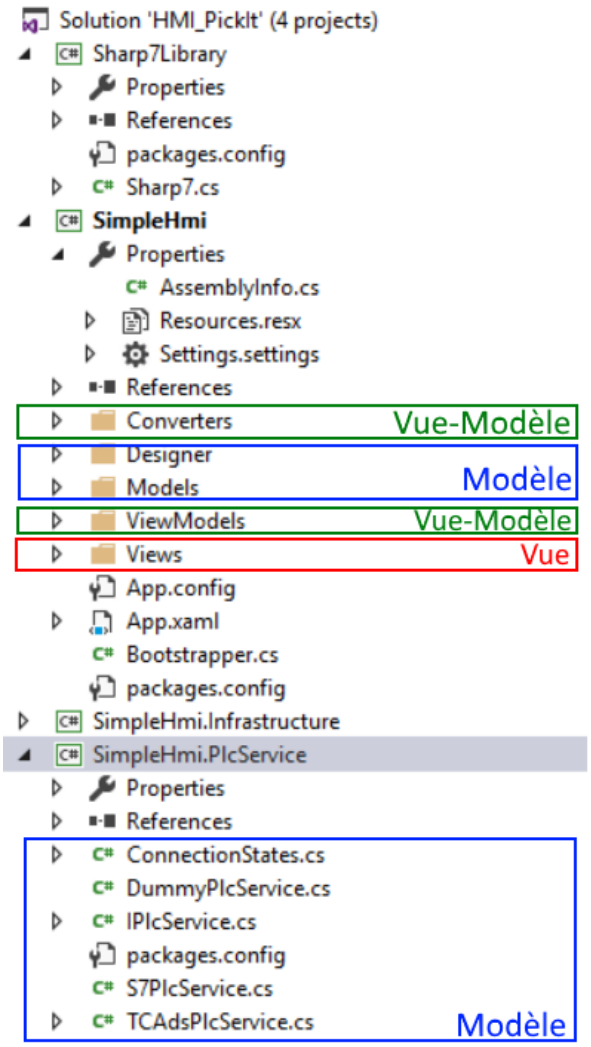


FIGURE 6.8 – Structure de l'application en Windows Presentation Foundation.

6.3.3.1 Flexibilité par rapport aux marques d'automate

Afin d'assurer une certaine modularité dans les connexions avec les automates, la classe IPlcService a été intégrée. Il s'agit d'une interface indépendante de la marque du PLC et qui définit les méthodes à implémenter de manière abstraite. Cette interface permet une grande réusabilité entre les marques d'automate, car seul le fichier de connexion est à modifier. La classe TCAdsPlcService en est un bon exemple puisqu'elle implémente les méthodes de l'interface IPlcService, mais en employant les fonctions de la bibliothèque TwinCAT.Ads. Il est à noter que, au sein même de la classe de connexion au PLC Beckhoff,

certaines fonctions génériques y ont été développées pour améliorer la sécurité et le confort de développement comme le montre la Figure 6.9.

```
private bool WriteBool(string nameVar, bool value)
{
    if (ConnectionState == ConnectionStates.Online)
    {
        lock (_locker)
        {
            try
            {
                this._client.WriteSymbol(nameVar, value, reloadSymbolInfo: true);
                return true;
            }
            catch (AdsErrorException)
            {
                return false;
            }
        }
    }
    else
        return false;

}
```

FIGURE 6.9 – Fonctions d'écriture d'une variable booléenne dans un automate.

6.3.3.2 Aspect visuel

Un des avantages des applications WPF est de faciliter l'amélioration du design de l'application. Il est ainsi possible de développer un thème commun à une entreprise et de l'appliquer à l'ensemble des projets. Plutôt que de mettre à jour l'interface graphique de tous les projets séparément, il suffirait ainsi de mettre à jour ce fichier commun pour améliorer l'aspect de toutes les applications.

Dans le cadre de ce projet, le coté graphique ne fait pas partie des expertises valorisées donc un thème tiré du projet Mahapps.Metro est appliqué pour donner un aspect moderne à l'application comme le montre la Figure 6.10. Ce thème est appliqué dans le fichier App.xaml et peut être configuré en suivant les directives reprises sur leur site web[18].

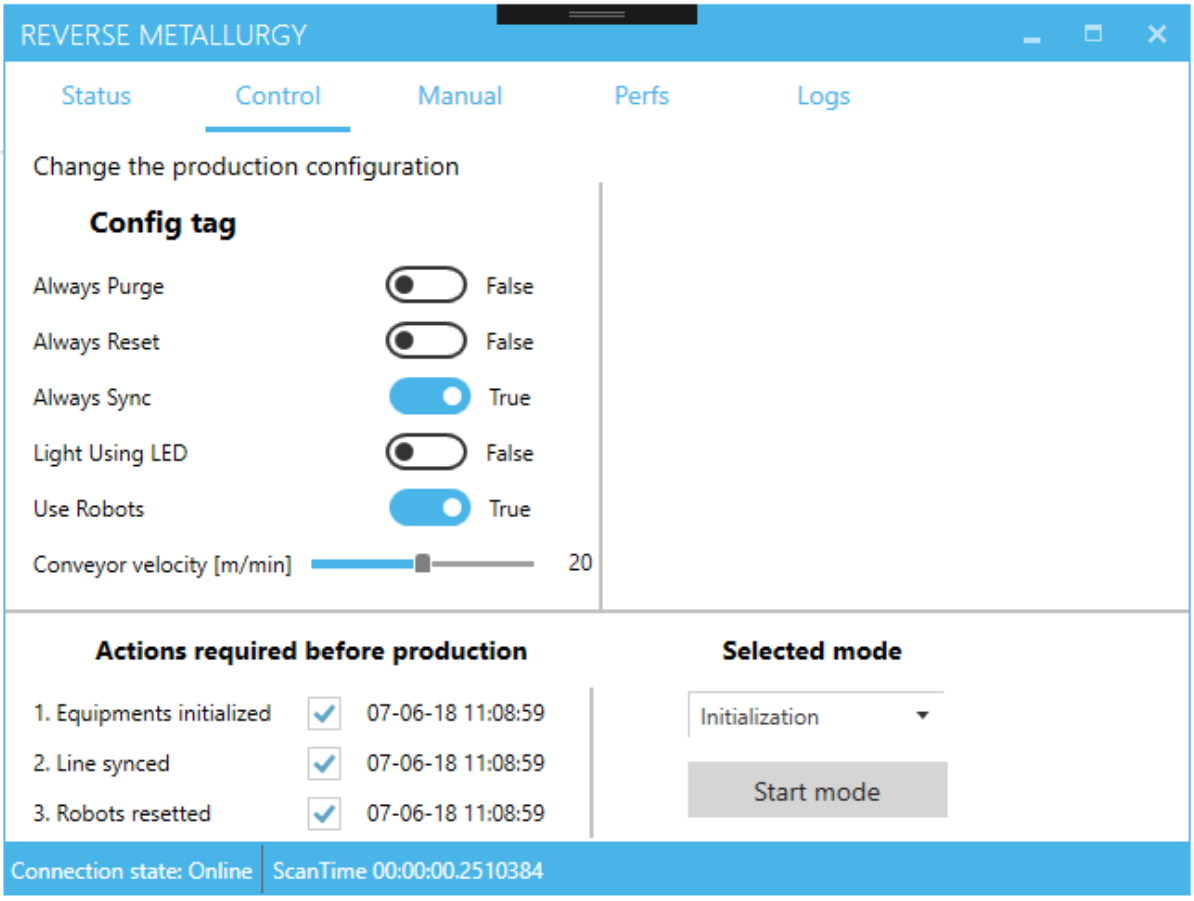


FIGURE 6.10 – Rendu de la page Control de l'application Windows Presentation Foundation.

6.3.3.3 Communication avec le programme PLC

Cette section vise à décrire comment le HMI est intégré au programme PLC en expliquant le parcours d'une variable. Cette description a également pour objectif d'illustrer les liens qui existent entre les différents modules de l'application

La Figure 6.11 représente l'enchaînement d'action qui se produit lorsque l'alimentation vibrante est allumée dans le mode manuel. Lorsque le bouton à bascule virtuel est appuyé, le changement de sa valeur va être détecté par le code du Vue-Modèle. Cet évènement va déclencher la méthode d'écriture du service PLC, le modèle, et la nouvelle valeur de la variable sera écrite dans la liste de variables globales, HMI_Variables, du PLC. Le

programme chargé de l'interface entre les variables de cette liste et les variables utilisées dans le programme va ensuite mettre à jour l'attribut manualEnabled de la machine. Ainsi, si cette variable vaut true et que la machine est en mode manuel, la méthode ManualTurnOn() sera lancée.

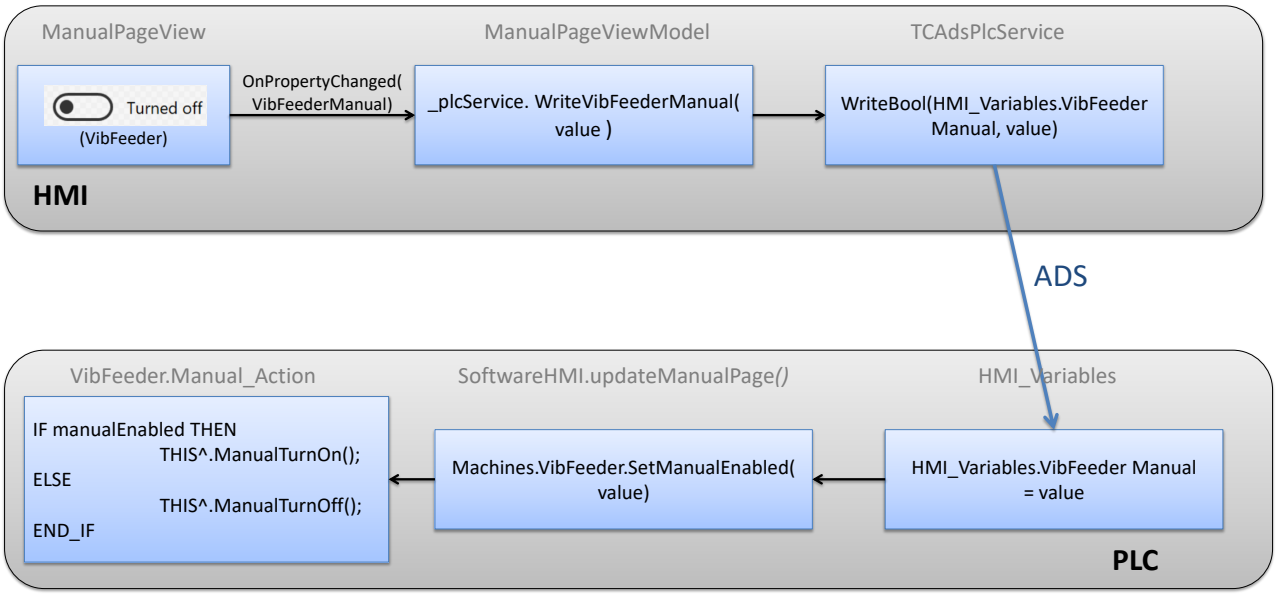


FIGURE 6.11 – Rendu de la page Control de l'application Windows Presentation Foundation.

Pour certaines variables, comme la vitesse du convoyeur, il est nécessaire d'une part de charger la valeur pour l'afficher et d'autre part de pouvoir la modifier. Sachant qu'une seule de ces actions est possible à la fois, se pose alors la question de savoir si le HMI doit écrire sa valeur dans le programme ou lire la valeur du programme pour l'afficher. L'évènement OnNavigatedTo du Vue-Modèle est utilisé pour régler ce conflit. Ainsi, lorsque la page de navigation est chargée la valeur de la vitesse actuelle du convoyeur est lue dans le programme puis affichée à l'écran. Ensuite, si le curseur est déplacé, l'évènement OnPropertyChanged déclenche simplement l'écriture de la nouvelle valeur dans le programme PLC.

Chapitre 7

Intégration et tests

L'objectif de ce chapitre est d'aborder l'intégration des solutions développées et de leurs tests sur le prototype réel.

Dans la mesure où d'autres personnes travaillent en même temps sur la ligne, un des challenges de cette intégration a été d'installer progressivement les solutions développées tout en permettant une compatibilité inverse avec les solutions précédentes.

7.1 Intégration

7.1.1 Câblage

Au commencement de l'intégration, le problème du câblage était d'être peu développé. En effet, parmi la cinquantaine d'entrées et de sorties, moins de dix seulement étaient réellement connectées. Comme cela ne faisait pas partie de l'objet de ce TFE et que cela nécessiterait énormément de temps, seules certaines machines ont pu être réellement raccordées à l'automate. À l'avenir, si une automatisation complète de la ligne est souhaitée, il sera nécessaire de câbler toutes les machines afin de pouvoir les piloter depuis

l'automate.

Dans le cadre de ce TFE, dans câbles ont donc étés tirés, dénudés puis sertis, pour effectuer le contrôle sur les machines suivantes :

- Les robots, 4 sorties ont étés attribuées pour pouvoir influencer le déroulement du code RAPID des robots. Il est à noter que les entrées des deux robots ont étés pontées.
- HMI hardware, 4 entrées ont étés attribuées pour pouvoir détecter l'utilisation des boutons poussoirs.
- La tour lumineuse, 5 sorties ont étés attribuées pour allumer les différentes lampes de la tour ainsi que le buzzer intégré.
- Le convoyeur, 4 sorties ont étés attribuées pour le contrôle du convoyeur en fonction du mode programmé de son inverter.

Afin de pouvoir recevoir et émettre tous les signaux nécessaires, des modifications dans l'armoire électrique ont étés apportées.

7.1.2 Mapping I/O

Après la mise à jour de l'automate de TwinCAT2 vers TwinCAT3, il a été nécessaire de recommencer le mapping de toutes les entrées et sorties du système. La procédure à suivre est expliquée dans la documentation de Bekchoff[7] et les fiches techniques du matériel peuvent également être nécessaires. Il y est notamment expliqué comment une entrée ou une sortie peut être associée à une variable.

Au vu des différences entre la nomenclature du schéma électrique, les adresses virtuelles attribuées par le programme et le nom des variables du programme, la Table 7.1 et Table 7.2 ont étés établies. La Table 7.1 et la Table 7.2 servent donc à clarifier les correspondances qui existent entre les références des entrées et des sorties respectivement. Dans la même optique, la Table 7.3 et la Table 7.4 servent à clarifier les correspondances entre

Tableau 7.1 – Références virtuelles et électriques des variables d'entrée du programme PLC.

Référence virtuelle			**Référence électrique**	
Nom de la variable	Terminal	Canal	Entrée	Terminal
I_White_PB_HMI	7	2	10	7
I_Blue_PB_HMI	7	3	11	7
I_Green_PB_HMI	7	4	12	7
I_Red_PB_HMI	7	5	13	7
I_Black_PB_HMI	7	6	14	7
I_Signal_Page	7	7	9	7
I_EStop	5	3	22	6
I_EStopOK	7	8	8	7

les références électriques de l'automate vers le convoyeur et le robot respectivement.

7.1.3 Configuration

Dans le cas du convoyeur, le manuel d'utilisation[22] spécifie au chapitre sur le commissioning la marche à suivre. Puisque l'appareil doit pouvoir être encore utilisé en mode manuel par les chercheurs du GeMMe, il a été décidé de choisir une configuration par défaut compatible avec le programme de la télécommande. En manoeuvrant ainsi avec l'outil de diagnostic, la configuration Terminals 0 (C00007) a pu être choisie. Il faut toutefois remarquer que son fonctionnement n'est actif que si le mode manuel de l'inverter a été quitté. Ce contrôle permet notamment le choix de la vitesse en pourcentage d'une vitesse enregistrée préalablement. Ce pourcentage est défini par rapport à une entrée analogique telle que 0V représente 0 % et 10V représente 100%.

En ce qui concerne les robots, le déclenchement de certaines actions en fonction des entrées détectées par leur automate doit être configuré au niveau de RobotStudio. Comme des entrées avaient déjà été configurées, il n'a pas été nécessaire de créer des variables supplémentaires. Il a suffi simplement de les lier à des séquences préprogrammées (MotorOn, StopInstruction, etc.) à l'aide de l'outil Configuration Editor et de l'onglet I/O. La marche à suivre complète peut être retrouvée ici [10].

Tableau 7.2 – Références virtuelles et électriques des variables de sortie du programme PLC.

Référence virtuelle			**Référence électrique**	
Nom de la variable	Terminal	Canal	Sortie	Terminal
O_Cnv_Start_Green	9	1	2	8
O_Cnv_Brk_Man	9	2	3	8
O_Cnv_SetPoint	9	3	4	8
O_RX_Relay	10	1	9	8
O_Rob1_MotOn	11	1	13	8
O_Rob1_Start	11	2	14	8
O_Rob1_RstEStop	11	3	15	8
O_Rob1_Stop	11	4	16	8
O_Buzzer	12	4	24	8
O_Lmp_Orange	12	5	25	8
O_Lmp_Green	12	6	26	8
O_Lmp_Blue	12	7	27	8
O_Lmp_Red	12	8	28	8
O_Enable_Encoder	12	1	21	8
O_Cnv_Speed	15	1	15	1

Tableau 7.3 – Références électriques des variables du convoyeur.

Référence électrique PLC			**Référence Convoyeur**	
Référence câble	Terminal	Canal	Sortie	Terminal
Cv_DI1_Start	9	1	24E	X4
Cv_DI2_Stop	9	2	DI3	X4
Cv_DI3_SetPoint	9	3	DI	X4
Cv_AI_Speed	15	1	A1U	X3
Cv_GIO_Ground	-	-	GIO	X4

Tableau 7.4 – Références électriques des variables du robot.

Référence électrique PLC			**Référence Robot**	
Référence câble	Terminal	Canal	Sortie	Terminal
O_Rob1_MotOn	11	1	3	X3
O_Rob1_Start	11	2	4	X3
O_Rob1_RstEStop	11	3	5	X3
O_Rob1_Stop	11	4	6	X3

7.2 Tests

Après revérification de tous les branchements, des tests ont pu être effectués pour voir si les modifications de la ligne avaient bien l'effet escompté.

Les différents boutons poussoirs du HMI ont ainsi pu être testé et il a été vérifié qu'il lançait bien les séquences désirées. De plus, l'activation du bouton d'arrêt d'urgence s'est également montrée efficace. L'utilisation de la tour lumineuse et de son buzzer a également été couronnée de succès.

Les différentes fonctionnalités configurées dans le code RAPID ont toutes étés testées lors du fonctionnement de l'application de lancer et les séquences se sont bien déroulées. Il faut toutefois remarquer qu'un problème lors de la reconnexion après un arrêt semble se produire, mais ce problème concerne le code du robot et sort donc du cadre de ce TFE.

Le contrôle du convoyeur a également été bien mis en place. La vitesse peut dès lors être contrôlée depuis le curseur du HMI et l'arrêt et la marche sont peuvent être gérés par le PLC. Il a été constaté que, lorsque le PLC contrôlait la vitesse du convoyeur, la fonction de marche et d'arrêt de la télécommande pouvait toujours fonctionner ce qui peut permettre aux chercheurs du GeMMe d'aussi le contrôler automatiquement. Idéalement, la vitesse réelle aurait dû être calculée à partir du voltage de la sortie et aurait pu être affichée sur le HMI directement. Des mesures de voltages et de vitesse ont ainsi étés faites pour établir une loi de correspondance, mais, après analyse des données, les vitesses mesurées correspondent à des vitesses bien trop lentes. Ces problèmes de données ont sans doute été causés par une erreur de configuration du tachymètre, mais aucune solution n'a été trouvée.

Finalement, les actions de la séquence de synchronisation semblent bien s'être déroulées et avoir été intégrées par l'application de l'ordinateur. Il est toutefois à noter que le résultat de cette synchronisation n'a pas pu être testé directement, car l'application en C++ n'implémentait pas encore toutes les fonctionnalités nécessaires et que celles-ci

nécessiteraient encore beaucoup de temps à développer.

Chapitre 8

Conclusions et perspectives

L'objectif de ce travail consistait à développer l'architecture et le programme de l'automate contrôlant la ligne de tri robotisée dans le cadre de l'axe Pick-It du projet Reverse Metallurgy. Suite aux retards sur le développement de la version industrielle de cette ligne, le but a du être redéfini pour rendre ce travail plus concret. L'objet s'est ainsi réorienté sur l'automatisation du prototype de la version industrielle. Afin toutefois de rendre ce travail applicable au démonstrateur, il a été exigé de développer des solutions les plus génériques possible. Au terme de ce travail de fin d'études, le framework développé concerne le contrôle des machines à l'aide de la norme CEI 61131-3 ainsi qu'une interface homme-machine (HMI) modulable. Ces solutions ont donc pu être appliquées pour automatiser certaines séquences du prototype de la ligne.

La première étape de ce travail a consisté à se familiariser avec le projet pour comprendre les besoins du client. L'objectif de la ligne, identifier puis trier des déchets métalliques parmi 26 classes et nuances a donc été précisé et caractérisé. Le rôle de chaque élément a ensuite été étudié en vue de l'automatisation de la chaîne. Cette étude a permis d'identifier le comportement que chaque élément doit suivre en fonction des différentes séquences ainsi que les moyens de communication disponibles pour permettre de contrôler chaque machine.

La deuxième étape du projet a concerné le développement des solutions génériques de programmation des automates. Une analyse en profondeur des nouvelles fonctionnalités de la norme CEI 61131-3 a ainsi permis l'utilisation du paradigme orienté-objet pour le développement des applications. Des blocs de fonction abstraits, assimilables à des classes dans d'autres langages de programmation, ont donc été employés pour obliger chaque machine à utiliser des variables et méthodes semblables à l'aide du mécanisme de l'héritage. Une certaine distance a toutefois été gardée par rapport à une approche purement orientée-objet afin d'intégrer certaines bonnes pratiques des automaticiens et d'éventuellement simplifier son adoption.

Le développement d'une solution d'interface entre l'homme et la machine (HMI) a également fait partie des challenges relevés dans ce travail. Pour présenter un réel intérêt, la solution se devait également d'être modulaire. Une application développée en .NET selon le framework WPF a ainsi été proposée, car la distinction entre la partie logique et graphique de l'application la rend facilement compatible avec plusieurs types d'automates tout en permettant un aspect visuel moderne.

Finalement, la dernière étape a été d'appliquer le framework au prototype de la ligne Pick-It, au laboratoire du GeMMe. Après une phase d'intégration, la séquence de synchronisation des machines a pu être testée avec succès grâce à la mise en place d'une connexion TCP/IP avec l'ordinateur d'acquisition. Le contrôle de certains éléments a pu ainsi être assuré et les règles de sécurité ont été intégrées dans les phases de contrôle. L'automatisation complète de la ligne n'a toutefois pas pu aboutir, car cela nécessiterait un travail important de câblage qui n'est sans doute pas adapté à la nature d'un prototype. De plus, un HMI a été développé en .NET sur base de leurs besoins actuels et futurs.

Au vu de l'envergure de Pick-It, il existe beaucoup de pistes qui peuvent permettre au projet d'être amélioré dans le futur. En ce qui concerne les outils génériques développés, leur implémentation dans d'autres projets mettrait sans doute en évidence des manque-

ments ou des fonctionnalités à peaufiner pour améliorer son ergonomie. La solution de HMI pourrait, de son côté, encore permettre un système de connexion d'utilisateurs, le chargement de configurations à partir d'un ficher XML ou encore être adaptée à un usage mobile. Il a été constaté que le prototype était l'outil idéal pour valider les technologies en prévision du démonstrateur. La répartition intelligente des tâches entre les robots ou la mise en place de structures de sécurité physiques et virtuelles pourraient donc y faire l'objet de futurs travaux.

De manière générale, ce travail a permis d'explorer l'utilisation de nouvelles technologies afin d'estimer leurs intérêts réels. Ces dernières années, les évolutions technologiques ont en effet créé de nombreuses opportunités dans le développement de solutions industrielles pour en arriver à l'avènement de l'industrie 4.0. Le choix de l'adoption de certaines de ces nouvelles technologies représente un défi compliqué puisque d'un côté, un investissement est nécessaire pour évaluer la maturité de la technologie et former le personnel, mais d'un autre côté, il existe un risque d'être progressivement dépassé au point de finir obsolète si les technologies ne sont pas adoptées. L'ambition de ce travail est notamment d'apporter un soutien pour faire ce choix dans les meilleures conditions.

Bibliographie

[1] Peter Adams. “The title of the work”. In : *The name of the journal* 4.2 (juil. 1993). An optional note, p. 201–213.

[2] Sick AG. *3D Cameras Ruler E - Technical Description*. URL : http://sick.tta.ru/sites/sick.tta.ru/files/File/pdf/Div01/ruler.pdf (visité le 25/04/2018).

[3] Ernie Hayden et Al. “An Abbreviated History of Automation Industrial Controls Systems and Cybersecurity”. In : *A SANS Analyst Whitepaper* (août 2014).

[4] Pierre Barnabé et Al. “Design and calibration of a two-camera (visible to near-infrared and short-wave infrared) hyperspectral acquisition system for the characterization of metallic alloys from the recycling industry”. In : *Journal of Electronic Imaging 24(6)* (2015).

[5] Mesta Automation. *HMI with C and WPF part 2 : Navigation with PRISM*. URL : https://www.mesta-automation.com/hmi-with-c-and-wpf-part-2-navigation-with-prism/ (visité le 18/05/2018).

[6] Beckhoff. *ADS-Communication*. URL : https://infosys.beckhoff.com/english.php?content=../content/1033/cx8095_hw/1610551947.html&id= (visité le 15/05/2018).

[7] BECKHOFF. *TwinCAT 3 : I/O - Adding an I/O Device*. URL : https://infosys.beckhoff.com/english.php?content=../content/1033/tc3_io_intro/html/tcsysmgr_configio_intro2.htm&id= (visité le 29/05/2018).

[8] BECKHOFF. *TwinCAT ADS .NET Component*. URL : https://infosys.beckhoff.com/english.php?content=../content/1033/tcadsnet/html/twincat.ads.intro.html&id= (visité le 15/05/2018).

[9] BECKHOFF. *TwinCAT TCP/IP Connection Server - Reference*. URL : https://infosys.beckhoff.com/english.php?content=../content/1033/tcpipserver/html/tcplclibtcpip_sample01_overview.htm&id=7285948521234898622 (visité le 22/05/2018).

[10] Henrik BERLIN. *How to configure I/O for a robot controller system in RobotStudio*. Mai 2013. URL : https://www.youtube.com/watch?v=JSdcVxgR6hg (visité le 29/05/2018).

[11] Microsoft COMMUNITY. *Data Binding (WPF)*. URL : https://docs.microsoft.com/en-us/dotnet/framework/wpf/data/data-binding-wpf (visité le 18/05/2018).

[12] David A. CREMERS. "The Analysis of Metals at a Distance Using Laser-Induced Breakdown Spectroscopy". In : (sept. 1987).

[13] Godefroid DISLAIRE. *Scrap sorting by hyperspectral imaging and delta robot gripping (Video)*. Nov. 2016. URL : https://www.youtube.com/watch?v=N7KQGn1ZgHA (visité le 11/05/2018).

[14] Pierre DUYSINX. *Cours d'Automatisation industrielle - Partie Automates Programmables*. Chapitre 1, Slide 15.

[15] FUTURA-SCIENCES. *Automates programmables*. URL : https://www.automation-sense.com/pages/c-est-quoi-un-automate.html (visité le 25/04/2018).

[16] Maxime GOBERT. "Travail de fin d'études - Étude du layout et dimensionnement d'une ligne de tri de déchets". Mém.de mast. Haute École de la Province de Liège, 2017.

[17] José Miguel Gutiérrez GUERRERO et Juan Antonio Holgado TERRIZA. "Mobile Human Machine Interface based in OPC UA for the control of industrial processes". In : *Jornadas de Automática* 36 (sept. 2015).

[18] MPaul JENKINS. *MahApps.Metro*. URL : `https://mahapps.com/` (visité le 20/05/2018).

[19] Norman MARLIER. "Travail de fin d'études - Apprentissage par renforcement en vue de l'amélioration d'une ligne de production robotisée". Mém.de mast. Université de Liège, 2018.

[20] Tuxel-vib S.A. N.V. *Circuits électroniques de commande pour vibrateurs électromagnétiques*. URL : `http://tuxel-vib.com/html/download/TUXEL-VIB_CONTROLLERS_FR.pdf` (visité le 25/04/2018).

[21] Reverse Metallurgy S.C.R.L. *Communiqué de presse - Création de Reverse Metallurgy*. URL : `http://www.reversemetallurgy.be/fr/presse.html?ID=76` (visité le 16/03/2018).

[22] Lenze SE. *Inverter Drives 8400 StateLine C - Operating Instructions*. URL : `http://download.lenze.com/TD/E84AVSCx__8400%5C%20StateLine%5C%20C__v1-2__EN.pdf` (visité le 25/04/2018).

[23] SMC. *Operation Manual and Communication Function - Thermo Chiller*. HRX-OM-M091-G - 7th edition : Feb.2011.

[24] VJ TECHNOLOGIES. *Operation Manual for IXS Series High Voltage X-Ray Units*. ZD3000-001 REV A1.

[25] Karl-Heinz John - Michael TIEGELKAMP. *IEC 61131-3 : Programming Industrial Automation Systems*. Second Edition. Springer, 2009-2010.

[26] Scott WHITLOCK. *TwinCAT 3 Tutorial : Building an HMI in .NET*. URL : `http://www.contactandcoil.com/twincat-3-tutorial/building-an-hmi-in-net/` (visité le 15/05/2018).

Analyse BOSCAR

.0.1 Background

Qui est le client ? Comet Traitement.

Comment l'idée est-elle née ? L'idée est née dans le cadre du projet Reverse Metallurgy qui vise à faire de la Région wallonne un pôle de l'économie circulaire.

Quels sont les problèmes auxquels on veut remédier ? À l'heure actuelle, le recyclage des déchets métalliques est très sommaire. Il consiste principalement à séparer les métaux ferromagnétiques de ceux qui ne le sont pas puis de fondre les deux ensembles et les vendre comme un mélange dont la composition est estimée. Un tri plus spécifique des déchets permettrait d'améliorer la pureté des familles de matériaux triés et d'augmenter leur valeur marchande. Cela aurait pour conséquence de permettre une meilleure réutilisation des déchets, de diminuer les besoins d'importation des métaux et finalement de réduire l'emprunte écologique de ces matériaux.

Quels sont les enjeux liés au projet ? Développer une cellule automatique capable d'identifier puis trier des déchets métalliques en fonction de leur composition afin de les réutiliser.

.0.2 Objectives

Que doit-on réaliser par ce projet ? Une ligne capable d'identifier un déchet parmi vingt-six familles de matériaux puis de le trier à l'aide d'un robot. La cadence de tri doit être d'au moins 4.7 déchets par seconde et le système doit coûter le moins cher possible pour espérer être rentable.

Que veut-on changer, de combien avec quelle qualité et dans quels délais ? Le principal défi est d'augmenter le nombre de nuances identifiées et triées. Idéalement, développer le système pour passer de deux nuances triées à vingt-six durant les quatre années pendant lesquelles le projet est financé.

En quoi le projet contribue-t-il à faire progresser l'entreprise ? L'entreprise est déjà spécialisée dans le traitement déchet mais un meilleur tri lui permettra d'augmenter la qualité de ses produits en augmentant la pureté de ses métaux.

Nos objectifs sont-ils significatifs, réalistes, correctement quantifiés ? Le réalisme des objectifs initiaux est difficile à estimer. Bien que les technologies nécessaires pour la mise en place d'une telle ligne existent, ils ne sont généralement pas utilisés à cet effet. Une validation technologique a donc été nécessaire

Quelles sont les éventuelles priorités ? La première priorité était la validation des technologies au moyen du prototype. Désormais, il est nécessaire de s'accorder sur le moyen d'implémenter une ligne industrielle la plus fonctionnelle et rentable possible.

.0.3 Scope

Quel est le périmètre du projet ? Le périmètre du projet est constitué par les membres du consortium Pick-It à savoir Citius Engineering, le GeMMe et Comet Traitement.

Quelles sont les limites du projet ? La cadence de tri est bornée par la vitesse de déplacement limitée des robots ainsi que le temps nécessaire pour l'ouverture et la fermeture des pinces pneumatiques. D'autre part, le poids de la charge que peut supporter le robot limite implicitement les dimensions des pièces manipulées.

Quels sont tous les interlocuteurs concernés ? Le périmètre du projet est constitué par les membres du consortium Pick-It, les responsables politiques liées au projet Reverse Metallurgy et, éventuellement, peut être étendu aux entreprises liégeoises de métallurgie.

.0.4 Constraints and Opportunities

Contraintes

- Temporelle : La durée du financement dépend des résultats intermédiaires du projet.
- Personnel : Le personnel se compose de l'équipe de Citius Engineering, des chercheurs du GeMMe et de l'équipe de Comet Traitement.
- Budgétaire : La bourse allouée au projet est de 70 millions d'euros, dont 42 d'investissement public.
- Design : Le grand nombre de nuances à trier par robot impose la grande taille et la complexité de la ligne.

Peut-on préciser les contraintes qui peuvent survenir lors de la réalisation du projet ? Certaines technologies d'identification sont encore très expérimentales et donc leur utilisation dans un cadre industriel pourrait poser des problèmes de fiabilité, de sécurité et d'ergonomie.

Quelles sont les opportunités à saisir pour initier le projet ? Dans un premier temps, il est faut identifier les technologies potentiellement capables de répondre aux besoins fonctionnels puis de s'accorder sur la coordination des éléments entre eux. Pour cette étape d'initialisation, la collaboration entre Citius, le GeMMe et Comet Traitement est primordiale afin de tirer profit de leurs expertises respectives.

Que risque-t-on si on reporte ou si on ne réalise pas le projet ? Puisque le travail des différentes entités est notamment financé par la Région wallonne, la non-réalisation du projet serait perçue comme une mauvaise allocation de l'argent public. De plus, les déchets métalliques continueraient à être peu revalorisés.

Quels sont les risques liés à la réalisation du projet ? Au terme de la réalisation du prototype, il se pourrait qu'aucune solution technique ne parvienne à répondre aux besoins de rentabilité et que le projet soit abandonné. De plus, comme les trois intervenants sont issus de milieux différents, des problèmes de coordination et d'approbation peuvent survenir.

Quelles sont les interrelations avec d'autres projets ? L'axe Pick-It s'inscrit dans un projet de plus grande envergure de la Région wallonne, Reverse Metallurgy, dont l'objectif est de stimuler l'économie circulaire.

.0.5 Assumptions

Quelles sont les incertitudes qui existent aujourd'hui quant au problème à traiter ? Les incertitudes concernant la faisabilité ont étés levées à l'aide du prototype, mais la configuration du démonstrateur et l'efficacité réelle de l'identification restent à déterminer.

Quelles sont les principales hypothèses de travail ? Les principales hypothèses de travail résident dans les éléments d'analyse spectrale, de robotique et d'automatisation.

Quelles sont les principales solutions proposées pour réussir le projet ? L'identification peut s'effectuer au moyen d'une caméra hyperspectrale, d'une caméra volumique, de rayons X et d'un rayon LIBS. Le tri peut quant à lui se réaliser soit à l'aide

de robots déplaçant les déchets en les attrapant ou en les faisant glisser soit à l'aide de soufflette.

Y a-t-il des solutions alternatives ? Une amélioration simple du système actuel pourrait se faire au moyen d'un tri manuel effectué par du personnel formé, mais ses performances sont limitées et son coût est élevé.

A-t-on calculé la rentabilité de la solution optimale choisie ? Un premier calcul de rentabilité a été effectué, mais, au vu de l'évolution à prévoir des technologies, ce résultat de ce calcul ne peut pas être considéré comme étant définitif.

.0.6 Reporting

Quels pourraient être les grandes étapes du projet ? Les étapes principales du projet peuvent s'exprimer comme suit : La définition du besoin, le choix des solutions techniques à tester via le prototype, l'installation et l'intégration des éléments du prototype, la conception du démonstrateur à partir des résultats du prototype, l'installation du démonstrateur.

Quel est le macro-planning ? Pour 2016, un prototype fonctionnel doit avoir été développé et pour de 2019, le démonstrateur doit être installé.

Quels sont les grands livrables du projet ? Le prototype de recherche et le démonstrateur industriel.

Quel est le rythme de reporting au comité de pilotage ? Le comité se réunit lorsqu'une décision importante doit être prise.

Quels seront les acteurs du projet ? Les principaux acteurs sont les ingénieurs de Citius Engineering et de Comet Traitement ainsi que les chercheurs du GeMMe.

Project Description	
Project :	P14074
Client :	CITIUS
Description :	Reverse Metallurgy
Place :	

PER		
BY	DATE	SIGNATURE

RELEASES				
RELEASE	BY	DATE	CONTROL	COMMENT
V00	ABA	09-05-16	MHU	
V02	STH	15-12-16	FTO	Update

Face avant armoire 1

Qté	Description	Marque	Référence	Etat
1	Sectionneur sur porte 20A	Schneider	VCF01	
1	Voyant lumineux Blanc 240V + LED	Schneider	XB4BVM1	
1	Bouton arrêt d'urgence	Schneider	XB4BS8445	
1	Selecteur à clé	Schneider	ZB4BG02	
1	Tête	Schneider	ZB4BZ009	
3	Voyant vert 24V	Schneider	XB4BVB3	

1	Bouton poussoir noir	Schneider	XB4BA21	

Panel

Qté	Description	Marque	Référence	
1	Boite à bouton	Schneider	XACB1215	
1	Bouton arrêt d'urgence	Schneider	XB4BS8445	
1	Bouton poussoir bleu	Schneider	XB4BA61	
4	Bouton poussoir blanc	Schneider	XB4BA3341	
2	Bouton poussoir vert	Schneider	XB4BA31	
2	Bouton poussoir flèche	Schneider	ZB4BA335	
1	Selecteur	Schneider	XB4BD21	
~~1~~	~~Tête potentiometre~~	~~Schneider~~	~~ZB4BD922~~	
~~1~~	~~Potentiometre~~	~~TE Connectivit~~	~~23ESA103MMF50NF~~	
5	Contact NO	Schneider	ZENL1111	
5	Contact NF	Schneider	ZBE1023	
1	Tour lumineuse	Banner		Citius

Armoire 1

Qté	Description	Marque	Référence	
1	Armoire 800x1200x300	Rittal	AE1280500	ULg
1	Porte Fusible 2p 10x38	Schneider	DF102	

3	Fusibles 10x38 1A	Ferraz	FR10GG50V1		
1	Disjoncteur IC60N 16A C 2P	Schneider	A9F79216		
1	Différentiel 300MA type A	Schneider	A9Q24225		
1	Disjoncteur IC60N 6A C 2P	Schneider	A9F79206		
1	Disjoncteur IC60N 10A C 2P	Schneider	A9F79210		
7	Disjoncteur 1P IC60N 6A C	Schneider	A9F79106		
1	Prise pour rail DIN	LeGrand	4280		
1	Alimentation sécurisé 10A	Phoenix	2866763		
1	Module de diagnostique	Siemens	6EP19612BA00		
1	Relais d'arrêt d'urgence S4	Pilz	750104		
1	Extension relais AU	Pilz	750107		

1	Bus Coupler EK1100	Beckhoff	EK1100		Citius
4	Carte 4DI	Beckhoff	EL1104		Citius
2	Carte 8DI	Beckhoff	EL1008		Citius
1	Carte 2AI	Beckhoff	EL3255	remplacé par EL3062	Citius
4	Carte 4DO	Beckhoff	EL2004		Citius
1	Potentiel Supply	Beckhoff	EL9100		Citius
2	Carte 8DO	Beckhoff	EL2008		Citius
1	Carte 2AO	Beckhoff	KL4002		Citius
1	PC Industriel	Beckhoff	C6930		Citius

200	Bornes ST2,5	Phoenix	3031212		
10	Bornes de terre ST2,5	Phoenix	3031238		
4	2m Rail Din	Erico	557950	Ou équivalent	
5	Goulottes 60x80mm	TEHALIT	BA780060G	Ou équivalent	
2	Patch ethernet 15m	RS PRO	791-7012	Ou équivalent	
2	Patch ethernet 2m	RS PRO	556-657	Ou équivalent	
1	Switch ethernet	Abitana	ABI-EL3008S0(	Ou équivalent, Minimum 8 entrées	
~~1~~	~~Drive pince + connecteur CN1 / CN5~~	~~SMC~~	~~LECP6~~		ULg
~~1~~	~~Encodeur + connecteur~~	~~Sick~~	~~DFV60~~		ULg
1	Répartiteur pour signaux de codeurs	Motrona	GV481		ULg
8	Convertisseurs de niveau	Motrona	IT251		ULg
1	Kit de fixation coffret	Rittal	2508.010		
1	Plaque passe-câbles en plastique	Rittal	2563.500		
4	Passe cloison Ethernet	Harting	09 45 452 1560		
4	Passe cloison USB	Harting	09 45 452 1901		
8	Bouchon de protection	Harting	09 45 502 0000		
1	Plaque brosse passe-câble	Rittal	2563150		
1	Brosse passe-câble	Rittal	7072200		
2	Barre maintien des câbles	Phoenix	0402174		
4	Accesoires Barre maintien des câb	Phoenix	0404428		
1	Répartiteur modulaire	Legrand	04880		

	~~Air entrée, sur arrêt d'urgence~~		~~A définir~~	
	~~Cartes pour encodeurs~~		~~A définir~~	
	~~Connecteur Harting ou équivalent~~		~~A définir~~	
	~~Intégrer 4 DO pour les pinces.~~		~~A définir~~	
	Alimentation 5VDC	896-2224		ULg
	Alimentation 12 VDC	896-2259		ULg

Armoire IRC5_1 & 2

Qté	Description	Marque	Référence	
2	Bus Coupler EK1100	Beckhoff	EK1100	Citius
4	Carte 8DI	Beckhoff	EL1008	Citius
4	Carte 8DO	Beckhoff	EL2008	Citius

Armoire 2

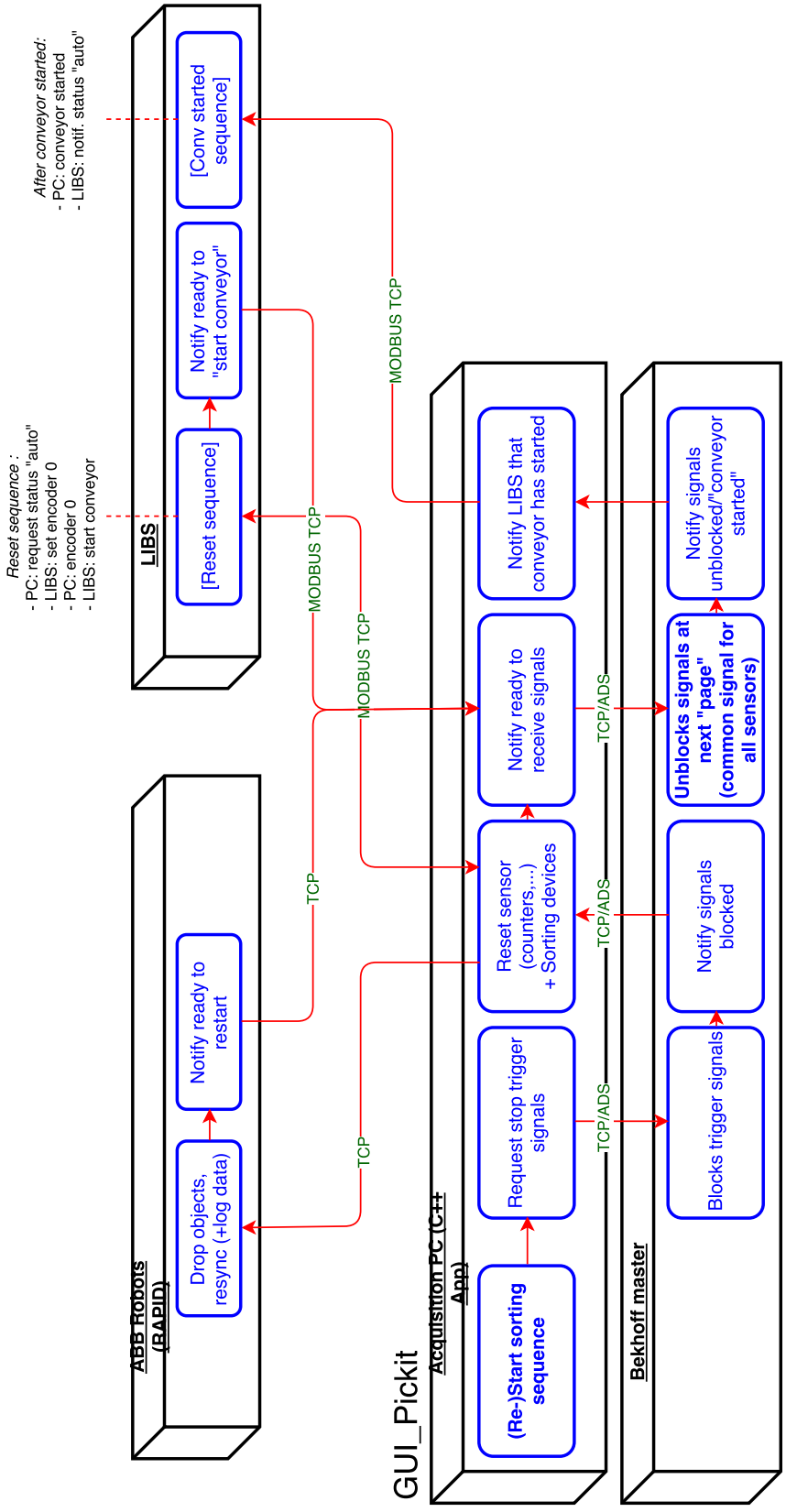
Qté	Description	Marque	Référence	
1	Coffret	Rittal		ULg
1	Sectionneur sur porte 32A	Schneider	VCF1	ULg?
1	Disjoncteur IC60N 6A 3P	Schneider	A9F79306	ULg?
1	Contacteur 9A	Schneider	LC1D09BD	ULg?
1	Drive moteur convoyeur	Lenze 8400	E84A VSC E7514 5 B0	ULg

Site

Qté	Description	Marque	Référence
3	Boutons d'arrêt d'urgence	Schneider	XALK178

Filerie

Qté	Description	Marque	Référence	
100	VOBst 2,5 Brun	EUPEN	V0BST2,5M R 100	
100	VOBst 2,5 Bleu	EUPEN	V0BST2,5B B 1500	
50	VOBst 2,5 Jaune/Vert	EUPEN	V0BST2,5YG R 100	
100	VTBst 0.75 Rouge	EUPEN	VTBST0,75R R 100	
100	VTbst 0.75 Blanc	EUPEN	VTBST0,75I R 100	
100	VTbst 0.5 Brun	EUPEN	VTBST0,5M R 100	
100	VTbst 0.5 Blanc	EUPEN	VTBST0,5I R 100	
10	VOBst 6 Jaune/Vert	EUPEN	V0BST6YG R 100	
100	LIYCY 4x0.75	CABLE DE C	LIYCY4X0,75	CA01, CA02, CA03, CA05
20	LIYCY 2x0.75	CABLE DE C	LIYCY2X0,75	CA04
50	LIYCY 20x0.75	CABLE DE C	LIYCY20X0,7	CA07
100	LIYCY 8x0.75	CABLE DE C	LIYCY8X0,75	CA09, CA10, CA11, CA12
25	LIYCY12x0.75	CABLE DE C	LIYCY12X0,7	CA08
50	LIYCY 16x0.75	CABLE DE C	LIYCY18X0,7	CA06, CA09
100	Embout simple 0.75	Schneider	DZ5CE007	
50	Embout double 0.75	Schneider	AZ5DE007	
100	Embout simple 2,5	Schneider	DZ5CE025	
50	Embout double 2,5	Schneider	AZ5DE025	


GUI_PickIt
ABB Robots (RAPID)
Drop objects, resync (+log data)
Notify ready to restart
LIBS
[Reset sequence]
Notify ready to "start conveyor"
[Conv started sequence]
Acquisition PC (C++ App)
(Re-)Start sorting sequence
Request stop trigger signals
Reset sensor (counters,...) + Sorting devices
Notify ready to receive signals
Notify LIBS that conveyor has started
Bekhoff master
Blocks trigger signals
Notify signals blocked
Unblocks signals at next "page" (common signal for all sensors)
Notify signals unblocked/"conveyor started"
Reset sequence :
- PC: request status "auto"
- LIBS: set encoder 0
- PC: encoder 0
- LIBS: start conveyor
After conveyor started:
- PC: conveyor started
- LIBS: notif. status "auto"
TCP
TCP
MODBUS TCP
MODBUS TCP
MODBUS TCP
TCP/ADS
TCP/ADS
TCP/ADS

Cahier des charges : HMI du boitier électrique Pickit

1. Introduction

Ce document liste les fonctionnalités envisagées pour le HMI (interface homme-machine) du boitier électrique du prototype Pickit, réalisé par Citius Engineering et actuellement situé dans le laboratoire de Gemme.

Il est important de noter que celui-ci ne reprend pas les points relatifs à la sécurité des équipements, qui ont déjà été discutés précédemment (i.e. : circuit interlock n'autorisant le fonctionnement de l'éclairage halogène que lorsque le convoyeur est en mouvement ; circuit n'autorisant le fonctionnement de l'éclairage LED que lorsque le refroidisseur est enclenché, etc.).[1]

2. Description

Afin d'orienter le prototype Pickit vers un instrument d'une finition de niveau « semi-industriel », il est souhaité de proposer une interface homme-machine centralisée, qui permet l'activation d'un nombre restreint d'actions/tâches qui pourraient répondre aux attentes de l'utilisateur industriel, mais également de continuer à faciliter le prototypage en laboratoire. Ceci a été pensé dans la mesure où le boitier dispose déjà d'un automate Bekhoff qui pourrait facilement faire l'interface entre l'utilisateur et les différent modules.

Nous proposons de doter le HMI de plusieurs pages d'affichage qui permettent d'interagir avec différents niveaux de fonctionnalités. Le HMI serait affiché sur un écran tactile (sur le boitier).

A chaque point est associé un niveau de priorité allant de 1 à 3 (1 étant le plus prioritaire).

Voici les différentes pages envisagées :

1. Page « **Status »** reprenant les états des équipements et permettant, pour la plupart, d'interagir avec ceux-ci. Nous proposons de distinguer trois niveaux d'état :
   « POWER OFF »/ « POWER ON » /« ACTIVE »
   Ceux-ci peuvent être matérialisés, par exemple, par trois boutons à deux états (toggle) dans le HMI, permettant, pour certains équipements, de contrôler leur état via cette page. NB : Le statut « Actif » implique évidemment que l'équipement est sous tension. L'état de l'équipement peut donc être renseigné de façon exclusive par un de ces 3 états. Dans certains cas, l'état « Actif » sera omis, comme détaillé dans le tableau qui suit. D'autre part, le contrôle des états ne doit pas systématiquement être accessible via le HMI, et les instruments sont détaillés ci-après au cas par cas (lorsque la mise sous tension/hors tension ne doit pas se faire via le HMI, la boîte « Contrôle » est assortie d'une croix).
   Priorité = 1

[1] Certains de ces points sont relevés dans le document envoyé en annexe, ainsi que la documentation de certains équipements, à titre de support

Equipement	Power On / Power Off	Active	Priorité
2 Robots ABB	Affichage ✓ Contrôle ✗	Ce statut doit être renseigné par les routines en RAPID directement via une connexion TCP avec les robots. Lorsque la routine qui reçoit les objets et lance les instructions de déplacement est en exécution => il est *actif*. Lorsque la routine est à l'arrêt / le robot attend la connexion avec PC ou la trame de synchronisation (Work object) => il est *inactif (i.e. « Power On »)*	1
Camera hyperspec.	Affichage ✓ Contrôle ✗	*Non nécessaire.*	2
Camera 3D	Affichage ✓ Contrôle ✗	*Non nécessaire.*	2
Eclairage LED	Affichage ✓ Contrôle ✓	*Non nécessaire.*	1
Eclairage Halogène	Affichage ✓ Contrôle ✓	*Non nécessaire.*	1
Convoyeur à bande (moteur)	Affichage ✓ Contrôle ✓	Ce statut doit être lié à une valeur limite de la vitesse du moteur. L'état 'start/stop' du moteur n'est pas suffisant, car en deçà d'une certaine vitesse (actuellement, via la télécommande, en deçà de 3), le convoyeur n'est plus en mouvement. 'Active' doit être lié au fait que le convoyeur est en mouvement ou non. Il est important de préciser que c'est cet état qui doit également être considéré pour la règle de l'arrêt de l'halogène lorsque le convoyeur est '*inactif*' (cf. règles de sécurité). L'activation du statut 'Actif' correspond au 'Start' de la télécommande, mais le statut ne doit passer à 'Actif' que si la vitesse réelle est non nulle (valeur du potentiomètre au-dessus d'une valeur limite).	1
Refroidisseur (chiller)	Affichage ✓ Contrôle ✓	L'état *actif* doit garantir que le refroidissement est en cours (se référer à la doc du chiller). C'est également cet état *actif* qui doit être considéré pour la règle liant l'état du chiller à l'allumage de l'éclairage LED.	1
Boitier soufflettes (autre PLC Bekhoff)	Affichage ✓ Contrôle ✗	Tout comme pour les robots ABB, l'état *actif* est lié aux routines du second Bekhoff et doit être renseigné via connection TCP.	3
Alimentation vibrante	Affichage ✓ Contrôle ✓	*Non nécessaire.*	1
Source RX	Affichage ✓ Contrôle ✓	Etat actif = tube à rayons X actif. NB : Ces états sont déjà connectés aux lampes situées au-dessus du boîtier, conformément aux règles de sécurité en vigueur concernant l'utilisation d'une source RX. Le contrôle de l'état actif ne peut se faire que via le software dédié.	1
Capteur RX	Affichage ✓ Contrôle ✓	*Non nécessaire.*	2
Système LIBS	*À définir*	*A définir après réception et prise en main de l'équipement.*	3
PC acquisition	Affichage ✗ Contrôle ✗	Une communication TCP avec le logiciel d'acquisition pourrait renseigner sont état et lui envoyer des instructions (cfr. Tâches, plus bas).	3

2. Page « **Rules** » listant les règles de sécurités (toutes actives par défaut) et permettant de les désactiver, à l'exception évidemment de l'enclenchement du bouton d'arrêt d'urgence. Pour plus de clarté, les règles de sécurité demandées sont rappelées ci-après :
    - Mettre hors tension l'éclairage halogène lorsque le convoyeur passe à l'arrêt (état inactif)
    - Mettre à l'état actif le refroidisseur si l'éclairage LED est activé (équivalent à mis sous tension) + mettre hors tension l'éclairage LED si le refroidisseur est désactivé.
    - Mettre à l'état inactif l'alimentation vibrante si le convoyeur passe à l'arrêt.

    Priorité = 1

3. Pages « **Tasks** » listant un certain nombre de tâches souhaitées pour le prototype, impliquant l'activation séquentielle de différent organes. Les tâches envisagées à l'heure actuelle sont les suivantes :
    - Start/Restart acquisition & sorting :
        - Mise sous tension et à l'état actif de tous les équipements si ce n'est le cas (dans un ordre qui satisfait les règles de sécurité)
        - Envoi d'une instruction de réinitialisation au PC d'acquisition, aux robots, et au PLC contrôlant les éjecteurs pneumatiques
    - Stop :
        - Passage des tous les équipements à l'état inactif. Similaire à l'arrêt d'urgence sur le fond mais, en revanche, ne met pas les équipements hors tension brutalement (ex. : pour les robots : envoi de l'instruction de se mettre en état « inactif », reçu par routine RAPID)
    - Start/Restart acquisition :
        - Similaire au premier point, mais ne met à l'état actif que les capteurs + sources et le PC, pour procéder à des campagnes de caractérisation sans tri robotisé.

    Priorité = 2 (nécessite l'implémentation de la communication du PLC avec robots, PC acquisition etc. Fonctionnalité approchant celle du produit fini)

4. Page « **Logs** » reprenant des informations liées aux performances du prototype ainsi que sur la composition du produit analysé / trié. Informations envisagées :
    - Pour chaque module de tri (robots et éjecteurs), réception et affichage des informations suivantes :
        - Nombres d'objets reçus à trier
        - Nombres d'instructions de mouvement / d'éjection envoyées avec succès

      Note : permet d'évaluer l'efficacité d'une campagne de tri
    - Pour le PC d'acquisition :
        - Nombres d'objets identifiés par classes
    - Autres informations : Durée de la campagne d'acquisition / tri
    - Autres informations d'intérêt liées aux équipements

    Priorité = 3

Communication protocol: PLC <-> PC <-> Machines (ABB, LIBS,...)

This document aims to describe the communication process between the computer, the PLC and other devices on the line. The idea is to define how the states evolve through the different interactions of the machines. Afterward, the content and the format of the message are described in order to set up a standard that will be shared by all devices communicating on the line.

0. Context

A. PC program

It is important to notice the difference between the **STATES** and the *actions* of the main software.

For a given state, specific actions are run by the software in order to affect the devices in its environment. When those devices have interpreted the order accordingly and finished their action, messages are sent back to the PC. After acknowledging all of those notices, the PC then change its state to the next one.

- **STOPPING_TRIGGER :**
  - *StopTrigger* (Necessary action to stop the trigger)
    - IF StopTriggerOK == TRUE THEN STATE = RESETTING_SENSOR ELSE Send(Bck, StopTrigger)

      NB: Where StopTriggerOK is set to TRUE if the PC reads the appropriate response from the Beckhoff (other thread to deal with that)

- **RESETTING_SENSOR :**
  - *ResetSensor* (Necessary action to reset the sensor)
    - IF ResetSensorOK == TRUE THEN STATE = NOTIFYING_READY ELSE Send(LIBS, Reset), Send(ABB, Reset), ...

      NB: Where ResetSensorOK is set to TRUE if the PC reads the appropriate response from the Libs & the ABB (other thread to deal with that)

- **NOTIFYING_READY**
  - *NotifyReady* (Necessary action to notify the PLC about the machines readiness)
    - IF ConveyorStartOK == TRUE THEN STATE = NOTIFYING_START ELSE Send(Bck, ReadyToReceive)

NB: Where ConveyorStartOK is set to TRUE if the PC reads the appropriate response from the Beckhoff (other thread to deal with that) → The complex part of this sequence happens in the PLC.

- **NOTIFYING_START**
  - *NotifyStart* (Necessary action to notify the PLC about the start of the conveyor)
    - IF NotifyStartOK == TRUE THEN STATE = RUNNING
      ELSE Send(LIBS, Started) AND NotifyStartOK = TRUE

      NB: Action to do only once? Is there a way to acknowledge the reception?

- **RUNNING**
  - *Run* (Necessary action to run the line)
    - IF Error THEN STATE = E_STOP

- **E_STOPPING**
  - *EStop* (Necessary action while the emergency stop is ON)
    - *Stop acquisition?*

B. PLC program

The PLC is running a TCP Client program that polls every 1.5 seconds port 3003 for data coming from the PC. Each instruction is then saved in a FIFO Queue. The Client_Application function block is then popping the instruction out of the list and the different flags are set to signal the LineModes program that an action is required by the PC.

NB: Some states might seem/be unnecessary but, in my opinion, it is easier to suppress some rather than add some. Verification or acknowledgment steps could therefore be easily added.

- **STAND_BY:**
  - / (No necessary action)
- **STOPPING_TRIGGER :**
  - *StopTrigger* (Necessary action to stop the trigger)
    IF StopTriggerOK == TRUE THEN STATE = STAND_BY
    ELSE Triggers.Stop() & StopTriggerOK = TRUE

    NB: Where the STOPPING_TRIGGER state is set, among others, by the content of the PC message.

- **NOTIFYING_STOP_TRIGGER :**
  - *NotifyStopTrigger* (Necessary action to notify the trigger stop)
    - IF *NotifyStopTrigger*OK== TRUE THEN STATE = STAND_BY
      ELSE Send(PC, StopTriggerOK) & NotifyStopTriggerOK = TRUE

NB: To reach this state, no external intervention from the PC is needed.

- **UNBLOCKING_SIGNAL :**
  - *UnblockSignal* (Necessary action to unblock the signals from the sensors)
    - IF UnblockSignalOK == TRUE THEN STATE = NOTIFYING_START
      ELSE Signals.Unblock() & UnblockSignalOK = TRUE

      NB: Where *UnblockSignal* is set to TRUE if the PLC reads the appropriate message from the PC.

- **NOTIFYING_START**
  - *NotifyStart* (Necessary action to notify the PC about the start of the signals)
    - IF NotifyStartOK == TRUE THEN STATE = RUNNING
      ELSE Send(PC, ConveyorStartOK) & NotifyStartOK = TRUE

      NB: To reach this state, no external intervention from the PC is needed.

- **RUNNING**
  - *Running* (Necessary action to run the line)
    - IF Error THEN STATE = E_STOP

- **E_STOPPING**
  - *EStop* (Necessary action while the emergency stop is ON)
    - *???*
    -

I. Protocol definition

A. PLC Input

i. Content

- ❏ Sender ID
- ❏ Receiver ID
- ❏ Message ID
- ❏ Phase (Synchronization, Acquisition, etc.)
- ❏ Action requested (Stop the triggers, start, stop, etc.)
- ❏ Action argument #1
- ❏ Action argument #2
- ❏ Action argument #3

ii. Type of interface:

TCP/IP over Ethernet

iii. Message composition

Because of the easiness of string manipulation in high level languages like C++ and Structured Text language (IEC 61131-3), the message is encoded as a string. The influence on performance is expected to be insignificant as the messages sent are small and sporadic. Moreover, this choice allows more flexibility because the two used languages (C++ and ST) do not use a common representation for the every types and the arguments are not type-specific.

Properties :

1. The elements of the frames are separated by hyphens (-) such their sizes can vary in a given range.
2. The content of the string is then encoded in bytes as requested by the TCP IP.
3. The message starts with a '@' and finishes by an end character, a '0' number (not the character one) represented in hexadecimal i.e. '\x00'.

→ Frame: @senderid-receiverid-messageid-phase-action-arg1-arg2-arg3-

Example of a frame with no arguments: @1-10-103-sync-stoptrig- - - - -

Dummy example with all arguments used: @1-10-1240-prod-start-120.1-25.0-225-

NB: The end character 0 ('\x00') is not represented as it is not within the string.

Table 1. Main characteristics of frame elements

Element name	Max number of chars	Type	Range
Sender ID	4	unsigned integer	[0-9999]
Receiver ID	4	unsigned integer	[0-9999]
Message ID	8	unsigned integer	[0 - 99 999 999]
Phase	8	char string	-
Action	16	char string	-
Argument 1	8	char string	-
Argument 2	8	char string	-
Argument 3	8	char string	-

A. Dictionaries

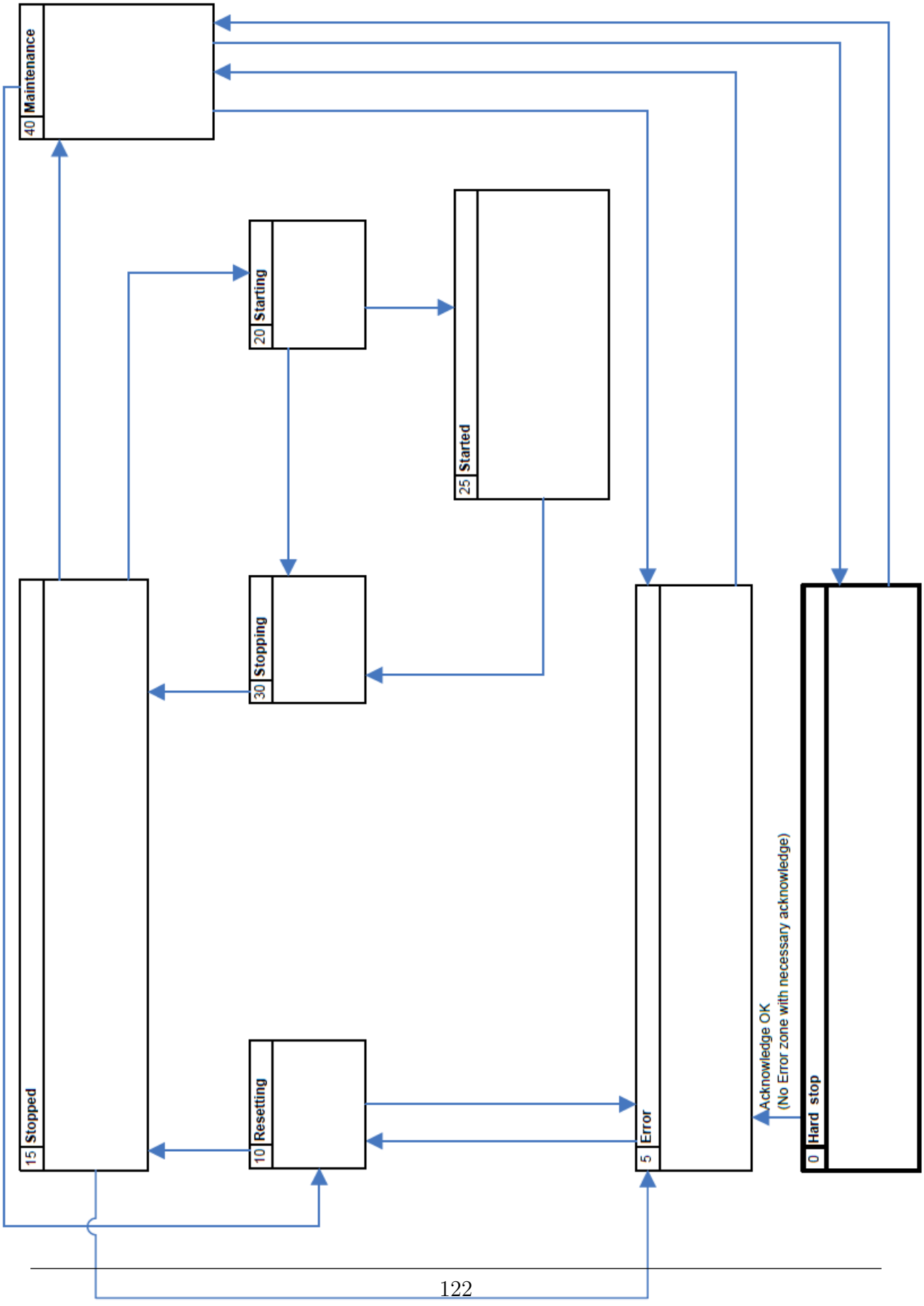
Table 2. Dictionary of machines

ID	Machine
1	PC
10	Main PLC

Table 3. Dictionary of phases and actions

Phases	Phase description	Action	Action description	Arguments	Argument description
sync	Pulse synchronization phase	stoptrig	PC asks the PLC to stop the triggers and start the sequence	/	/
		enabletrig	PC asks the PLC to enable the triggers	/	/
		stoptrigok	PLC notifies the PC that the triggers has been stopped	/	/
		enabletrigok	PLC notifies the PC that the triggers has been enabled	/	/

manual	Manual phase where the machines can be manually turned ON/OFF				
acq	Acquisition phase where the robots do not sort the wastes				


40 Maintenance
20 Starting
25 Started
30 Stopping
15 Stopped
10 Resetting
5 Error
0 Hard stop
Acknowledge OK
(No Error zone with necessary acknowledge)