Titre: Title:	Développement d'un contrôleur natif STEP-NC de machine-outil à commande numérique interopérable et compact
Auteur: Author:	Julien Bechtold
Date:	2022
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Bechtold, J. (2022). Développement d'un contrôleur natif STEP-NC de machine-outil à commande numérique interopérable et compact [Master's thesis, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/10459/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/10459/
Directeurs de recherche: Advisors:	Christophe Danjou, & Walid Jomaa
Programme: Program:	Maîtrise recherche en génie industriel

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Développement d'un contrôleur natif STEP-NC de machine-outil à commande numérique interopérable et compact

JULIEN BECHTOLD

Département de mathématiques et de génie industriel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*

Génie industriel

Août 2022

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Ce mémoire intitulé :

Développement d'un contrôleur natif STEP-NC de machine-outil à commande numérique interopérable et compact

présenté par **Julien BECHTOLD**

en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*

a été dûment accepté par le jury d'examen constitué de :

Bruno AGARD, président
Christophe DANJOU, membre et directeur de recherche
Walid JOMAA, membre et codirecteur de recherche
Roland MARANZANA, membre

REMERCIEMENTS

Je souhaite ici remercier un certain nombre de personnes qui ont joué un rôle clé durant cette maitrise.

Je souhaite tout d'abord remercier mon directeur de recherche, Christophe Danjou pour m'avoir accompagné le long de ces deux années et m'avoir apporté son soutien et sa confiance dans le déroulement de cette maitrise. Je tiens aussi à remercier mon co-directeur Walid Jomaa pour son expertise et ses connaissances. Merci à vous deux pour votre implication, votre bonne humeur et vos précieux conseils.

Je tiens aussi à remercier les techniciens de Polytechnique : Dominic De Lillis Gonçalves et Nicholas Veerabadren qui ont su prendre de leur temps pour m'apporter leurs aides, indispensables à la bonne réalisation des expérimentations.

Je souhaite remercier tous mes amis qui ont su rendre cette expérience au Canada inoubliable : Arthur, Aurèle, les trois Baptistes, Blanche, Florian, Jérôme, Louis et Nathan.

Enfin, un grand merci à mes parents pour le soutien qu'ils m'ont apporté.

RÉSUMÉ

Les machines-outils à commande numérique (MOCN) sont encore aujourd'hui commandées par le standard ISO 6983, appelé communément Code-G, qui a été inventé dans les années 1950 et qui n'a connu que très peu d'évolution. A contrario, les MOCN ont connu d'importantes innovations, le Code-G ne répond aujourd'hui plus aux exigences de la fabrication moderne et constitue un obstacle au développement de MOCN plus intelligentes et interopérables.

L'ISO 14649, communément appelée STEP-NC, est un langage de commande pour MOCN qui a été conçu pour remplacer les divers langages (dont le Code-G fait partie) qui composent aujourd'hui la chaine numérique d'industrialisation CAO - FAO - CN (Conception assistée par ordinateur, fabrication assistée par ordinateur, commande numérique) par un protocole de communication moderne, interopérable et normalisé. C'est une extension du standard ISO 10303 appelé STEP, qui permet d'intégrer toute la chaine numérique d'industrialisation dans un unique standard. Contrairement au Code-G, qui est un langage de bas niveau, incluant uniquement une faible quantité d'information (le parcours que doit suivre l'outil) spécifique à chaque machine et uniquement utilisable pour transmettre les informations entre le post-processeur et la MOCN, le STEP-NC est un langage de haut niveau qui permet un stockage d'une grande quantité d'information de haut niveau depuis la conception de la pièce jusqu'à sa fabrication, sans perte d'information.

Afin d'exploiter le STEP-NC, il faut concevoir de nouveaux contrôleurs de MOCN compatibles avec ce nouveau standard. En effet, le STEP-NC contient les informations de « quoi produire », mais très peu d'information de « comment produire », contrairement au Code-G qui contient uniquement le parcours d'outils spécifique à la machine. Le contrôleur STEP-NC doit alors interpréter ces informations et en déduire le parcours d'outils. Il est alors bien plus complexe et demande une plus grande puissance de calcul. Des prototypes de contrôleur STEP-NC sont en cours de développement depuis une vingtaine d'années et prouvent la viabilité et les avantages du STEP-NC, mais le manque de contrôleur est toujours considéré comme le principal problème qui limite l'utilisation du STEP-NC.

Ce projet de recherche se concentre sur le développement d'un contrôleur STEP-NC. L'état de l'art a permis de mettre en avant la complexité, le coût et l'encombrement des prototypes de contrôleur STEP-NC développés jusqu'ici, ce qui est un réel obstacle à l'utilisation et à la popularisation du

standard STEP-NC. L’objectif principal de ce projet de recherche est de développer un contrôleur STEP-NC interprété à architecture ouverte, en mettant l’accent sur la réduction du coût, de l’encombrement et en permettant d’avoir un contrôleur facilement reproductible. Pour cela, le contrôleur est basé sur un micro-ordinateur Raspberry Pi, peu cher et compact, qui peut être directement connecté à une MOCN. Le logiciel du contrôleur est codé en C++, peut interpréter un fichier ISO14649, générer un parcours d'outil, simuler le parcours d'outil, et envoyer l'information à la MOCN pour réaliser l'usinage. Une pièce de tournage issue de la norme est usinée sur un tour à commande numérique EMCO PC TURN 55 pour valider le fonctionnement de notre contrôleur. Les tests ont montré que le contrôleur est capable de produire une pièce issue du standard ISO 14649, sans utiliser de Code-G sur un Raspberry Pi, directement installée dans un tour à commande numérique.

Pour conclure, ce projet de recherche a permis de montrer qu’il était possible de réaliser un contrôleur STEP-NC interprété, facilement reproductible, peu coûteux et facilement implantable dans une MOCN commerciale existante.

ABSTRACT

Most of modern computer numerical control (CNC) machine tools (CNCMTs) are still controlled by the ISO 6983 standard, commonly known as Code-G, which was invented in the 1950s and has undergone very little evolution. On the other hand, CNC has undergone significant innovations, the G-Code no longer meets the requirements of modern manufacturing and is an obstacle to the development of more intelligent and interoperable CNC.

ISO 14649, commonly known as STEP-NC, is a control language for CNCMT that has been designed to replace the various languages (of which G-Code is a part) that make up today's CAD-CAM-CNC (Computer Aided Design, Computer Aided Manufacturing, Computer Numerical Control) industrialization digital chain with a modern, interoperable, and standardized communication protocol. It is an extension of the ISO 10303 standard called STEP, which allows the integration of the entire digital industrialization chain in a single standard. Unlike G-Code, which is a low-level language, including only a small amount of information (the path that the tool must follow) specific to each machine and only usable to transmit information between the post-processor and the CNCMT, STEP-NC is a high-level language that allows the storage of a large amount of high-level information from the design of the part to its manufacture, without loss of information.

To exploit STEP-NC, it is necessary to design new CNCMT controllers compatible with this new standard. Indeed, STEP-NC contains the "what to make" information, but not the "how to make" information, unlike G-Code which only contains the machine-specific toolpath. The STEP-NC controller must then interpret this information and deduce the tool path. It is therefore more complex and requires more computing power. STEP-NC controller prototypes have been under development for the past two decades and prove the viability and benefits of STEP-NC, but the lack of a controller is still considered the main hindrance that limits the use of STEP-NC.

This research project focuses on the development of a STEP-NC controller. The state of the art has highlighted the complexity, cost and cumbersome nature of the STEP-NC controller prototypes developed so far, which is a real obstacle to the use and popularization of the STEP-NC standard. The main objective of this research project is to develop an interpreted STEP-NC controller with an open architecture, focusing on cost reduction and reproducibility of the controller. For this purpose, the controller is based on a low cost and compact Raspberry Pi board that can be directly

connected to a MOCN. The controller software is coded in C++, can interpret an ISO14649 file, generate a toolpath, simulate the toolpath, and send the information to the CNCMT to perform the machining. A turning part from the standard is machined on an EMCO PC TURN 55 CNC lathe to validate the operation of our controller. The tests showed that the controller is able to produce a standard part without using G-code on a Raspberry Pi directly installed in a CNC lathe.

To conclude, this research project has shown that it was possible to realize an easily reproducible, inexpensive, and easily implemented STEP-NC controller in an existing commercial CNCMT.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

BOB	BreakOut Board
CAO	Conception Assistée par Ordinateur
CNC	Computer Numerical Control
CNCMT	Computer Numerical Control Machine Tool
CPU	Central Processing Unit
DCN	Directeur de Commande Numérique
FAO	Fabrication Assistée par Ordinateur
MLI	Modulation de largeur d'impulsion
MOCN	Machine-outil à commande numérique
NCK	Numerical Control Kernel
OAC	Open Architecture Controleur
PLC	Programmable Logic Control
RPi	Raspberry Pi
RTOS	Real Time Operating System
SBC	Single Board Computer
Soft-NC	Soft Numerical Control
STEP	Standard for the Exchange of Product model data compliant
STEP-NC	Standard for the Exchange of Product model data compliant Numerical Control

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION

Depuis l'invention des machines-outils à commande numérique (MOCN) dans les années 1950, leurs utilisations n'ont cessé d'augmenter dans l'industrie manufacturière. En effet, le marché des MOCN était estimé à 80 milliards USD en 2019 et il est prévu qu'il atteigne les 115 milliards USD d'ici 2026 (Facts&Factors, 2020). Les capacités et performances des MOCN ont beaucoup évoluées depuis leur invention, notamment par l'ajout d'axes de travail, l'augmentation de la vitesse de coupe, de la vitesse de déplacement ou encore l'amélioration de la précision.

Ces importantes innovations ont rendu la programmation de ces machines de plus en plus complexes. Le système de programmation appelé communément Code-G, créé dans les années 1960 et normalisé sous la référence RS274D en 1980 par l'ISO 6983 (1982) à lui, connu que très peu d'évolution. C'est pourtant ce langage, qui doit transmettre les informations générées par la conception assistée par ordinateur (CAO) et par la fabrication assistée par ordinateur (FAO), à la commande numérique (CN). Le Code-G est aujourd'hui un réel frein à l'évolution des MOCN : premièrement, du fait de la faible évolution de la norme, de nombreuses extensions et variantes ont été ajoutées par chacun des fabricants, ce qui rend aujourd'hui impossible l'exécution d'un même programme sur des MOCN différentes. Le Code-G est aussi un langage de bas niveau, peu compréhensible par un opérateur, et qui contient uniquement une faible quantité d'information (Suh, S.-H. et al., 2008). La machine exécute uniquement des déplacements et des actions sans avoir connaissance de l'opération réelle qui est effectuée. Enfin, son flux d'information est unidirectionnel, il ne permet pas le retour d'information de la fabrication vers la conception (Othman et al., 2017).

Dans une industrie de plus en plus compétitive et dans un contexte d'une industrie mondialisée, la complexité est à l'échange et à l'harmonisation des données, l'objectif étant d'atteindre le *« DA-BA-SA »* : *Design Anywhere, Build Anywhere, Support Anyways,* soit le fait de concevoir et de fabriquer n'importe où, et de pouvoir fournir de l'assistance (Dharmawardhana, M. et al., 2018). Les machines actuelles utilisant le Code-G ne peuvent atteindre ces objectifs, il est nécessaire d'étudier un nouveau moyen d'échange de données, qui permet le transfert d'informations bidirectionnel dans toute la chaine numérique d'industrialisation : CAO - FAO – CN.

Le projet de création d'un langage de donnée permettant de répondre aux exigences citées précédemment, a commencé en 1999 sous le nom de STEP compliant Numerical Control (STEP-NC) qui est une extension du standard STEP (Standard for the Exchange of Product model data), un standard d'échange de données de produit né dans le milieu des années 1980, normalisé au travers de l'ISO 10303 (1994) et structuré en plusieurs parties. STEP-NC en est une extension, standardisé en 2003 sous la norme ISO 14649 (2003b). L'objectif du STEP-NC n'est alors plus de transmettre qu'une petite quantité d'informations de comment produire (*how to make*), spécifique à chaque machine et à chaque procédé de fabrication, mais une grande quantité d'informations de haut niveau, décrivant quoi produire (*what to make*). Le processus de décision de comment produire est alors laissé au contrôleur de la machine-outil, au lieu d'être traité précédemment par un logiciel de FAO (Latif et al., 2021). Cela permet aussi de s'affranchir de l'étape du post-processeur qui traduit les instructions dans un langage spécifique à un unique contrôleur ou machine.

Depuis sa première version uniquement réservée au fraisage, le STEP-NC a évolué afin d'introduire de nouveaux modèles permettant de prendre en charge le tournage, l'électroérosion et la fabrication additive. Malgré les avancées effectuées sur le langage et l'ambition qui est placée dans le projet, 20 ans plus tard il n'est toujours pas utilisé dans l'industrie. Le principal frein à son intégration étant la nécessité d'adapter toute la chaine numérique de l'entreprise comprenant le logiciel de CAO, de FAO et le contrôleur de machines-outils. Mais beaucoup d'entreprises, telles que Airbus, Boeing ou encore Scania, s'y intéressent de près, du fait des opportunités que ce projet peut apporter dans la productivité de l'entreprise (Cha et al., 2016). En effet, l'utilisation de STEP-NC permet d'ouvrir des portes à de nombreux projets permettant d'améliorer considérablement la productivité, on peut citer par exemple l'utilisation des données de mesure pour adapter les paramètres de fabrication, appelé Closed-loop manufacturing (Po et al., 2014) ou l'utilisation d'un même fichier STEP-NC pour produire sur des machines de technologie différentes, appelé multiprocess manufacturing (Rauch, M. & Hascoet, 2012).

Il n'existe actuellement aucun contrôleur commercial compatible avec STEP-NC. Bien qu'il existe aujourd'hui une cinquantaine de prototypes de contrôleur STEP-NC pour un usage académique exclusivement, la majorité permet uniquement une traduction du STEP-NC en Code-G (Contrôleur de type 1). Le manque de contrôleur de type 2 et 3, c'est-à-dire permettant d'interpréter nativement le STEP-NC (type 2) et qui possède des fonctionnalités dites intelligentes et collaboratives (type 3), est considéré comme étant l'obstacle le plus important à l'implantation de STEP-NC dans

l'industrie (Othman et al., 2017). De plus, les prototypes de contrôleurs de type 2 qui ont été développés jusqu'à aujourd'hui, ont une architecture complexe, sont peu flexibles et sont onéreux, ce qui est frein à leur implantation dans l'industrie (Danjou, 2015). Les fabricants de machines-outils étant fermés et conservateurs, l'implantation du STEP-NC dans l'industrie ne peut passer que par le développement de contrôleurs innovants permettant de montrer les avantages d'un contrôleur STEP-NC. En effet, la partie contrôleur d'une MOCN représente environ 30% de la valeur de la MOCN, ce qui rebute l'industrie des machines-outils à réaliser d'importante modification (Rauch, Matthieu et al., 2014). C'est donc un réel enjeu économique et opérationnel de vouloir profondément modifier les contrôleurs de MOCN. Dans un contexte d'industrie 4.0, l'industrie manufacturière se prive alors d'un potentiel d'innovation très important, celle de pouvoir intégrer les MOCN dans une architecture plus intelligente, capable d'acquérir et de partager des données et permettant l'amélioration de la productivité via des MOCN totalement intégrées à l'ensemble de la chaine numérique d'industrialisation.

L'objectif principal de cette recherche est donc de développer un contrôleur STEP-NC pour un tour à commande numérique, permettant d'interpréter et d'exécuter des instructions en STEP-NC en employant une architecture flexible, facilement reproductible et peu coûteuse. Cela permettra de montrer la viabilité et les avantages de STEP-NC et ouvrira par la suite les possibilités de développement de nouvelles fonctionnalités et de nouveaux produits innovants utilisant STEP-NC.

Ce mémoire est structuré de la façon suivante : premièrement, le chapitre 2 présentera une revue de littérature sur les standards d'échange de données pour la programmation des MOCN et un état de l'art des principaux projets de contrôleur STEP-NC sera dressé en présentant leurs technologies leurs points forts, mais aussi leurs limitations. Le chapitre 3 présentera la méthodologie de recherche qui sera utilisée. Le chapitre 4 portera sur le développement matériel et logiciel d'un contrôleur STEP-NC. Le chapitre 5 décrira l'implantation et les tests réalisés afin de valider expérimentalement le développement. Finalement, le chapitre 6 conclura ce mémoire après avoir proposé une discussion sur les travaux menés.

CHAPITRE 2 REVUE DE LITTÉRATURE

Ce chapitre permet de prendre connaissance du contexte de la programmation des MOCN et réalise un état de l'art des contrôleurs STEP-NC présents dans la littérature. Dans un premier temps, les notions liées à la chaine numérique de fabrication et à ses problématiques seront présentées. Cela permettra d'aborder le standard STEP-NC et son implication dans la modification de la chaine numérique d'industrialisation actuelle. Nous poursuivrons ensuite sur une présentation de la notion de contrôleurs à architecture ouverte qui est une nécessité pour l'usage de STEP-NC. Dans une quatrième partie, nous réaliserons un état de l'art des contrôleurs STEP-NC et réaliserons une revue systématique de la littérature, ce qui nous permettra de conclure sur une revue critique de l'existant en faisant ressortir les manques de la littérature.

2.1 La chaine numérique d'industrialisation

2.1.1 Une chaine numérique discontinue

La chaine numérique de fabrication se compose de différents maillons qui permettent la fabrication d'une pièce à partir de sa conception sur ordinateur. Entre chaque étape, il existe des fichiers qui permettent l'échange de données. Il existe actuellement une multitude de types de fichier entre chacune des étapes, dont certains normalisés comme le STEP entre la CAO et la FAO ou d'autres qui sont propriétaire comme le CATpart de CATIA. Le Code-G est le standard qui permet actuellement le transfert d'informations entre la partie FAO et la MOCN. Le Code-G est généré par un post-processeur qui transforme les informations présentes dans le fichier en sortie de FAO en informations spécifiques pour le contrôleur de la MOCN. La chaine numérique de fabrication est résumée dans la Figure 2.1.

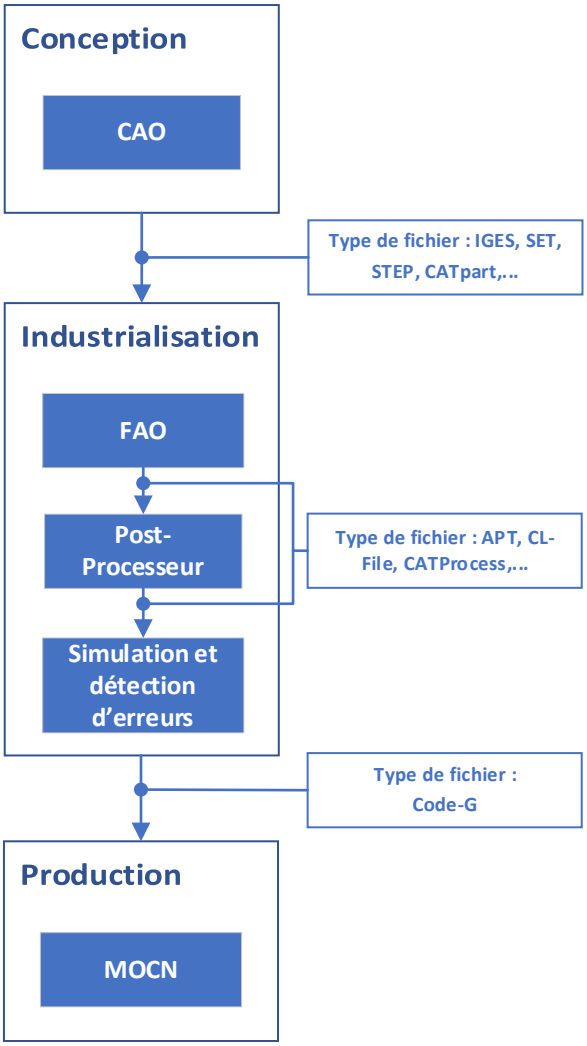


Figure 2.1 Chaine numérique de fabrication actuelle, adaptée de (Danjou, 2015; Hamilton et al., 2014)

On remarque que les échanges de données sont problématiques dans cette chaine numérique, car il n'y a pas une intégration complète des maillons dans un unique standard. De plus, il existe une perte importante d'informations, notamment après l'étape de post-processeur du fait de la nécessité d'utiliser du Code-G, qui ne supporte que très peu d'information, pour piloter les MOCN.

Dans la chaine numérique d'industrialisation actuelle, il faut séparer le fichier en sortie du logiciel de CAO ou de FAO pour chaque type de procédé différent, voir même pour chaque machine différente, puisqu'il existe des incompatibilités entre les différents contrôleurs de MOCN. Ce fichier sera ensuite traité séparément, pour chaque machine en fonction du logiciel de

FAO qui est adapté avec cette dernière. La Figure 2.2 schématise le processus pour une chaine numérique actuelle utilisant le Code-G.

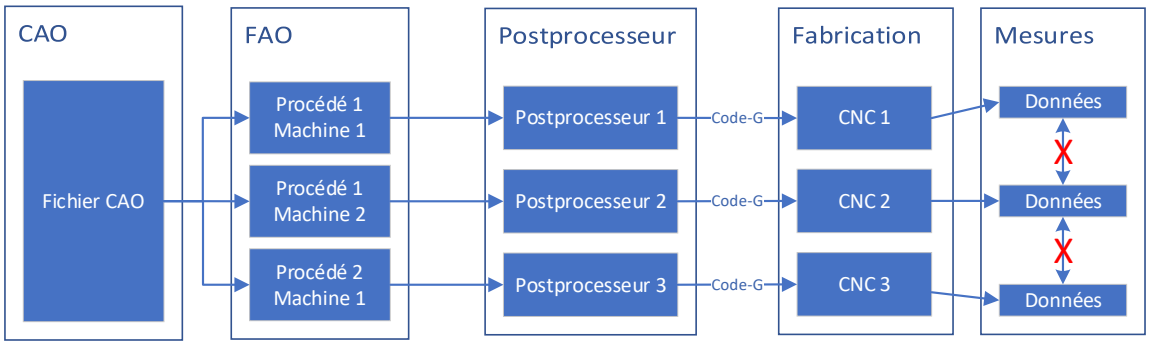


Figure 2.2 Chaine numérique actuelle pour une production multi procédé utilisant le Code-G, adapté de (Rauch, M. & Hascoet, 2012)

2.1.2 La problématique du Code-G dans la chaine numérique d'industrialisation

Le Code-G possède de nombreuses limitations qui empêchent aujourd'hui l'amélioration de la chaine numérique (Othman et al., 2017):

- Un langage de bas niveau : Le Code-G contient uniquement les informations de bas niveau, comme la position des axes, leur vitesse et l'activation des actionneurs. En revanche, aucune information sur la nature des opérations d'usinage qui seront effectuées n'est présente. La MOCN ne fait qu'exécuter le code sans connaitre les mouvements qui y sont effectués.
- Manque d'interopérabilité : Comme nous l'avons vu dans son historique, le Code-G n'a pas suivi l'évolution des machines-outils, chaque constructeur ajoute donc ses extensions et variantes pour rendre le Code-G compatible avec sa machine (par exemple des fonctions préparatoires G et auxiliaires M spécifiques comme chez Fanuc et Siemens). Il existe alors aujourd'hui plus de 5000 dialectes de Code-G différent (Monkova et al., 2019). Le post-processeur est alors nécessaire afin de rendre le code compatible, mais aussi spécifique à un contrôleur de MOCN. Il n'est alors plus possible d'utiliser le même code pour le faire fonctionner sur une machine dotée d'un autre contrôleur.

- Perte d'informations : Le Code-G perd la majorité des informations qui étaient contenues dans le fichier CAO et FAO comme la géométrie de la pièce, la géométrie des outils, les éléments à usiner (*features)* ou encore les opérations qui seront effectuées. De plus, le langage étant de bas niveau, il peut être compliqué à comprendre par un opérateur. En effet, le parcours d'outils étant défini point par point, il devient difficile à comprendre pour des pièces avec une géométrie complexe. La modification du programme au niveau de l'atelier devient alors peu intuitive. Il faut alors repasser par les étapes de FAO et de post-processeur pour effectuer des modifications sur un programme complexe.
- Unidirectionalité du flux d'information : il n'est pas possible avec le Code-G de faire remonter des informations depuis la MOCN vers la FAO ou la CAO pour effectuer des modifications. De la même manière, il est très complexe d'optimiser la production ou de corriger des erreurs d'une pièce à partir des informations obtenues lors de l'usinage. En effet, il faut soit modifier le fichier directement sur la machine, ce qui est complexe au vu du langage, ou alors repasser par la FAO ou la CAO et remodifier tous les fichiers Code-G qui ont été générés spécifiquement pour chaque MOCN.

L'ensemble de ces limitations ont mené à la recherche de nouveaux standards de programmation afin d'intégrer toute la chaine numérique d'industrialisation dans un même standard et éliminer toutes les limitations qui empêchent l'amélioration du flux d'information et l'optimisation de la chaine numérique d'industrialisation. Dans la section suivante, nous allons étudier le STEP-NC comme outil pour intégrer toute la chaine numérique de fabrication dans un même standard.

2.1.3 Une nouvelle chaine numérique d'industrialisation à l'aide de STEP-NC

2.1.3.1 STEP-NC : historique

Le STEP-NC a été développé afin de répondre aux problématiques liées à la chaine numérique de fabrication que nous venons d'énoncer. Le projet STEP-NC, pour STEP Numerical Control, est une extension d'un standard déjà existant et utilisé, le STEP (Standard for the Exchange of Product model data). La norme STEP, publiée sous l'ISO 10303 (1994) est un standard pour l'échange de données produits qui est déjà implanté pour le partage des données entre la CAO et d'autres technologies assistées par ordinateur comme la FAO par exemple.

Le développement a débuté en 1999 et la norme STEP-NC a été développée par plusieurs projets de recherche, principalement par le projet OPTIMAL (Optimised Preparation of Manufacturing Information with Multi-Level CAM-CNC Coupling) de l'initiative européenne ESPRIT puis par le projet international IMS (Intelligent Manufacturing System). De nombreux partenaires industriels ont participé à son développement dans différents domaines : CAO/FAO (Dassault, OpenMind), contrôleur (Siemens, Fanuc), machine-outil (CMS) ainsi que des utilisateurs (Chrysler, Volvo) (Xu et al., 2005). La norme a finalement été publiée sous le standard ISO 14649 (2003b) et sous le protocole d'application 10303-AP238 (2007).

La norme STEP-NC a ensuite été améliorée pour prendre en compte de nouveaux procédés de fabrication. La première version prenant uniquement en compte le fraisage, les versions suivantes ont ajouté le tournage, l'électroérosion et plus récemment une version pour la fabrication additive. Son développement est toujours en cours.

Il existe actuellement deux méthodes d'implémentations de STEP-NC qui sont développées simultanément par ISO (Lan et al., 2008):

- L'ISO 14649 qui concerne le niveau ARM (Application Reference Model) de la norme, en définissant le modèle conceptuel de la norme STEP-NC.
- L'ISO 10303 via le protocole d'application AP238 définit le niveau AIM (Application Interpreted Model) en définissant le modèle d'implémentation de la norme STEP-NC

Ces deux versions de STEP-NC se différencient en fonction de leurs cas d'utilisation. L'ISO 14649 sera plus utilisée pour une implémentation dans un environnement technique proche des données des MOCN alors que l'AP-238 sera plus adapté à une intégration complète entre CAO, FAO et MOCN, ce qui le rend plus complexe a utilisé (Lan et al., 2008). La comparaison des deux modèles a été résumée dans le Tableau 2.1 par (Xu et al., 2005).

Tableau 2.1 Comparaison entre ISO 14649 et ISO 10303-AP238 (Xu et al., 2005)

Critère de comparaison	ISO 14649 (ARM)	ISO 10303-AP238 (AIM)
Espace de stockage	~10 fois moins que AIM	~10 fois plus que ARM
Programmation	Facile	Plus complexe
Lisibilité par un humain	Difficile	Quasi impossible
Compatibilité avec STEP	Partiellement conforme	Totalement conforme
Cohérence des données	Les informations sur la conception originale sont abandonnées	Les informations sur la conception originale sont conservées

2.1.3.2 Une intégration complète de la chaine de fabrication grâce à STEP-NC

Le STEP-NC a pour but d'étendre le standard STEP, pour permettre le partage des données jusqu'à la MOCN en intégrant les informations en vue de la fabrication. L'objectif est d'intégrer le STEP-NC tout au long de la chaine CAO-FAO-CN pour qu'il n'y ait aucune perte d'informations entre la CAO et la MOCN, ce qui n'est pas le cas dans la chaine actuelle. Un seul fichier est alors nécessaire de la conception jusqu'à la MOCN, en autorisant un flux bidirectionnel des données. De plus, l'utilisation d'un post-processeur n'est plus utile, puisque le langage est totalement standardisé et c'est au contrôleur de la MOCN d'interpréter les données contenues dans le fichier STEP-NC aux vues de la fabrication. La chaine numérique d'industrialisation utilisant STEP-NC est schématisée Figure 2.3 en comparaison à la chaine numérique actuelle utilisant du Code-G, présentée précédemment Figure 2.1. On remarque bien l'usage du standard STEP-NC tout au long de la chaine numérique d'industrialisation qui intègre et étend le standard STEP déjà utilisé entre les étapes de CAO et de FAO.

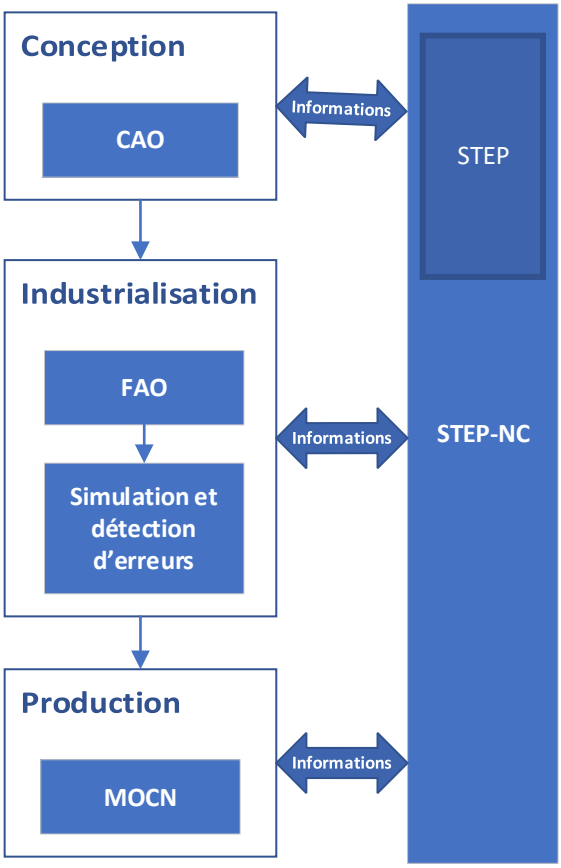


Figure 2.3 Chaine numérique de fabrication utilisant le STEP-NC, adapté de Rauch, Matthieu et al. (2009)

Le STEP-NC est alors une réponse aux limitations du code G (Othman et al., 2017) :

- Un langage de haut niveau : Un fichier STEP-NC contient des informations de haut niveau, telles que la géométrie de la pièce brute et finie, la définition des géométries d'usinage sous forme d'éléments à usiner, les opérations d'usinage avec leurs stratégies ou encore des indications sur les outils à utiliser. Ces informations de haut niveau permettent au contrôleur de la MOCN de faire des choix (vitesse d'avance, outils ou parcours d'outils le plus approprié en fonction des informations disponibles) et d'avoir connaissance des actions qui sont effectuées (opération, géométrie, les paramètres utilisés…) et ainsi de pouvoir les rectifier ou optimiser les paramètres automatiquement, sans avoir besoin d'un expert ou d'un système externe à la MOCN.
- Interopérabilité : En supprimant la nécessité d'avoir un post-processeur et en utilisant un même fichier et un même langage standardisé pour toutes les machines et tous les logiciels

utilisés le long du cycle de vie du produit, on assure ainsi une interopérabilité qui permet la communication et l'échange des données de la conception à la fabrication.

- Intégrité de l'information : Il n'y a aucune perte d'information de la conception à la fabrication du produit. Un unique fichier STEP-NC comporte toutes les informations de la CAO à la MOCN.
- Bidirectionnalité du flux d'information : La remontée d'information est possible, on peut par exemple apporter des modifications sur la géométrie de la pièce après mesure des dimensions sur une MOCN sans avoir à repasser par la CAO. Ce fichier pourra ensuite être directement utilisé sur d'autres machines.

Le STEP-NC permet alors d'optimiser le temps pour passer de la conception à la fabrication. On peut estimer un gain de 35% dans le temps de programmation de la MOCN, une réduction de 75% du nombre d'itérations de dessin qui doivent être envoyés à la production et une réduction du temps d'usinage de 50% pour les lots de petite et moyenne taille (Lan et al., 2008).

2.1.3.3 Structure de la norme STEP-NC

Dans la suite du mémoire, nous utiliserons la norme ISO 14649 pour des raisons de facilité de compréhension. De plus, nous utiliserons le format physique de fichier ISO 10303 part 21 qui utilise une syntaxe en langage Express. Il existe aussi une syntaxe en XML (ISO 10303 part 28), mais qui n'est pas pertinente dans notre cas, puisque moins compréhensible par un humain. Ces choix seront justifiés dans le chapitre 4.

La norme ISO 14649 est structurée en 14 parties listées dans le Tableau 2.2.

Tableau 2.2 Structure de la norme ISO 14649

Partie de la norme ISO 14649	Description
Part 1	Overview and fundamental principles
Part 2	Language bindings, Fundamentals
Part 3	Language binding in Java
Part 4	Process data for cutting
Part 5	Tools for cutting operations
Part 10	General process data
Part 11	Process data for milling
Part 12	Process data for turning
Part 13	Process data for wire-EDM
Part 14	Process data for sink-EDM
Part 17	Process data for additive manufacturing
Part 111	Tools for milling
Part 121	Tools for turning
Part 201	Machine tool data for cutting processes

Le STEP-NC permet le stockage des informations de haut niveau sous forme de données orientées objet. Les données contenues correspondent à des informations sur ce qui doit être usiné, et non sur comment cela doit être usiné.

Un fichier STEP-NC est composé de deux parties. Un entête (*HEADER)*, qui contient les informations sur le programme d'usinage, comme le nom du programmeur, le numéro du programme, des caractéristiques du produit. Une section *DATA*, qui comporte tout le programme d'usinage, c'est-à-dire la séquence à usiner, les éléments à usiner, les outils utilisés, la géométrie de la pièce… Un exemple de programme d'usinage est illustré Figure 2.4.

Chaque ligne de la section *DATA* correspond à une entité qui possède un numéro unique qui l'identifie, précédé d'un symbole « # ». Chaque entité possède ensuite une ou plusieurs propriétés, obligatoires ou optionnelles, qui la caractérisent. Les propriétés peuvent être définies soit explicitement par un nombre ou une chaine de caractère par exemple, soit par un nombre qui fait référence à une autre entité. Le symbole « $ » signifie qu'une propriété optionnelle n'est pas donnée.

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(('piece exemple pour tournage : piece02'),'');
FILE_NAME('piece02','2021-07-01T11:32:00-05:00',('Julien Bechtold'),('Polytechnique Montreal'),'','',
ENDSEC;

DATA;

/*Project*/
#1=PROJECT('EXEMPLE TOURNAGE 02',#2 ,(#100) ,#3,$,$);
#2=WORKPLAN('MAINWORKPLAN',(#300,#301) ,$,#200 ,$);

/*Workpiece*/
#100=WORKPIECE('Workpiece01',#101,0.010,#103,#1065,$,());
#101=MATERIAL('DIN EN 10027-1','E 295',(#102));
#102=NUMERIC_PARAMETER('ELASTIC MODULUS',2.E11,'pa');

#103=WORKPIECE('Rawpiece01',#101,0.010,$,#2065,$,());

/*Setup*/
#200=SETUP('SETUP POUR EXEMPLE DE TOURNAGE 02',$,#700,(#201));
#201=WORKPIECE_SETUP(#100,#710 ,$,$,());

/*Workinstep*/
#300=MACHINING_WORKINGSTEP('Dressage ebauche 01',#700 ,#400,#500,$);
#301=MACHINING_WORKINGSTEP('Dressage finition 01',#700 ,#400,#501,$);

/*Feature*/
#400=REVOLVED_FLAT('FACE',#100,(#500,#501),#730 ,#732 ,0.000,#733);

/*Operation*/
#500=FACING_ROUGH($,$,'Operation dressage ebauche 01',$,$,#600 ,#801 ,#800 ,#910 ,#911 ,#900 ,0.500);
#501=FACING_FINISH($,$,'Operation dressage finition 01',$,$,#610 ,#803 ,#800 ,#910 ,#911 ,#901 ,$);

/*Tool*/
#600=TURNING MACHINE TOOL('outil ebauche',#601,(#602),$,$,$);
```

Figure 2.4 Exemple d'un programme STEP-NC

Le programme est ensuite structuré en entités : L'entité *PROJECT* est la racine, présente dans chaque programme, elle permet de définir le *WORKPLAN* principal qui sera utilisé, c'est-à-dire le plan de fabrication qui comprend toutes les étapes d'usinage. Il comprend aussi le ou les *WORKPIECE*, c'est-à-dire les pièces qui seront usinées ainsi que d'autres propriétés utiles à la réalisation de la pièce. Un *WORKPLAN* est ensuite constitué d'un ou plusieurs *WORKINGSTEP* (ou d'autre *WORKPLAN*) qui définissent chacun une unique opération machine. Un *WORKINGSTEP* est ensuite notamment constitué de deux éléments, une *OPERATION* qui définit l'opération machine qui devra être effectuée et une *FEATURE* qui définit la géométrie à usiner. D'autres paramètres sont présents et permettent de définir des repères, des vecteurs et d'autres paramètres géométriques utiles au placement et à la définition de l'entité. La structure du programme continue de la même manière pour stocker toutes les informations qui ont besoin d'être définies comme les outils utilisés, leurs géométries, la position des pièces, les stratégies utilisées pour les opérations…

On peut résumer quelques entités utilisées dans un fichier STEP-NC Figure 2.5 sous forme d'arborescence.

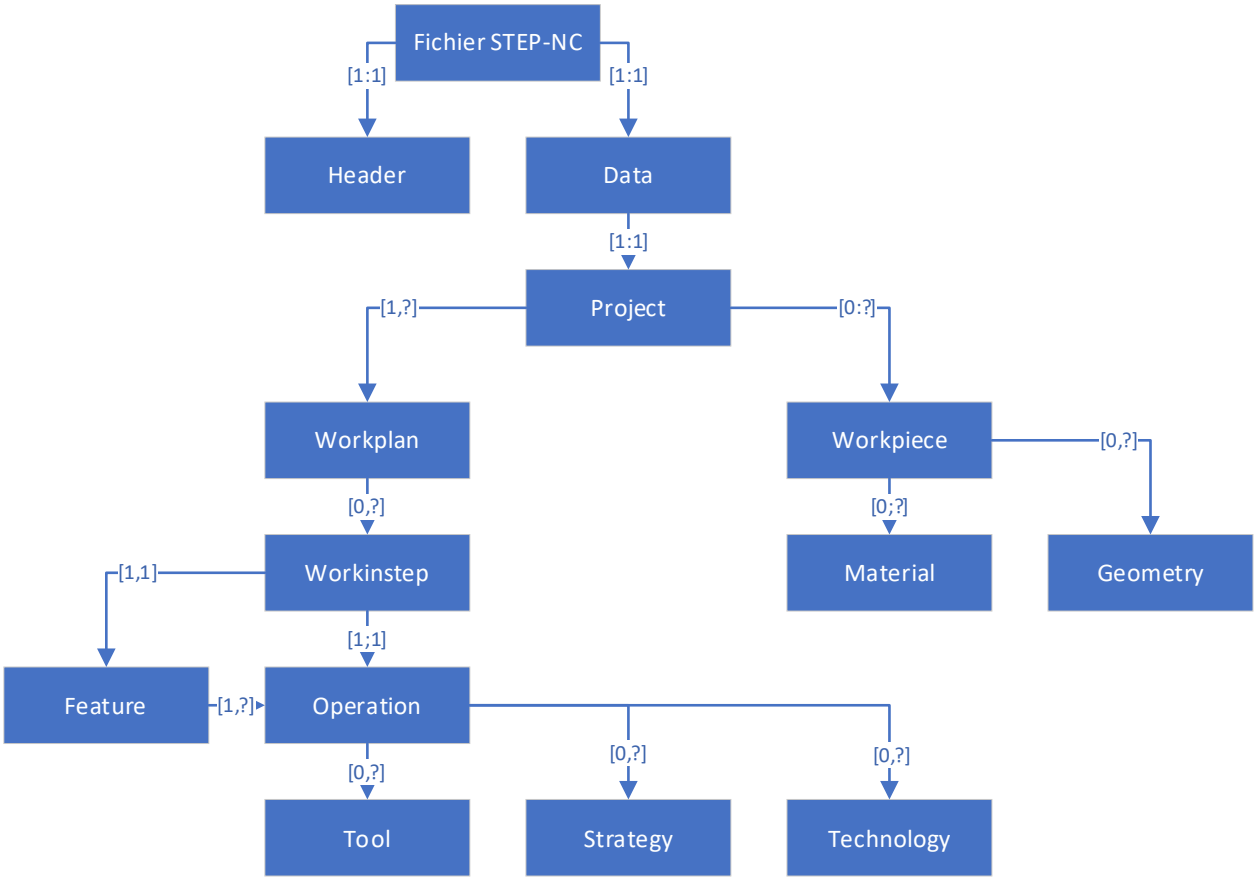


Figure 2.5 Structure du modèle de donnés STEP-NC, issue de International Organization for Standardization (2003a)

Pour intégrer STEP-NC dans la chaine numérique, il est nécessaire de revoir complètement l'architecture de plusieurs maillons de la chaine. Il faut notamment revoir l'architecture des contrôleurs de MOCN qui sont pour le moment fermés et uniquement capables d'interpréter du Code-G. Il faut s'intéresser à de nouveaux contrôleurs dits à architecture ouverte, pour qu'ils puissent être capables d'accueillir le langage STEP-NC et d'exploiter ses bénéfices. La section suivante s'intéresse aux contrôleurs à architecture ouverte.

2.1.4 Contrôleur à architecture ouverte

2.1.4.1 Définition et objectif

Les contrôleurs présents actuellement dans les MOCN sont des contrôleurs dits à architecture fermée. En effet, ils se comportent comme des « boites noires », ce qui rend difficile le développement de nouvelles fonctionnalités pour la MOCN (Yusof & Latif, 2015b). De plus, ils ne sont souvent pas compatibles avec d'autres logiciels ou services que l'entreprise voudrait utiliser et la réparation ou la maintenance est aussi très coûteuse. Pour ajouter de nouvelles fonctionnalités spécifiques, il faut passer par le fabricant du contrôleur de la MOCN, ce qui est très coûteux et demande une longue durée de développement (Rauch, Matthieu et al., 2014).

Pour contrer cette problématique, de nouveaux types de contrôleurs à architecture ouverte doivent être utilisés, appelés « Open Architecture Contrôleur » (OAC). Cela signifie que la partie matérielle du contrôleur peut être achetée séparément de la partie logicielle et ensuite être installée sur n'importe quel contrôleur de MOCN. Pour l'instant, il n'existe que très peu de contrôleurs OAC, la majorité étant uniquement des prototypes pour un usage académique.

2.1.4.2 Historique des contrôleurs à architecture ouverte

Le développement de contrôleur OAC a débuté à la fin des années 1980 en utilisant un PC comme base matérielle pour le contrôleur. L'avantage est d'avoir un composant matériel très facile à programmer afin d'ajouter des fonctionnalités au contrôleur. Le principal problème de l'époque était la faible puissance des PC, en effet, les contrôleurs commerciaux utilisant souvent une architecture avec des composants électroniques spécialisés et plusieurs CPU pour effectuer certaines fonctions comme le noyau de contrôle du contrôleur. Un contrôleur basé sur PC doit alors effectuer tous ces calculs de manière logiciel et non plus matériel. On parle alors de « Soft-NC » pour Software Numerical Control (en opposition à « Hard-NC » pour Hardware). Il fallait alors utiliser deux PC pour arriver à réaliser toutes les tâches du contrôleur. Ce problème a pu être réglé dans les années 2000 avec l'amélioration de la puissance des CPU (Suh, S.-H. et al., 2008).

Les principaux projets de contrôleur OAC ont été listés par Hascoet et Rauch (Hascoet & Rauch, 2016). On retrouve actuellement très peu de contrôleurs OAC présent dans des MOCN commerciales, en effet le contrôleur représente environ un tiers du prix d'une MOCN ce qui pousse les fabricants à garder leurs contrôleurs fermés pour ne pas divulgué leur savoir-faire (Rauch,

Matthieu et al., 2014). En revanche, on trouve des contrôleurs OAC basés sur PC pour moderniser des MOCN vieillissantes ou dans la conception de MOCN personnelles en utilisant notamment le logiciel libre LinuxCNC qui constitue la partie logicielle du contrôleur.

2.1.4.3 Avantages et limitations d'un contrôleur OAC

Un contrôleur OAC permet d'apporter de nombreuses solutions à des MOCN qui cherchent à être de plus en plus intégrées et qui souhaitent avoir plus de flexibilité (Hascoet & Rauch, 2016):

- La portabilité d'un contrôleur OAC permet à une application développée pour un contrôleur d'être utilisée sur un autre, sans modifications.
- Son extensibilité permet d'ajouter à tout moment des modules permettant d'apporter de nouvelles fonctionnalités au contrôleur
- L'interopérabilité permet au contrôleur de communiquer avec d'autres machines ou d'autres logiciels utilisés dans l'entreprise.

Pour atteindre ces spécifications, un contrôleur OAC ne doit pas être spécifique à un fabricant, il doit être basé sur une architecture standardisée, aussi bien matérielle que logicielle.

Les limitations des contrôleurs OAC sont principalement dues à son développement qui est encore récent. Bien que des contrôleurs OAC soient disponibles à la vente, ils sont pour l'instant réservés à un usage non industriel. Les contrôleurs OAC industriel sont encore au stade de prototype pour les mêmes raisons que celles évoquées pour le développement des contrôleurs STEP-NC (fermeture des MOCN actuelles et conservatisme de la part des constructeurs) et sont utilisés dans un cadre principalement académique. Il y a donc encore une nécessité de développement pour rendre les OAC viables à un usage commercial. En revanche, on remarque depuis une dizaine d'années une tendance grandissante dans le développement et l'utilisation de contrôleur OAC (Latif et al., 2021).

L'utilisation d'un contrôleur ouvert est donc primordiale dans la conception d'un contrôleur STEP-NC. En effet, c'est la philosophie même de la norme STEP-NC, de réaliser une architecture ouverte, indépendante du fabricant de la MOCN afin de permettre une amélioration permanente aussi bien matérielle que logicielle dans l'optique d'optimiser les moyens de production. Un contrôleur fermé ne permettrait pas d'utiliser tout le potentiel du langage STEP-NC. Nous allons

étudier dans la partie suivante les propositions de contrôleurs STEP-NC qui ont été faites au cours de son développement.

2.2 État de l'art des contrôleurs STEP-NC

2.2.1 Stratégie de recherche

Une revue systématique de littérature a été effectuée grâce aux bases de données Web of Science, Compendex et Inspec afin de répertorier les travaux sur les contrôleurs STEP-NC ainsi que d'étudier leur approche et leurs technologies.

En recherchant uniquement « STEP-NC » dans les bases de données Web of Science, Compendex et Inspec, on trouve 345 articles, cependant une partie de ces documents ne correspond pas à la zone de notre recherche puisque STEP-NC s'étend sur toute la chaine numérique CAO-FAO-CN. Le développement de STEP-NC peut être décomposé en 4 zones de recherche (Zhao et al., 2008) :

- La traduction des données géométriques en fonction machine (feature recognition)
- La planification des processus de fabrication (process planning)
- La construction de modèle de données de contrôle (inspection data)
- L'interprétation et l'exécution du code STEP-NC sur une MOCN.

C'est uniquement cette dernière zone qui nous intéresse dans notre recherche.

Un plan de concept, résumé dans le Tableau 2.3, a ensuite été élaboré afin de réaliser une recherche exhaustive et pertinente par rapport au sujet de notre recherche qui comporte trois concepts clés :

1. Dans un premier temps, on introduit la notion de machine-outil à commande numérique pour orienter la recherche vers des prototypes appliquée aux machines-outils.
2. Ensuite, la notion de contrôleur ou d'interpréteur qui délimite la zone de recherche STEP-NC sur laquelle le projet va se développer.
3. Enfin la notion de « STEP-NC », le nom du standard au cœur de la recherche, qui peut prendre plusieurs dénominations selon la norme et l'écriture utilisées.

Tableau 2.3 Plan de concept de la revue de littérature

Concept 1 : MOCN	Concept 2 : Contrôleur	Concept 3 : STEP-NC
CNC NC Machin* “Numerical control*” Lathe* Mill* EDM “Electric discharge machining” “Machine tool” Manufact*	Control* Interpret* Convert* Translat* Prototy* STEP-CNC STEPcNC “CNC system”	STEP-NC STEPNC STEP-compliant “ISO 14649” ISO14649 “ISO 10303-238” “ISO 10303 AP238” “ISO10303 AP238” ISO10303AP238 AP238

D’après le plan de concept, la recherche suivante a été effectuée :

> TS=((STEP-NC OR STEPNC OR STEP-Compliant OR ISO14649 OR ‘’ISO 14649’’ OR ‘’ISO 10303-238’’ OR ‘’ISO 10303 AP238’’ OR ‘’ISO10303 AP238’’ OR ISO10303AP238 OR AP238)
>
> AND (Control* OR Interpret* OR Convert* OR Translat* OR Prototy* OR STEP-CNC OR STEPcNC OR "CNC systems" or "CNC system")
>
> AND (CNC OR NC OR Machin* OR “Numerical Control*” OR Mill* OR Lathe* OR EDM OR “Electric discharge machining” or “Machine tool” or Manufact*))

La recherche a été effectuée le 16 juin 2021, puis un suivi des publications a été réalisé via des alertes basées sur le même plan de concept. Aucune limitation de langue n’a été imposée, tous les articles présents étant en anglais. Les articles de conférences et de journaux ont été sélectionnés avec une limitation des dates entre 1999 (début des travaux sur STEP-NC) et nos jours.

On obtient ainsi 233 résultats. Après lecture des titres et résumés, un article est retenu s’il traite directement de la conception ou de la réalisation d’un contrôleur. En effet, une partie des articles s’attardent sur d’autres aspects de la recherche STEP-NC et ne font que citer la notion de contrôleur, ils n’ont donc pas été conservés. On obtient ainsi 92 articles. Après lecture complète des articles, on conserve 81 articles. Avec une analyse plus profonde durant notre projet, 37 articles seront finalement pertinents pour notre recherche. La méthodologie de la stratégie de recherche documentaire est résumée dans la Figure 2.6.

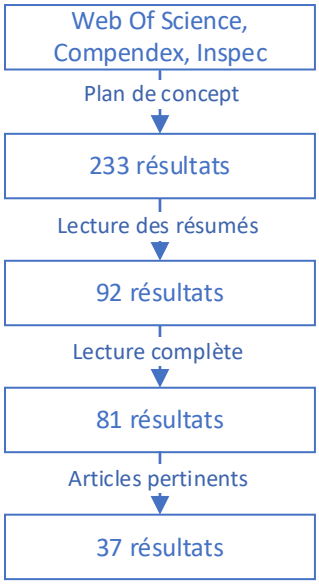


Figure 2.6 Méthodologie de la stratégie de recherche

Ces articles ont ensuite été traités plus en profondeur et triés par projet (il y a souvent plusieurs articles traitant d'un même projet)

Une revue de littérature similaire réalisée en 2021 par Latif et al. (2021) nous a aussi permis de vérifier la pertinence et l'exhaustivité de notre recherche.

La section suivante détaillera les différents types de contrôleurs STEP-NC existant ce qui permettra de classer les projets trouvés dans cette section.

2.2.2 Les contrôleurs STEP-NC de MOCN

Pour commander une MOCN, on utilise une séquence d'instruction qui sera interprétée par un contrôleur, aussi appelé directeur de commande numérique (DCN) afin d'agir sur les actionneurs et de recevoir les informations des différents capteurs. Comme vu précédemment, les MOCN sont actuellement commandées par des instructions utilisant le langage Code-G. Les contrôleurs sont donc exclusivement capables d'interpréter du Code-G, qui de plus, doit être compatible avec le contrôleur utilisé puisqu'une partie des commandes est spécifique à chaque contrôleur. Il n'existe actuellement aucun contrôleur commercial permettant d'interpréter le langage STEP-NC.

En opposition au Code-G, qui contient uniquement des informations de bas niveau telles que des données géométriques et des fonctions simples, le STEP-NC contient une grande quantité d'information de haut niveau. Le rôle du contrôleur est alors bien plus complexe, puisqu'il ne suffit

plus de lire les instructions de déplacement, mais il faut interpréter les informations pour en déduire notamment le parcours d'outils, les actions de la machine ou encore les outils à utiliser.

Les contrôleurs STEP-NC peuvent être classés en trois types selon leurs niveaux d'implémentation de STEP-NC :

- Type 1, contrôleur indirect : Le principe est ici **d'interpréter le code STEP-NC et de le traduire en Code-G** afin de le rendre compréhensible par un contrôleur commercial déjà présent sur les MOCN. Ce premier type de contrôleur a été largement développé dans le début de l'utilisation du STEP-NC, car il permettait de tester la viabilité du projet STEP-NC en expérimentant sur des contrôleurs de MOCN déjà existant. En revanche, ce type de contrôleur limite grandement le potentiel de STEP-NC puisqu'il utilise toujours le Code-G.
- Type 2, contrôleur interprété : Dans ce type de contrôleur, **les axes et commandes de la machine sont directement contrôlés par les informations présentes dans le fichier STEP-NC** et via les caractéristiques de la machine. Le Code-G n'est alors plus utilisé. Des données de capteurs externes peuvent être utilisées pour modifier le comportement de la machine. On peut citer comme exemple l'utilisation d'un palpeur qui permettrait la modification du parcours d'outils après mesure de la pièce pour apporter un correctif.
- Type 3, contrôleur intelligent : Ce dernier type est l'objectif que la communauté souhaite atteindre en développant le standard STEP-NC. **Il possède des fonctions intelligentes et autonomes**, le contrôleur est ici capable d'estimer en temps réel via des données acquises par la machine, mais aussi des données emmagasinées par des productions précédentes, les paramètres optimaux de la machine et le parcours d'outils le plus adaptés. Il est aussi capable de gérer les erreurs lors de la production, par exemple lors de la casse d'un outil. De plus, il intègre des outils de collaboration qui permet le partage des informations acquises lors de la production à d'autres machines partout dans le monde.

La Figure 2.7 résume les trois types de contrôleurs et leurs caractéristiques.

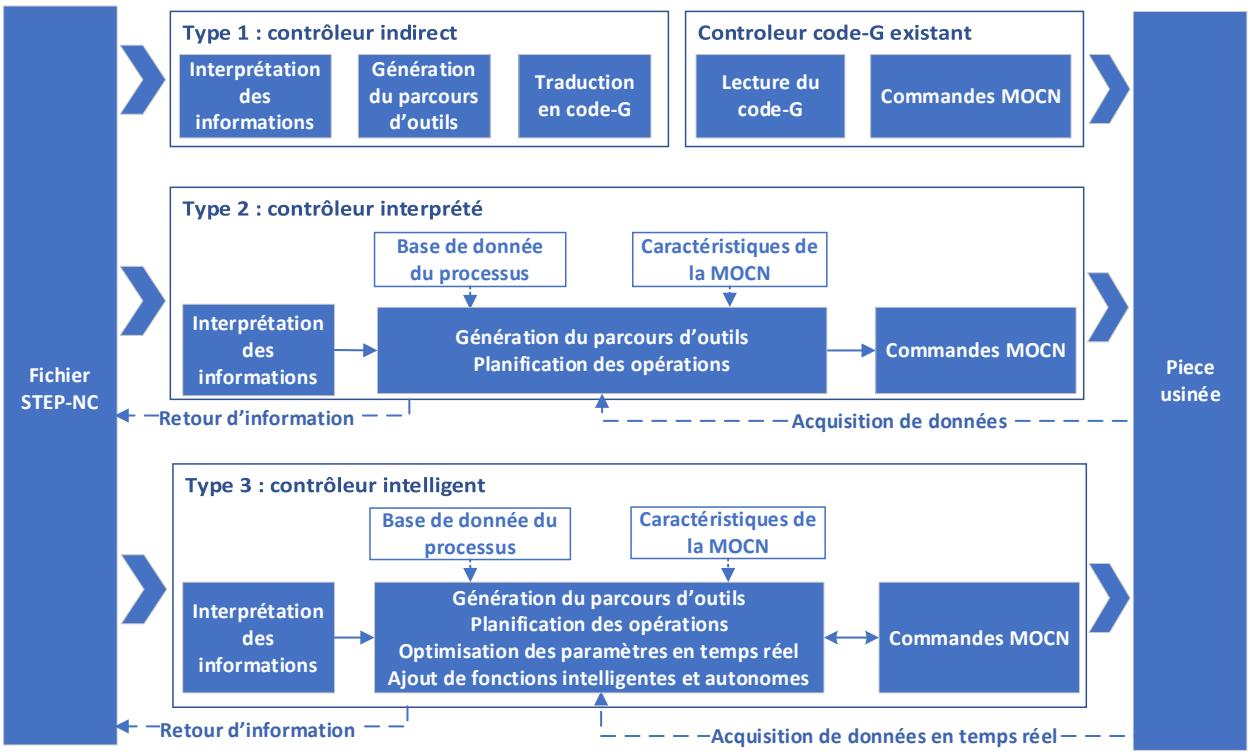


Figure 2.7 Niveaux d'implémentation d'un contrôleur STEP-NC, adapté de Rauch, Matthieu et al. (2012) et de (Suh, S. H. et al., 2003)

Les sous-sections suivantes proposent l'analyse des différents travaux recensés au 2.2.2 pour déterminer les caractéristiques de chacun des projets. Pour effectuer cette analyse, nous ferons l'étude pour chacun des types en mettant en exergue l'architecture utilisée, les choix technologiques effectués, les points forts et les limites de chaque projet.

2.2.3 Contrôleur indirect (type 1)

Les contrôleurs indirects continuent d'utiliser du Code-G afin de pouvoir exécuter la production sur un contrôleur de MOCN commercial, mais ne permettent pas d'exploiter correctement les bénéfices de STEP-NC. Ils agissent en tant que traducteurs pour traduire le langage STEP-NC en Code-G. Ce type de contrôleur peut être utile pour tester des fonctions telles que la traduction des données géométriques en fonction machine (feature recognition) ou la planification des processus de fabrication, fonctions qui n'ont pas forcément besoin de retours d'informations. Les principaux projets de contrôleur indirect ont été listés par Cha et al. (2016), Latif et al. (2021), nous allons présenter les projets les plus pertinents :

- European Strategic Programme on Research in Information Technology (ESPRIT)

  Le projet ESPRIT est composé de la collaboration entre plusieurs laboratoire européen et des industriels tel que Siemens, qui ont permis de tester et de développer les premières bases de STEP-NC dès 1999 en utilisant un contrôleur Siemens Sinumerik 840D qui permettait la traduction du langage STEP-NC en Code-G et ainsi l'exécution et la vérification de la viabilité de STEP-NC (Latif et al., 2021; Mueller & Hyu, 2001).

- STEP-NC Platform for Advanced and Intelligent Manufacturing (SPAIM)

  L'objectif de ce projet est d'utiliser le STEP-NC pour créer un contrôleur qui permet la fabrication d'une pièce sur plusieurs machines différentes à partir d'un seul et même fichier STEP-NC. De plus, des outils d'optimisation et de simulation ont aussi été développés.

  Dans le projet SPAIM, un fichier STEP-NC est utilisé tout au long de la chaine numérique pour transporter les informations jusqu'à la machine et dans toutes les étapes de FAO. C'est uniquement une fois que le fichier est importé dans la machine qu'il est traduit en Code-G spécifique à la machine utilisée. L'intelligence du post-processeur est ainsi déplacée directement dans la machine, ce qui permet une simplification du processus de FAO puisqu'il n'est pas spécifique à chaque machine et une simplification lorsqu'il y a des modifications à apporter à la pièce (Rauch, Matthieu et al., 2012). Le processus est résumé dans la Figure 2.8.

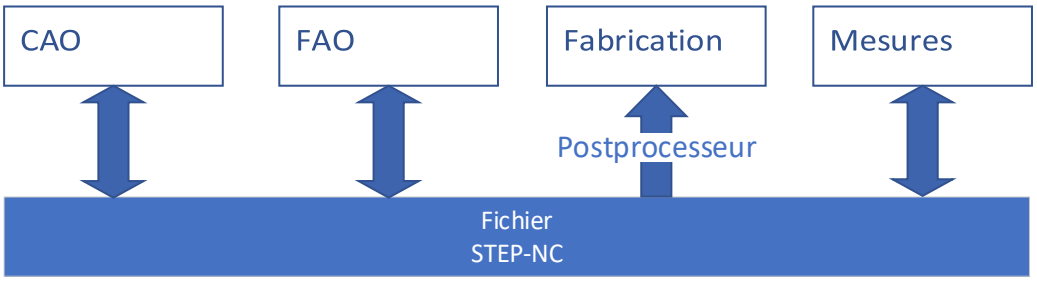


Figure 2.8 Chaine numérique en utilisant le STEP-NC adapté de (Rauch, M. & Hascoet, 2012)

Le projet propose aussi des méthodes d'optimisation, en utilisant des données acquises pendant la production de pièces précédentes pour optimiser les futurs parcours d'outils et les paramètres d'usinage.

Ce projet permet de mettre en œuvre les améliorations qui sont possibles grâce au STEP-NC. En revanche, il utilise encore un contrôleur Code-G qui reste un problème pour implanter d'autres fonctionnalités et utiliser toutes les possibilités du STEP-NC, notamment pour faire de la correction en temps réel. Une fois le parcours d'outils converti en Code-G il n'est plus possible de le modifier en cours d'exécution. L'équipe de chercheurs indique elle-même que le fait d'utiliser un contrôleur fermé est le plus gros problème pour une intégration totale du STEP-NC : « A major step would lead to solve the remaining problems of transparency of current CNC controllers that is still a problem for their total integration in the system » (Laguionie et al., 2011).

- Un troisième projet intéressant traite de la modernisation d'un tour par Xu (2006) afin d'ajouter des fonctionnalités de STEP-NC. Le projet consiste à remplacer le contrôleur Code-G de la machine par un contrôleur programmable 6K de la marque Compumotor. Ce contrôleur permet d'être programmé grâce à un langage propriétaire et permet une interopérabilité avec d'autres logiciels, de CAO par exemple. Un interpréteur est développé permettant la traduction d'un fichier STEP-NC en langage propriétaire 6K qui va pouvoir être lu par le contrôleur et exécuter l'usinage de la pièce. Bien que la machine n'utilise plus de Code-G, elle conserve les mêmes désavantages du fait d'utiliser une traduction dans un langage spécifique avant d'être exécutée et ainsi limite les possibilités de modification en temps réel du parcours d'outils ou des paramètres d'usinage. Le développement de nouveaux contrôleurs, qui remplacent entièrement celui des MOCN est indispensable pour développer le standard et profiter de toutes les possibilités qu'offre STEP-NC. « Admittedly, 6K has no further advantage over Code-G in its role as a control language between. It is nonetheless more open and easier to program and interface with other applications. » (Xu, 2006)

Nous avons pu observer les améliorations que peut apporter l'utilisation d'un contrôleur indirect qui utilise le STEP-NC, mais la traduction en langage spécifique en fin de chaine, ne permet pas l'exploitation de STEP-NC. Il est primordial aujourd'hui de développer des contrôleurs de niveau supérieur qui s'affranchissent de l'utilisation du Code-G ou d'un autre langage spécifique à la machine. La prochaine sous-section s'intéresse, pour ces raisons au contrôleurs de type 2 qui permettent d'exploiter le potentiel de STEP-NC.

2.2.4 Contrôleur interprété (type 2) et intelligent (type 3)

Ce type de contrôleur est au cœur des recherches puisque c'est ce qui permet d'exploiter toute la puissance et les avantages du STEP-NC. Nous allons maintenant passer en revue les projets de contrôleurs STEP-NC interprétés, en analysant leurs technologies, leurs avantages et leurs faiblesses.

2.2.4.1 School of Mechanical and Industrial Engineering, POSTECH, Corée du Sud (Korea STEP-NC)

C'est l'un des premiers prototypes de contrôleur interprété qui a été développé par l'université de POSTECH en Corée pour être développé sur une fraiseuse à commande numérique. Ils ont tout d'abord proposé une architecture fonctionnelle après avoir réalisé une analyse du besoin, ce qui a fait ressortir les besoins du contrôleur (Suh, S. H. & Cheon, 2002):

- Autonomie : le logiciel doit pouvoir fonctionner de manière autonome sans que l'opérateur ait besoin d'intervenir. Il faut minimiser les décisions et interactions humaines si le besoin d'un opérateur est obligatoire
- Gestion des erreurs : le contrôleur doit pouvoir gérer les erreurs et problèmes pendant la phase d'usinage.
- Qualité : pour limiter les erreurs géométriques, le contrôleur doit pouvoir gérer la mesure sur la machine (On Machine Mesurement OMM)
- Adaptatif : Les paramètres de coupe doivent pouvoir être adaptés pendant l'usinage
- Apprentissage : le processus doit acquérir des données et les utiliser pour améliorer une base de données et améliorer les futures productions.

- Architecture :

  Le projet est composé de plusieurs modules (Suh, S. H. et al., 2002) :

  - Shop floor programming / tool path generation : Ce module permet la modification et la programmation des pièces et des paramètres machine directement depuis le contrôleur de la machine dans l'atelier. Le modèle génère ensuite le parcours d'outils.

- Un module de contrôle qui gère plusieurs fonctions d'exécution du programme, notamment l'exécution du parcours d'outils qui active le déplacement des axes et le contrôle des commandes. Le module comprend une partie communication qui permet de réintroduire des informations d'usinage dans le fichier STEP-NC qui pourra donc être exploité dans la partie CAO/FAO pour des modifications.
- Une base de données commune à tous les modules, qui stocke toutes les données liées à la mesure pendant et après la fabrication afin de les utiliser pour améliorer le processus d'usinage.

Cette architecture a ensuite été mise en œuvre et validée dans un prototype (Suh, S. H. et al., 2002) puis brevetée en 2005 sous le nom « Intelligent STEP-NC controller » (Suk-Hwan Suh, 2005)

- Choix technologique :

Le module SFP/TPG a été développé en C++ en utilisant un outil de développement, développé par l'entreprise STEP Tools. Le module de contrôle est lui aussi développé en C++. Le programme est exécuté sur deux PC distincts, l'un sous Windows NT permet d'effectuer toutes les actions qui n'ont pas besoin d'être en temps réel (Modification de la pièce, génération du parcours, d'outils, visualisation du parcours d'outils) et l'autre gère la partie temps réel en envoyant notamment les signaux de déplacement aux moteurs via une carte d'entrée/sortie branchée sur la MOCN. Ce deuxième PC utilise une version de Windows NT modifiée pour être compatible avec le temps réel (real time operating system ou RTOS) (Suh, S.-H. et al., 2002; Suh, S. H. et al., 2003).

- Points forts :

Ce projet est l'un des premiers et pourtant l'un des plus complets avec toute une architecture qui intègre une bonne partie de la chaine numérique de fabrication, depuis la CAO jusqu'à la CNC. La structure du projet cherche à intégrer tous les aspects que permet le STEP-NC : planification automatique du procédé, simulation, visualisons, modification depuis l'atelier, exécution et modification en temps réel des paramètres d'usinage. La structure du projet et l'analyse de besoin qui ont été détaillées sont d'ailleurs la base de nombreux travaux qui ont suivi.

- Limites :

  La principale faiblesse provient de la complexité d'utiliser deux PC, puisqu'à l'époque du développement du projet, un seul n'aurait pas été suffisamment puissant. Cela représente un encombrement important, un prix élevé et n'est pas viable pour une utilisation commerciale.

2.2.4.2 School of Mechanical and Industrial Engineering, POSTECH, Corée du Sud (TurnSTEP)

Ce projet est une adaptation et évolution de Korea STEP-NC qui a été poursuivie dans le même laboratoire de POSTECH par Choi et al. (2006) et par Suh et al. (2006) afin d'adapter le projet pour une utilisation sur un tour à commande numérique.

- Architecture :

  L'architecture est très proche de celle de Korea STEP-NC.

- Points forts :

  L'une des nouveautés est la possibilité pour le contrôleur de modifier des paramètres d'usinages entre deux phases d'usinage, par l'ajout d'un capteur d'effort par exemple, qui renvoie les données aux contrôleurs.

- Limites :

  Bien qu'étudiée en théorie, la modification des paramètres d'usinages entre phases d'usinage n'est pas testée sur la machine. On retrouve les mêmes faiblesses que pour le projet de Korea STEP-NC

2.2.4.3 School of Mechanical and Aerospace Engineering, Seoul National University, Corée du Sud

Ce projet de Lee, W. et al. (2006). est orienté vers l'utilisation du fichier STEP-NC en langage XML (ISO 14649 part 28) et non en EXPRESS (ISO 14649 part 21) comme la majorité des autres projets.

- Technologie

Un traducteur de fichier part 21 vers part 28 a été développé. Un contrôleur capable d'interpréter le fichier, de générer le parcours d'outil et d'exécuter les fonctions sur la MOCN a ensuite été développé en C++ et exécuté sur un unique PC sous Windows avec l'aide d'une carte de contrôleur moteur. Le système est basé sur une architecture entièrement ouverte et donc adaptée à la modification et à l'ajout de modules.

- Points forts :

  L'avantage du 'XML est sa comptabilité pour être partagé, en effet sa structure en arborescence permet le partage d'une partie d'un fichier STEP-NC, très facilement. Ce qui n'est pas possible en EXPRESS puisque chaque ligne fait référence à plusieurs autres.

- Limites :

  La recherche s'est principalement axée sur l'utilisation du XML et ne prend pas en charge de nouveaux modules liés au développement du STEP-NC comme un module de visualisation, ou de simulation (Lee, Wonseok & Bang, 2010; Lee, W. et al., 2006). Il y a toujours l'utilisation d'un matériel complexe et spécifique. Le fichier XML sera aussi plus lourd qu'un même fichier en EXPRESS.

2.2.4.4 Dipartimento di Informatica e Sistemistica Università degli Studi di Napoli Federico II, Italy

- Technologie :

  La particularité de ce projet développé par Calabrese et Celentano (2007) est l'utilisation d'un microcontrôleur comme base matérielle du contrôleur au lieu d'utiliser un PC.

- Points forts :

  Le principal avantage est le prix du microcontrôleur qui est très peu cher comparé au PC très puissant utilisé précédemment dans la littérature. De plus, son encombrement est réduit ce qui permet de l'intégrer facilement dans une MOCN.

- Limites :

  Le microcontrôleur est complexe à programmer ce qui limite le développement de nouveau module. Il est aussi plus compliqué d'ajouter des périphériques pour faciliter l'utilisation

du contrôleur dans l'atelier (écran, clavier, souris). Il faut alors passer par un PC branché sur le même réseau pour accéder au microcontrôleur.

2.2.4.5 School of Mechatronics Engineering, Harbin Institute of Technology Ha, Chine (HitCNC et CS-STP)

- Technologie :

  Le développement de Po et al. (2014) se concentre sur la création d'un contrôleur à architecture ouverte dans l'optique de l'intégrer à un prototype de contrôleur permettant l'usinage en boucle fermée « Closed Loop Manufacturing (CLM) » (Tao et al., 2006) .

- Architecture :

  Le prototype est basé sur PC avec une carte de contrôle et une carte d'acquisition.

- Points forts :

  L'ajout d'un module CLM permet d'intégrer la prise de mesure aux données d'entrées pour les prochaines pièces qui vont être usinées afin d'optimiser la production, que ce soit en qualité ou en rapidité. Le développement est donc orienté vers l'acquisition de donnée et la réutilisation des données pour la prise de décision. Le projet sera ensuite amélioré en 2016 en utilisant notamment une approche par mémoire tampon afin de ne pas générer le parcours d'outils en entier avant de lancer l'usinage, mais au fur et à mesure de l'avancement de l'usinage. (Hu, Fu, et al., 2016)

- Limites :

  Il y a toujours l'utilisation d'un PC et des cartes de contrôles pour permettre la communication avec la MOCN qui rendent le contrôleur complexe et coûteux.

2.2.4.6 Faculty of Engineering, DRB-HICOM University of Automotive Malaysia, Malaisie

Le projet de l'université Tun Hussein Onn Malaysia (UTHM) est l'un des projets les plus aboutis actuellement. Sa conception a commencé en 2013 par Elias et al. (Elias et al., 2013) avec l'élaboration d'un modèle de donnée.

- Technologie :

Le développement utilise la plateforme LabVIEW. L'avantage de cette plateforme est la compatibilité avec plusieurs autres langages de programmation et l'intégration avec d'autres logiciels, comme des logiciels de CAO par exemple. En effet, ils ont utilisé plusieurs autres logiciels pendant le développement comme Mastercam un logiciel de FAO, ST-Developer et d'autres modules développés en Java. L'objectif du projet est de développer une plateforme intégrant toute la chaine numérique, depuis la CAO jusqu'à la fabrication et la gestion de la base de données. Le prototype, développé ensuite dans (Latif & Yusof, 2015; Latif et al., 2017; Latif et al., 2016; Yusof & Latif, 2015b) est capable de traduire les données STEP-NC en commande machine, de simuler en 3D le parcours d'outils généré. Il est exécuté sur un PC muni d'une carte de contrôle et d'une carte d'acquisition afin de contrôler une fraiseuse 3 axes.

- Points forts :

  La plateforme développée intègre toute la chaine numérique de la CAO à la fabrication. De nombreuses fonctionnalités sont ajoutées, comme le monitoring vidéo en direct, la possibilité de réaliser du CLM et du contrôle en temps réel.

- Limites :

  Bien que le projet soit très abouti au niveau des fonctionnalités, il est difficile à implanter dans une usine, car il utilise des cartes de contrôles développées spécifiquement, ainsi qu'un PC qui doit être puissant pour réaliser toutes les tâches. Ils seraient donc très coûteux et complexes à mettre en place pour une entreprise. La plateforme LabVIEW utilisée limite la reproductibilité du projet et sa fermeture ne répond pas aux exigences des contrôleurs OAC.

2.2.4.7 College of Mechanical and Electrical Engineering Harbin Engineering University, Chine Harbin,China

Le projet développé par Li et al. (Li & Liang, 2011; Li et al., 2009) puis par Liang (Liang & Li, 2013) se concentre sur l'intégration de l'usinage de forme libre en STEP-NC. Les formes libres appelées Non-Uniform Rational Basis Splines (NURBS) sont complexes à usiner et le Code-G est très peu adapté pour réaliser ce genre de surface.

- Technologie :

Un prototype de contrôleur est développé, capable d'interpréter et d'exécuter des NURBS. Le contrôleur est codé en C++ et basé sur PC sous Windows avec un module temps réel développé par VenturCOM et permet de contrôler une fraiseuse à cinq axes.

- Points forts :

  L'utilisation du STEP-NC permet de faciliter l'usinage des NURBS puisqu'il est possible d'intégrer beaucoup plus d'information comme la représentation mathématique sous forme d'équation paramétrique des surfaces dans l'espace. Ces équations peuvent ensuite être traitées par le contrôleur pour exécuter le parcours d'outils.

- Limites :

  Comme pour les précédents projets, l'architecture matérielle sur PC est complexe et onéreuse.

2.2.4.8 Autres projets de type 2 aux fonctionnalités pertinentes

On peut ensuite citer plusieurs projets qui ont ajouté au fur et à mesure des années des fonctionnalités et des choix technologiques différents pour réaliser un contrôleur STEP-NC. Lan et al. (2008) reprennent une architecture assez similaire aux projets vus précédemment. La valeur ajoutée du projet est d'utiliser un système d'intelligence artificielle par agent-multiple pour apporter de l'autonomie au contrôleur. Ils sont utilisés pour la prise de décision dans plusieurs cas de figure, on peut citer par exemple la planification du procédé, le contrôleur est codé en C++ en utilisant notamment ST-Developer et des librairies développées par STEP Tools. Le projet est uniquement testé en simulation. Wang, K. et al. (2012) ajoutent la possibilité d'utiliser le réseau Ethernet pour communiquer entre le PC et la MOCN afin d'assurer une meilleure interopérabilité. Mohamed et al. (2013) utilisent une approche de programmation par bloc de fonction (FBDK) pour coder le contrôleur. Sang et Xu (2013) proposent un contrôleur qui se concentre sur la récupération d'un parcours d'outils après la casse d'un outil en utilisant un parcours d'outils générer en temps réel. C'est uniquement un modèle théorique qui est proposé et n'est pas mis en application dans un prototype. Enfin, Zhang et al. (2018) proposent un contrôleur codé en Java et basé sur une architecture d'un PC industriel (IPC) afin de rendre le contrôleur plus fiable. Ils utilisent ensuite un deuxième PC esclave pour

gérer les commandes de la machine. La solution est assez complexe et coûteuse à mettre en place.

Les contrôleurs de type 2 permettent d'exploiter réellement les avantages STEP-NC en opposition à ceux de type 1. En revanche, il manque encore la dimension d'intelligence et de collaboration qui doit être implémentée pour réaliser un contrôleur de type 3 que nous allons étudier dans la sous-section suivante.

2.2.4.9 Contrôleur intelligent (type 3)

Il n'existe pour l'instant aucun contrôleur qui peut être considéré comme un contrôleur intelligent. En effet, il faut déjà pouvoir concevoir un contrôleur interprété (type 2) avant de pouvoir développer un contrôleur qui intègre des fonctionnalités dites intelligentes et une notion de collaboration et comme nous l'avons vu précédemment il existe toujours un manque de contrôleur de type 2.

En revanche, il existe déjà certaines fonctionnalités qui apportent de l'intelligence aux contrôleurs comme nous avons déjà vu dans la partie précédente. On peut citer par exemple l'usinage en boucle fermée (Brecher et al., 2010), la correction automatique en cours d'usinage des paramètres machines (Zhao et al., 2008) ou encore la capitalisation de connaissance depuis la MOCN pour aider à la programmation de la fabrication (Danjou et al., 2016).

2.3 Revue critique

Dans la partie précédente, nous avons pu examiner les principaux projets de contrôleur STEP-NC qui ont été réalisés au cours des 20 dernières années. Le Tableau 2.4 est une synthèse des principaux prototypes de contrôleur interprété. On peut remarquer que l'intérêt est toujours présent dans le développement du standard STEP-NC, mais qu'il reste encore beaucoup à développer pour rendre le projet viable pour un usage commercial.

Tableau 2.4 Synthèse des principaux prototypes de contrôleur STEP-NC interprété

Référence	Langage principal utilisé	Plateforme matérielle	Norme ISO utilisée
(Suh, S. H. et al., 2002) (Suh, S. H. & Cheon, 2002) (Suh, S.-H. et al., 2002) (Suh, S. H. et al., 2003) (Suk-Hwan Suh, 2005)	C++	2 PC	14649
(Suh, S.-H. et al., 2006) (Choi et al., 2006)	?	?	14649
(Lee, W. et al., 2006) (Lee, Wonseok & Bang, 2010)	C++	PC	14649
(Calabrese & Celentano, 2007)	C	Microcontrôleur RCM 3700	14649
(Po et al., 2014) (Tao et al., 2006)	C++	PC	14649
(Elias et al., 2013) (Yusof & Latif, 2015a) (Yusof & Latif, 2015b) (Latif et al., 2016) (Latif et al., 2017)	LabVIEW	PC	14649
(Li et al., 2009) (Li & Liang, 2011) (Liang & Li, 2013)	C++	PC	14649
(Lan et al., 2008)	C++	?	10303 AP238
(Wang, G. X. et al., 2012)	Java	PC	10303 AP238
(Mohamed et al., 2013)	Java	PC	10303 AP238
(Sang & Xu, 2013)	C#	PC	14649
(Hu, Fu, et al., 2016) (Hu, Han, et al., 2016)	C++	PC	14649
(Zhang et al., 2018)	Java	PC	10303 AP238

Cette synthèse nous permet de réaliser que la majorité des projets utilisent un PC comme plateforme matérielle du contrôleur. En effet, c'est le matériel qui est recommandé lorsque l'on souhaite réaliser un contrôleur OAC du fait de sa puissance et de sa versatilité. En revanche, un PC qui utilise le plus souvent un système d'exploitation Windows doit être modifié au niveau de ce dernier, pour accueillir un module temps réel pour assouvir les besoins de déterminisme que nécessite le contrôle des moteurs. Le module le plus souvent utilisé est RTX développé par IntervalZero qui permet de rendre Windows compatible avec la commande en temps réel. De plus, un PC ne possède dans la plupart des cas, pas de ports d'entrée/sortie qui permettent la communication avec la MOCN, il faut alors ajouter une carte de commande et une carte d'acquisitions pour réaliser ces tâches. Ces cartes sont souvent développées directement par le laboratoire comme dans le cas du projet de Latif et al. (2017). La diversité des PC utilisés, des

systèmes d'exploitation et des cartes de commande est un frein à la reproductibilité des expérimentations. En effet, un logiciel développé sur une plateforme ne sera peut-être pas compatible avec un contrôleur développé sur un autre PC avec une autre carte de commande.

L'utilisation d'un PC et de cartes de contrôles développées spécifiquement est onéreuse en plus d'être assez complexe. On peut estimer à plus de 5000$ les contrôleurs basés sur un PC et cartes de contrôle. On a ainsi pu observer un projet utilisant un microcontrôleur par Calabrese et Celentano (2007), bien moins onéreux (environ 500$) que les prototypes présentés précédemment et permettant d'être intégré directement dans le bâti de la machine du fait de son faible encombrement. Le désavantage de celui-ci est la complexité de programmation et de connexion avec d'autres périphériques.

Les deux projets les plus aboutis sont le projet Korea STEP-NC de l'université de POSTECH et celui de l'UTHM. Le projet Korea STEP-NC est aujourd'hui obsolète d'un point de vue matériel. En effet, il devait utiliser deux PC pour avoir assez de puissance pour réaliser les diverses tâches. Le deuxième propose un contrôle abouti, mais qui utilise la plateforme LabVIEW qui est une plateforme propriétaire, bien qu'elle puisse accepter des modules écrits dans d'autres langages. Cela peut être un frein à son développement et ne respecte pas l'ouverture imposée par un contrôleur OAC. Il utilise aussi dans son architecture une carte de contrôle développé spécifiquement pour le projet afin de pouvoir connecter le PC à la MOCN ce qui limite la reproductibilité de leur contrôleur.

Au niveau de la programmation, il semble y avoir une majorité de projets utilisant le langage C++ qui a l'avantage d'être rapide, compatible avec de nombreuses plateformes et très puissant. En revanche, certains langages plus modernes comme le Java ont aussi été utilisés dernièrement pour leur facilité de codage.

2.4 Conclusion

En conclusion, la revue de littérature nous a permis de présenter les principaux projets de contrôleur STEP-NC. Les contrôleurs de type 1 sont assez nombreux et ne nécessitent plus de développement du fait qu'ils ne peuvent pas profiter totalement du standard STEP-NC. En revanche, il existe toujours un manque de contrôleur interprété (type 2) qu'il faut continuer de développer afin de tester de nouvelles architectures et de nouvelles fonctionnalités. C'est uniquement une fois que ce

type de contrôleur sera bien développé, qu'on pourrait envisager le développement de contrôleur intelligent (type 3) qui permettra l'exploitation totale des avantages du STEP-NC.

En effet, le manque de contrôleur est toujours considéré comme le principal problème qui limite l'utilisation du STEP-NC : « The lack of STEP-NC controllers has been considered as a major hindrance to popularization of STEP-NC » (Othman et al., 2017). La difficulté (voir l'impossibilité) de se procurer un contrôleur ne permet pas de montrer et de tester les avantages de STEP-NC. Cela limite l'intérêt que pourrait porter le secteur industriel sur le projet et freine son développement.

Bien qu'il existe des projets aboutis, comme nous l'avons vu dans la partie précédente, qui mettent en avant les avantages majeurs que permet STEP-NC, ces contrôleurs possèdent une architecture complexe, manquent de flexibilité et sont onéreux. Ces défauts empêchent le développement des contrôleurs STEP-NC dans l'industrie manufacturière et privent les entreprises d'un potentiel d'innovation très important.

Il est donc approprié de vouloir continuer de développer des contrôleurs STEP-NC, en le rendant moins coûteux, moins complexe, plus flexible et avec un faible encombrement afin de faciliter son implantation dans l'industrie. De plus, aucun contrôleur STEP-NC étant disponible commercialement et aucun code de contrôleur n'étant partagé, il est nécessaire de développer d'abord un contrôleur de type 2 avant de pouvoir développer de nouvelles fonctionnalités afin de tendre vers un contrôleur de type 3. La méthodologie pour le développement de notre contrôleur sera détaillée dans le chapitre suivant.

CHAPITRE 3 MÉTHODOLOGIE

Ce chapitre a pour but de présenter les objectifs de notre projet de recherche et de préciser ensuite la méthodologie qui va être employée pour y répondre.

3.1 Objectif de recherche

La revue de littérature a permis de mettre en avant la nécessité de développer de nouveaux contrôleurs STEP-NC, en axant la recherche sur la réduction du coût, la simplification du matériel utilisé, la flexibilité et la facilité de reproduire le prototype. Une attention particulière a été mise sur l'ouverture du projet afin de suivre la philosophie des contrôleurs à architecture ouverte. Pour cela, il est nécessaire que la partie logicielle qui a été développée soit évolutive et que l'architecture matérielle soit facilement adaptable et elle aussi évolutive. Cela permet d'avoir une base de travail qui pourra évoluer et accueillir de nouvelles fonctionnalités par la suite.

La problématique qui résulte de notre analyse est la suivante : comment développer un contrôleur STEP-NC interprété (type 2), peu coûteux, flexible, compact et facilement reproductible, permettant la lecture, l'interprétation puis l'exécution sur un tour à commande numérique d'un fichier STEP-NC ?

On détaille les sous-objectifs (S-O) suivant :

- S-O 1 : Recenser et spécifier les requis pour le contrôleur afin qu'il réponde aux exigences fixées : peu coûteux, adaptable, évolutif et compact. Nous avons répondu à ce S-O dans le chapitre 2.
- S-O 2 : Développer le contrôleur d'un point de vue du logiciel et d'un point de vue de l'architecture matériel. Nous répondrons à ce S-O dans le chapitre 4.
- S-O 3 : Valider le contrôleur sur un cas réel. Nous répondrons à ce S-O dans le chapitre 5.

3.2 Méthodologie de recherche

Cette section permet de présenter la méthodologie de recherche qui sera utilisée pour répondre à nos objectifs de recherche.

3.2.1 La revue de littérature

La revue de littérature, réalisée dans le chapitre 2 et synthétisée dans le tableau 2.4, nous a permis de faire un état de l'art des prototypes de contrôleur STEP-NC existants, d'étudier les choix technologiques utilisés, les aboutissements et les limitations de chacun des projets. Cette revue de littérature nous a permis de justifier la création d'un nouveau contrôleur et de guider les choix technologiques que nous avons faits. Une étude approfondie du standard STEP-NC, du standard actuel Code-G et des projets de contrôleur à architecture ouverte nous a permis de bien comprendre les besoins et les attentes actuels en matière de contrôleur pour MOCN. Cela nous a permis de répondre au S-O 1.

3.2.2 Le développement

Le développement est divisé en quatre sous-étapes qui correspondent à la méthode de développement proposée par Suh, S. H. et Cheon (2002) et utilisée par la suite par les autres principaux projets. Ils sont développés dans le chapitre 4 :

- Choix technologiques : Une analyse du matériel capable de répondre aux exigences demandées a été effectuée et un choix de matériel a été fait. Le langage de programmation approprié et adapté aux exigences a été alors choisi.
- Interprétation : La première partie de développement se concentre ensuite sur la partie logicielle, qui devra être capable d'interpréter un fichier STEP-NC. Une méthode de stockage des données sous forme d'instanciation d'objet a été développée.
- Génération du parcours d'outils : La deuxième étape de développement consiste à utiliser ses données interprétées et à les exploiter afin de générer le parcours d'outils et toutes les actions nécessaires à l'usinage.
- Contrôleur bas niveau : Une dernière étape de développement permet d'utiliser les données générées par la partie précédente pour faire fonctionner les différents actionneurs de la MOCN. Cette étape permet de répondre au premier objectif de recherche. On utilise la méthode développée dans le livre de Suh, S.-H. et al. (2008). Ces étapes de développement logiciel permettent de répondre au S-O 2.

3.2.3 Validation expérimentale

L'expérimentation a pour but de valider les étapes de développement précédemment effectuées. Elle sera effectuée dans le chapitre 5.

Un tour à commande numérique EMCO PC TURN 55, présenté Figure 3.1, a été modifié afin de remplacer son contrôleur Code-G par le nouveau contrôleur STEP-NC précédemment développé.

La MOCN utilisée est un tour à commande numérique possédant deux axes de travail, dont les caractéristiques sont détaillées dans le chapitre 5.

Figure 3.1 EMCO PC TURN 55 utilisé pour les essais

Afin d'effectuer un essai qui permet la validation de notre prototype, on utilise l'exemple du code STEP-NC présenté dans l'Annexe D de la norme (2003c) dont le schéma est représenté Figure 3.2. Ce fichier STEP-NC standard assure la reproductibilité des tests.

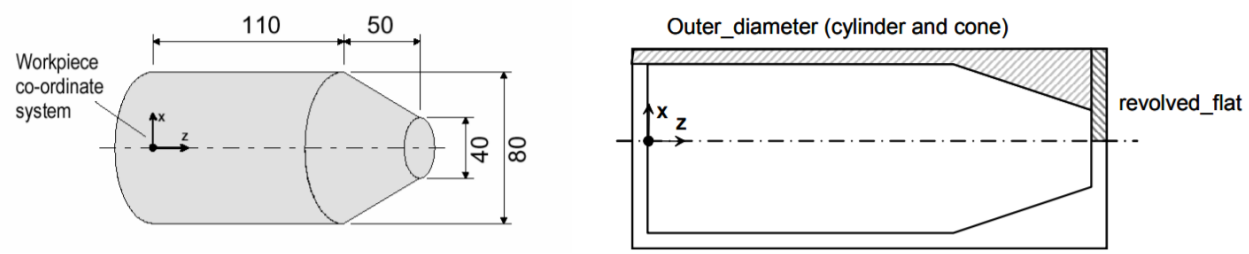


Figure 3.2 Schéma de la pièce Annexe D de la norme ISO14649 Part 12 (ISO, 2003c)

La pièce illustrée Figure 3.2 est constituée d'une face, d'un cône et d'un cylindre dont les dimensions sont données ci-dessus. Elle est constituée de quatre opérations et de trois éléments à usiner. Le détail des phases d'usinage de cette pièce est présenté dans le chapitre 5.

Une première étape d'expérimentation a pour but de tester le matériel utilisé. Pour cela, un déplacement manuel des axes de la MOCN a été effectué en utilisant les commandes générées par le contrôleur. On peut ainsi conclure sur le fonctionnement et la performance du matériel utilisé.

Une deuxième étape permet de tester, le fonctionnement automatique de la MOCN via la lecture et l'interprétation d'un fichier STEP-NC par le contrôleur. C'est alors le code du contrôleur qui peut être testé et ses performances évaluées.

Cette étape permet de valider le S-O 3.

3.3 Conclusion

Ce troisième chapitre nous a permis de définir les objectifs de notre recherche et de développer la méthodologie du développement qui a été utilisée pour répondre aux objectifs définis. La méthodologie est résumée Figure 3.3. La méthodologie de l'expérimentation a aussi été présentée et permet la validation du prototype. Le prochain chapitre présente le développement du contrôleur qui répond aux exigences que nous avons définies.

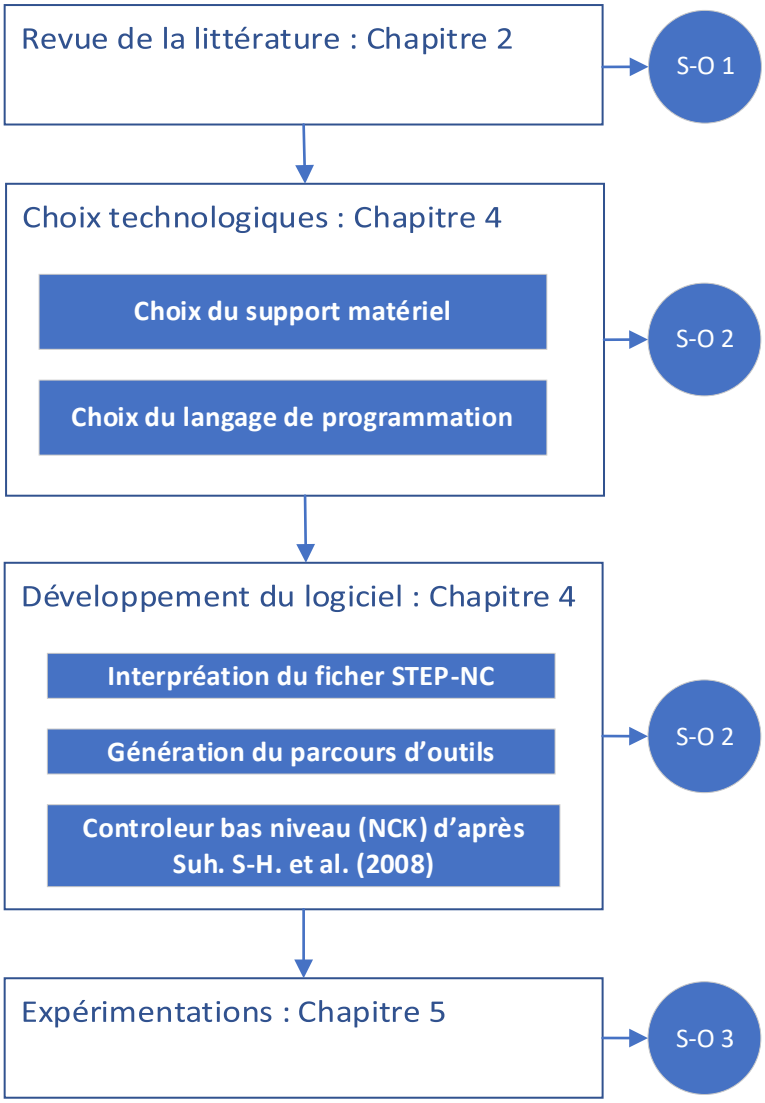


Figure 3.3 Méthodologie de recherche

CHAPITRE 4 DÉVELOPPEMENT

Ce chapitre présente le processus de développement d'un contrôleur STEP-NC de type 2 en détaillant tout d'abord le choix d'architecture matérielle et logicielle puis le développement du logiciel du contrôleur.

4.1 Introduction

Notre objectif est de développer un contrôleur STEP-NC de type 2, c'est-à-dire un contrôleur capable de lire un fichier STEP-NC, d'interpréter les informations présentes dans le fichier et de les utiliser pour réaliser le déplacement des axes d'une MOCN et l'activation des fonctionnalités nécessaires à l'usinage de la pièce. Le contrôleur est donc totalement dépourvu de Code-G, et aucune information n'est perdue dans l'interprétation du fichier STEP-NC.

4.2 Choix d'architecture matérielle et logicielle

Cette section a pour but d'expliquer les choix qui ont été faits du point de vue matériel et logiciel du contrôleur. Le matériel doit être en mesure de supporter l'exécution du logiciel précédemment développé et de communiquer avec les autres composants de la machine (contrôleurs de moteur, capteurs, actionneurs…). La partie matérielle des contrôleurs Code-G actuels, est composée d'une électronique propriétaire qu'il n'est pas possible de modifier et qui est totalement fermée. De plus, le contrôleur ne nécessite pas de puissance de calcul importante puisqu'il ne fait qu'exécuter le parcours d'outils inscrit dans le fichier Code-G. Le contrôleur STEP-NC nécessite une plus grande puissance de calcul puisqu'il doit interpréter les données pour générer le parcours d'outils, permettre une visualisation de l'usinage et il peut y avoir d'autres modules qui seront ajoutés par la suite. La puissance qui est normalement requise par le logiciel de FAO pour générer le parcours d'outils est maintenant nécessaire directement dans le contrôleur de la MOCN.

4.2.1 Choix de la plateforme matériel

La revue de littérature nous a permis de mettre en avant les différentes approches qui ont été utilisées dans le développement matériel des contrôleurs STEP-NC. Nous avions vu qu'une majorité de projets choisissait d'utiliser des PC ou des PC industriels. Or, nous avions mis en avant les problèmes liés à son utilisation : un prix élevé, un manque de connectivité pour communiquer

avec la MOCN, ce qui nécessite l'ajout une carte de contrôle et enfin son encombrement élevé. Il en résulte une très grande variété de PC, de cartes de contrôle, de systèmes d'exploitation, ce qui limite la comptabilité et le partage des codes des contrôleurs STEP-NC qui seront créés, puisqu'il faut les adapter à toutes les configurations.

Calabrese et Celentano (2007) ont montré qu'il était possible de développer un contrôleur STEP-NC de type 2 pour un microcontrôleur intégré basé sur un microprocesseur RCM3700 à un très faible coût. En revanche, le développement logiciel d'un microcontrôleur est complexe et est donc un frein aux développements et aux partages de nouveau module. Son interface avec l'opérateur est aussi limitée puisqu'il ne présente pas de possibilité d'ajouter un écran ou d'interagir à l'aide d'un clavier et d'une souris par exemple. Il faut passer par un PC qui communiquera avec le contrôleur.

Si l'on se penche sur les contrôleurs OAC développés pour être utilisés avec le Code-G, on observe une augmentation de l'utilisation des Single Board Computer (SBC), ou ordinateur à carte unique en français, comme support matériel du contrôleur. On peut notamment citer le Raspberry Pi (RPi) qui est l'un des plus connus. Un SBC consiste en une unique carte électronique comportant tous les composants d'un PC, bien que moins puissant. Ses principaux avantages sont qu'ils sont très peu coûteux, compacts, mais surtout qu'ils possèdent des ports d'entrée sortie, appelée GPIO pour General Purpose Input/Output, ce qui permet de communiquer directement avec une MOCN sans carte de contrôle annexe.

Les SBC sont très utilisés depuis quelques années dans des projets amateurs de conceptions de MOCN, mais aussi dans des projets de modernisation de contrôleur d'ancienne machine devenu obsolète. On doit cela notamment au développement de logiciel de contrôleur comme LinuxCNC qui est un logiciel contrôleur OAC fonctionnant sous Linux permettant de contrôler une MOCN et fonctionnant sur plusieurs types de support, dont le RPi.

Grigoriev et Martinov (2016) ont démontré qu'un SBC était suffisamment puissante pour remplir les principales fonctions d'une MOCN à plusieurs axes en réalisant un contrôleur Code-G OAC.

Nous avons donc choisi d'utiliser pour notre support matériel un Raspberry Pi 4B. Il possède un processeur ARM quatre cœurs cadencés à 1,5 GHz, 8 Go de RAM, des connectiques USB et HDMI, ainsi qu'un GPIO à 40 pins. Ce SBC rempli bien les conditions demandées, il est disponible facilement pour environ 100 CA$, son système d'exploitation est Raspberry OS qui est une

distribution Linux, elle accepte le développement de programme dans plusieurs langages, notamment en C++ et en Python. Il est très facile d'ajouter de nouveaux capteurs ou actionneurs grâce à son GPIO de 40 pins. Il offre de nombreuses possibilités de connectivité grâce à son port Ethernet et sa connexion Wifi pour de futures améliorations. Ce SBC remplit donc bien les conditions demandées pour réaliser un contrôleur OAC.

4.2.2 Choix du langage de programmation

Afin de développer la partie logicielle dans la section suivante, capable d'interpréter un fichier STEP-NC et de générer des mouvements et actions machine, il est nécessaire de sélectionner un langage de programmation qui répond à plusieurs critères classés par ordre d'importance décroissante :

- Premièrement, le langage devrait être open source afin d'assurer une possibilité d'évolution et un développement facilité de nouveaux modules, cela répond aussi aux conditions d'un contrôleur OAC.
- La rapidité et les performances du langage sont aussi à prendre en compte, en effet, une partie des opérations doit être effectuée en temps réel. Un langage performant sera donc requis pour effectuer toutes les opérations.
- Le langage doit aussi être compatible avec une large gamme de machines et de systèmes d'exploitation afin de répondre aux exigences d'un contrôleur OAC. Il faut qu'il puisse être compatible avec un PC sous architecture X86 ou ARM sous Windows ou Linux afin d'assurer la facilité de développement.
- Puisque le langage STEP-NC est basé sur une logique de programmation orientée objet, il est indispensable que le langage utilisé soit compatible avec la programmation orientée objet, afin d'avoir une transcription intuitive des entités définies par la norme STEP-NC vers le langage de programmation.
- Enfin, notre revue de littérature a pu mettre en avant les langages utilisés dans les précédents projets. Utiliser un langage populaire dans le domaine permet d'avoir potentiellement des bibliothèques ou libraires déjà existantes et faciliter la collaboration dans le développement du code et assurer une compatibilité entre les modules qui pourraient être développés.

On liste les langages de programmation les plus couramment utilisés et on les compare, via les critères cités précédemment, dans le Tableau 4.1.

Tableau 4.1 Matrice de décision pour le choix du langage de programmation

	Facilité de codage	Compatibilité	Open source	Performance	Popularité dans la littérature	**Total pondéré**
Pondération	2	3	5	4	1	
LabVIEW	4	3	0	3	3	**43%**
C++	**1**	**5**	**5**	**5**	**5**	**89%**
C#	3	3	5	3	3	**73%**
Java	3	3	5	2	3	**68%**
Python	5	2	5	3	2	**73%**

1 étant la note la plus faible
5 étant la note la plus élevée

Au vu des critères de sélection, le langage C++ est le plus adapté à notre projet. Il a été utilisé dans plusieurs autres projets comme nous l'avons vu dans la revue de littérature, ce qui conforte notre choix. De plus, le C++ est un langage très performant, il est aussi compatible avec une multitude d'architectures et de systèmes d'exploitation et notamment avec le RPi que nous avons choisi dans notre cas.

4.2.3 Communication entre le contrôleur et la machine

Pour pouvoir communiquer avec le contrôleur de moteur, notre contrôleur STEP-NC doit envoyer des pulsations avec une tension de 5V. Or le GPIO du RPi fonctionne en 3V. On ajoute donc une carte éleveur de tension, 3V vers 5V (appelée level shifter), facilement trouvable dans le commerce pour un prix très faible. On pourrait ensuite directement relier le GPIO aux contrôleurs de moteurs de la MOCN, mais pour protéger la carte d'éventuelle surtension et pour faciliter le branchement, on utilise une Break Out Board (BOB) qui permet l'interface entre le RPi et les drivers des moteurs de la MOCN. Une BOB pour MOCN 5 axes avec coupleurs optiques est trouvable facilement pour moins de 20 CA$. La BOB permet aussi la connexion de la RPi avec le contrôleur de fréquence variable (VFD pour Variable Frequency Drive) qui contrôle la broche de la MOCN via un signal en modulation de largeur d'impulsion qui fait varier une tension de 0 à 10V.

La BOB ainsi que le GPIO de la Rpi sont capables aussi bien d'envoyer des signaux que d'en recevoir. Il est donc adapté à un développement futur pour brancher des capteurs ou d'autres actionneurs.

Les signaux envoyés par la Rpi aux drivers doivent être envoyés en temps réel. Le système d'exploitation utilisé sur le Rpi (Raspberry OS) n'est pas adapté pour respecter les contraintes de temps réel, il faut avoir un système d'exploitation temps réel appelé Real Time Operating System (RTOS), mais contrairement à un système Windows qui nécessite un module payant et fermé pour être transformé en système d'exploitation temps réel (RTX de Venture COM est souvent utilisé), les distributions Linux possèdent un correctif temps réel à installer, pour adapter Raspberry OS en un RTOS.

L'architecture matérielle du contrôleur est résumée dans la Figure 4.1 et le matériel utilisé est présenté Figure 4.2.

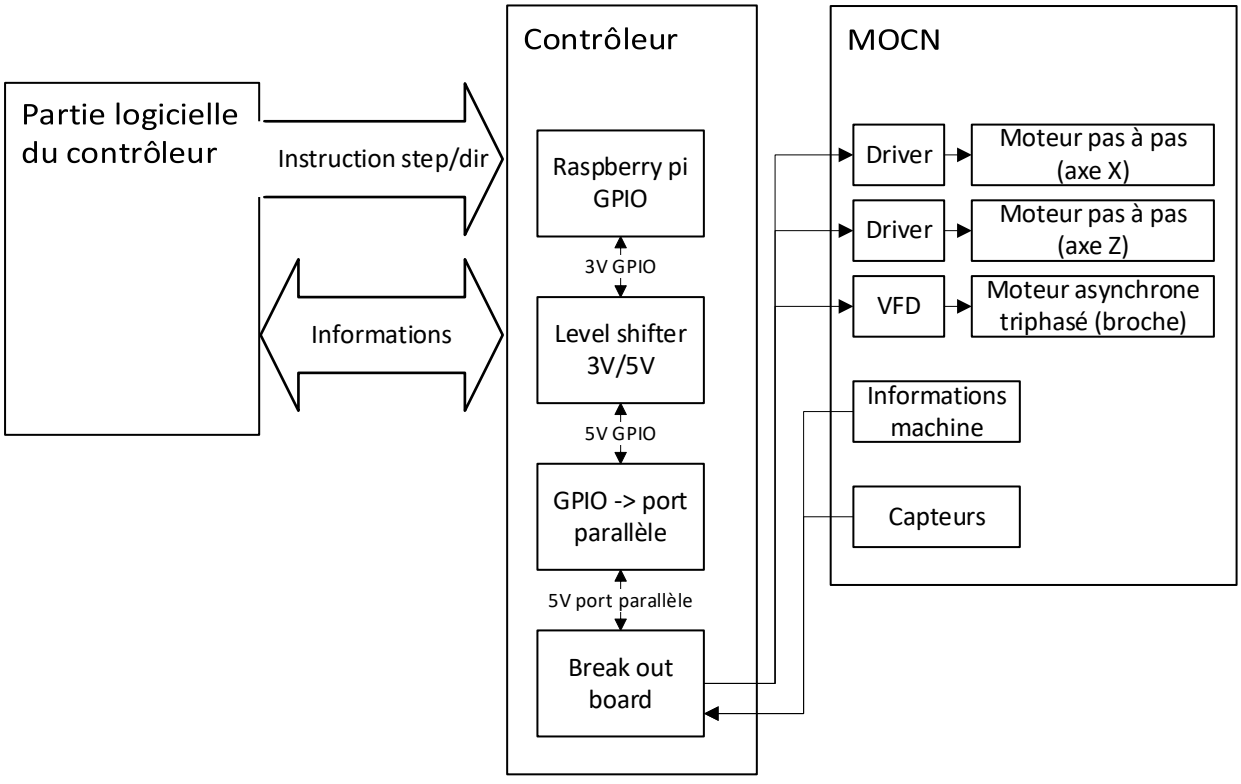


Figure 4.1 Architecture matérielle du contrôleur

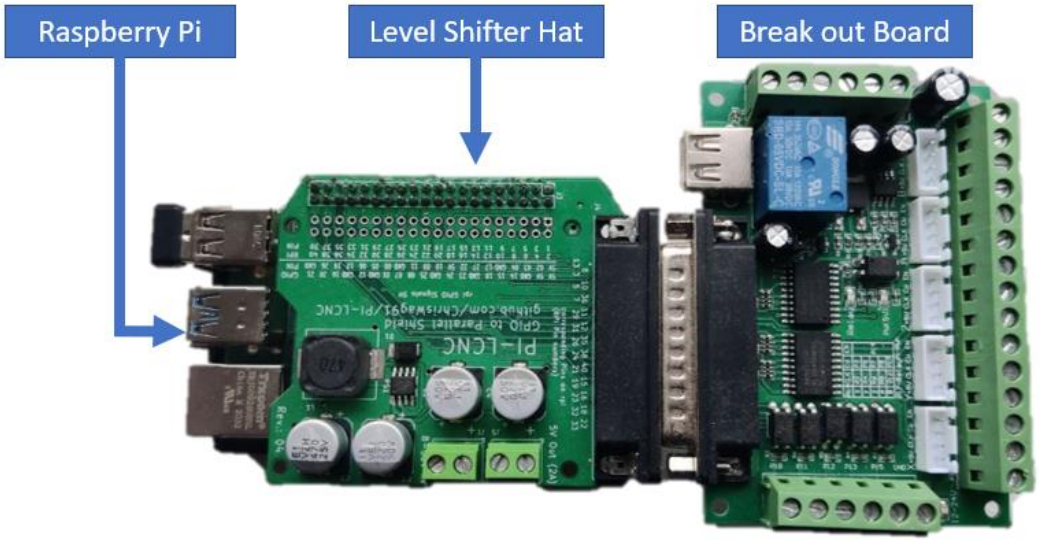


Figure 4.2 Matériel du contrôleur développé

Maintenant que l'architecture matérielle ainsi que le langage de programmation ont été choisis, nous allons présenter les étapes de développement du logiciel du contrôleur.

4.3 Développement du logiciel du contrôleur

Cette partie permet d'expliquer le fonctionnement et la structure de la partie logicielle du contrôleur. Elle est divisée en plusieurs modules qui composent le contrôleur et qui permettent, l'interprétation des informations présentes dans un fichier STEP-NC pour générer des signaux permettant de mettre en mouvement et en action la MOCN pour usiner la pièce souhaitée.

4.3.1 Présentation des informations d'un fichier STEP-NC

Il n'existe actuellement aucun logiciel de CAO/FAO, disponible commercialement, permettant de générer un fichier STEP-NC, il est donc nécessaire d'écrire le fichier manuellement, en générant toutes les entités nécessaires qui permettent de décrire le processus d'usinage de la pièce. Il existe aussi des exemples de fichier STEP-NC présent dans la norme pour chaque procédé de fabrication. Nous utiliserons le fichier de la norme pour le procédé de tournage lors de nos essais dans le chapitre 5.

Afin de bien comprendre les fonctionnalités nécessaires dans le développement de notre logiciel, il faut tout d'abord avoir connaissance des informations qui sont présentes dans un fichier STEP-NC. Pour cela, nous allons présenter dans les sous-sections suivantes des exemples des principales

entités qui composent un fichier STEP-NC afin d'identifier les données qu'elles contiennent et leur structure.

4.3.1.1 Header

```
HEADER;
FILE_DESCRIPTION(('piece exemple pour tournage : piece 01'),'');
FILE_NAME('piece01','2021-07-01T11:32:00-05:00',('Julien Bechtold'),('Polytechnique Montreal'),'','','');
ENDSEC;
```

Figure 4.3 : *Header*

La section *HEADER* est composée d'informations générales sur le programme d'usinage, comme sa description, les noms des auteurs, sa date de création et sa version par exemple.

Ces éléments bien que non indispensables à la fabrication de la pièce, permettent d'accéder à des informations qui pourront être affichées à l'utilisateur sur la MOCN ou permettre le suivie de la pièce par exemple.

4.3.1.2 Data

La section *DATA* comporte le programme d'usinage en lui-même et peut être divisée en plusieurs sous-sections qui permettent de décrire l'ensemble de la séquence d'usinage, la géométrie de la pièce, les opérations à effectuer, les outils à utiliser, etc.

4.3.1.2.1 Workpiece

On définit les pièces qui seront usinées dans le programme avec des caractéristiques liées à la pièce comme dans notre exemple Figure 4.4 son matériau. En effet, STEP-NC permet de définir une liste de pièces qui seront usinées dans un même programme.

```
/*Workpiece*/
#100=WORKPIECE('Workpiece01',#101,0.010,#103,#1065,$,());
#101=MATERIAL('DIN EN 10027-1','E 295',(#102));
#102=NUMERIC_PARAMETER('ELASTIC MODULUS',2.E11,'pa');
```

Figure 4.4 Workpiece

4.3.1.2.2 Feature

Cette sous-section permet de définir les géométries à usiner. On définit dans notre exemple Figure 4.5 :

```
/*Feature*/
#400=REVOLVED_FLAT('FACE',#100,(#500,#501),#730 ,#732 ,0.000,#733);
#401=OUTER_DIAMETER('CONE',#100,(#502,#503),#731,#733,#503,#504);
```

Figure 4.5 Features

- *REVOLVED_FLAT* qui correspond à une géométrie de révolution avec un profil linéaire définie comme dans la Figure 4.6 :.

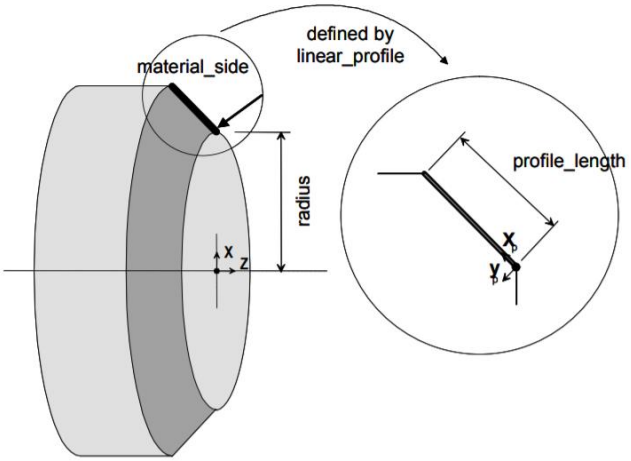


Figure 4.6 Revolved flat

- *OUTER_DIAMETER* qui correspond à la géométrie du cône ou à celui du cylindre en fonction du paramètre défini comme dans la Figure 4.7.

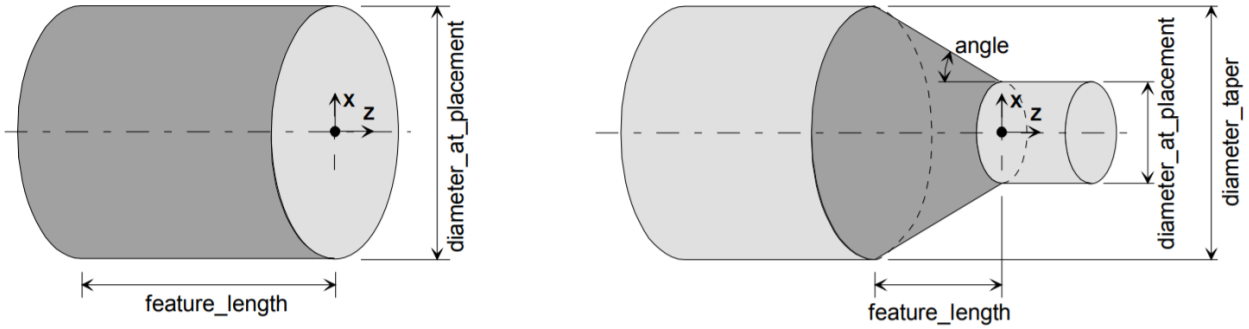


Figure 4.7 Outer diameter

4.3.1.2.3 Operation

Une fois les géométries définies, il faut définir les opérations qui vont permettre de réaliser la pièce. Notre exemple Figure 4.9 en comporte quatre, dont deux (FACING_ROUGH et FACING_FINISH) correspondent à l'usinage de la feature REVOLVED_FLAT (une opération d'ébauche et une de finition) et les deux dernières (CONTOURING_ROUGH et CONTOURING_FINISH) qui correspondent aux opérations de contournage d'ébauche et de finition des features OUTER_DIAMETER correspondant au cône et au cylindre.

```
/*Operation*/
#500=FACING_ROUGH($,$,'dressage ebauche 01',$,$,#600 ,#801 ,#800 ,#910 ,#911 ,#900 ,0.500);
#501=FACING_FINISH($,$,'dressage finition 01',$,$,#610 ,#803 ,#800 ,#910 ,#911 ,#901 ,$);
#502=CONTOURING_ROUGH($,$,'contouring ebauche 01',$,$,#600,#802,#800,#912,#913,#902,0.500);
#503=CONTOURING_FINISH($,$,'contouring finition 01',$,$,#610,#804,#800,#912,#913,#903,0.000);
```

Figure 4.8 : Operation

Le déroulement des opérations définies dans la norme International Organization for Standardization (2003c) sont représentées Figure 4.9 et Figure 4.10.

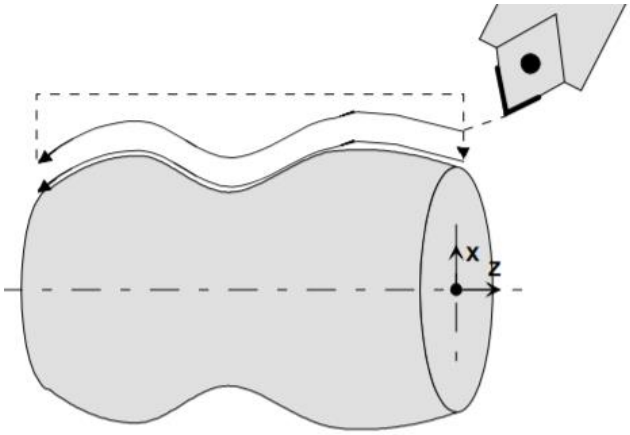


Figure 4.9 Contouring (International Organization for Standardization, 2003c)

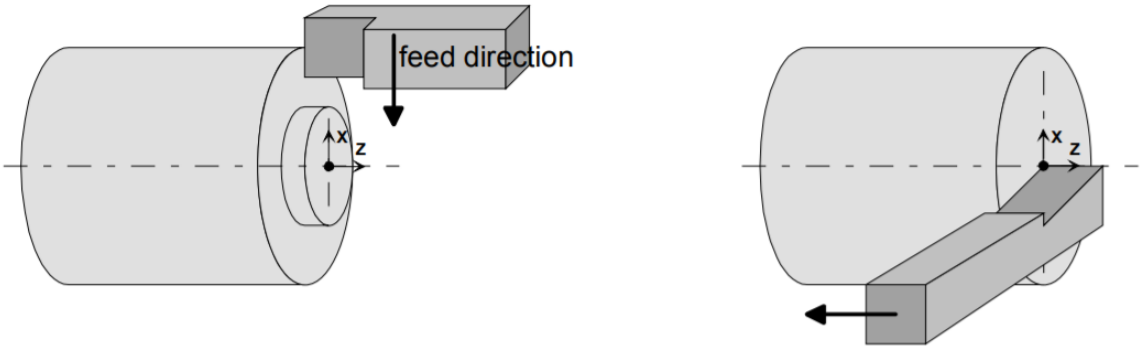


Figure 4.10 Facing

4.3.1.2.4 Project

```
/*Project*/
#1=PROJECT('EXEMPLE TOURNAGE 02',#2 ,(#100) ,#3,$,$);
#2=WORKPLAN('MAINWORKPLAN',(#3,#4) ,$,#5 ,$);
#3=MACHINING_WORKINGSTEP('Dressage ebauche 01',#700 ,#400,#500,$);
#4=MACHINING_WORKINGSTEP('Dressage finition 01',#700 ,#400,#501,$);
#5=SETUP('SETUP POUR EXEMPLE DE TOURNAGE 02',$,#700,(#201));
#6=WORKPIECE_SETUP(#100,#710 ,$,$,());
```

Figure 4.11 Project

La sous-section *PROJECT* permet de définir la séquence des phases d'usinage. L'entité *WORKPLAN* correspond à une phase d'usinage et donne la séquence d'exécution des *WORKINGSTEP* à effectuer et l'entité *PROJECT* définit le ou les *WORKPLAN(s)* et la ou les pièce(s) utilisée(s). Chaque *WORKINGSTEP* correspond à l'union d'une unique *OPERATION* et d'une ou plusieurs FEATURE(s). La pièce en exemple en possède donc quatre opérations, qui correspondent ainsi aux quatre *OPERATIONS* vues précédemment. On définit enfin une entité *SETUP* qui décrit la mise en position de la pièce dans la machine pour le *WORKPLAN*.

4.3.1.2.5 Functions / Technology

```
/* ***** Functions / Technology ***** */
#40=TURNING_MACHINE_FUNCTIONS(.T.,$,$,(),.F.,$,$,(),$,$,$);
#41=TURNING_TECHNOLOGY($,.TCP.,#45,0.300,.F.,.F.,.F.,$);
#42=TURNING_TECHNOLOGY($,.TCP.,#46,0.200,.F.,.F.,.F.,$);
#43=TURNING_TECHNOLOGY($,.TCP.,#47,0.300,.F.,.F.,.F.,$);
#44=TURNING_TECHNOLOGY($,.TCP.,#48,0.200,.F.,.F.,.F.,$);
#45=CONST_SPINDLE_SPEED(5.000);
#46=CONST_CUTTING_SPEED(2.500,10.000);
#47=CONST_CUTTING_SPEED(2.500,10.000);
#48=CONST_CUTTING_SPEED(2.200,10.000);
```

Figure 4.12 Functions and Technology (ISO, 2003c)

La sous-section *FUNCTIONS/TECHNOLOGY* permet de définir les paramètres machines qui doivent être appliquées pour chaque opération, comme la vitesse de la broche, la vitesse d'avance, l'utilisation d'huile de coupe, etc. Dans notre cas ont défini une vitesse d'avance de 0,3 mm par révolution pour les opérations d'ébauches et 0,2 mm par révolution pour les opérations de finition.

4.3.1.2.6 Strategie

La sous-section *STRATEGIE* permet de définir les stratégies d'usinage, dans notre exemple Figure 4.13 on utilisera un tournage unidirectionnel (*UNIDIRECTIONAL_TURNING*) pour réaliser les opérations *FACING_ROUGH* et *FACING_FINISH*. On définit aussi les stratégies d'approche et de retrait, ici on utilise une approche tangente à la *FEATURE* d'un rayon de 60 mm et un retrait en angle de 45° sur une longueur de 4 mm.

```
/*Strategies*/
#900=UNIDIRECTIONAL_TURNING($,$,(2.000),$,#720,#720,$,$,1.000,$,$);
#901=UNIDIRECTIONAL_TURNING($,$,(0.500),$,#720,#720,$,$,2.000,$,$);

#910=AP_RETRACT_TANGENT($,60.000);
#911=AP_RETRACT_ANGLE($,45.000,4.000);
```

Figure 4.13 Strategies

4.3.1.2.7 Placements / Lengths / Planes

De nombreuses entités ont besoin de se référencer à des objets géométriques pour se situer dans l'espace et définir des sens et des directions. Cette sous-section permet de définir ces objets géométriques, comme des plans, des vecteurs et des points.

```
/*Placements / longueurs / Plans*/
#700=PLANE('secplane',#701);
#701=AXIS2_PLACEMENT_3D('Axis2_Placement',#702,$,$);
#702=CARTESIAN_POINT('SECPLANE: LOCATION',(44.000,0.000,100.000));

#710=AXIS2_PLACEMENT_3D('WORKPIECE',/#711,#712,#713);
#711=CARTESIAN_POINT('WORKPIECE: LOCATION',(0.000,0.000,0.000));
#712=DIRECTION('WORKPIECE: AXIS',(0.000,0.000,1.000));
#713=DIRECTION('WORKPIECE: REF_DIRECTION',(1.000,0.000,0.000));
```

Figure 4.14 Placement, length, plane

4.3.1.2.8 Tool

Cette dernière sous-section du fichier STEP-NC de notre exemple présente les outils qui seront utilisés pour l'usinage. Il donne uniquement des indications sur les dimensions, la forme et la matière que devrait avoir l'outil. En revanche, il ne donne pas l'outil exact à utiliser, ce sera au contrôleur de choisir l'outil présent dans la machine qui correspond aux mieux aux paramètres

donnés.

```
/*Tool*/
#600=TURNING_MACHINE_TOOL('outil ebauche',#601,(#602),$,$,$);
#601=GENERAL_TURNING_TOOL(#603,.LEFT.,$,$,$,$,$);
#602=CUTTING_COMPONENT(50.,#604,$,$,$);
#603=TURNING_TOOL_DIMENSION($,$,$,$,$,10.000,$,20.000,$,0.300,$);
#604=MATERIAL('T15K6','CEMENT CARBIDE',(#605));
#605=NUMERIC_PARAMETER('ELASTIC MODULUS',3.E11,'pa');
```

Figure 4.15 Tool

4.3.2 Structure générale du contrôleur

La structure logicielle du contrôleur peut être séparée en trois grands modules.

- L'interpréteur de fichier STEP-NC : ce module permet d'interpréter les informations présentes dans le fichier STEP-NC et d'instancier les objets C++ correspondants afin qu'ils puissent être exploités par le logiciel par la suite.
- Le générateur de parcours d'outils : il permet d'utiliser les informations précédemment interprétées pour générer le parcours d'outils et toutes les actions qui doivent être menées par la machine afin de fabriquer la pièce souhaitée.
- Le contrôleur bas niveau est la partie du logiciel qui doit fonctionner en temps réel puisqu'il est en charge d'envoyer les informations à la MOCN au moment où les actions doivent être effectuées. Il est composé de deux modules : le Numerical Control Kernel (NCK) qui interprète les informations du parcours d'outils et génère les signaux qui permettent le mouvement des axes de la MOCN. Le Programmable Logic Control (PLC) quant à lui gère tous les actionneurs auxiliaires tels que la vitesse de la broche, le magasin d'outil ou encore l'activation de l'arrosage. Toutes ces instructions sont envoyées à la MOCN et permettront de réaliser la pièce souhaitée.

La structure générale de la partie logicielle du contrôleur est résumée dans la Figure 4.16.

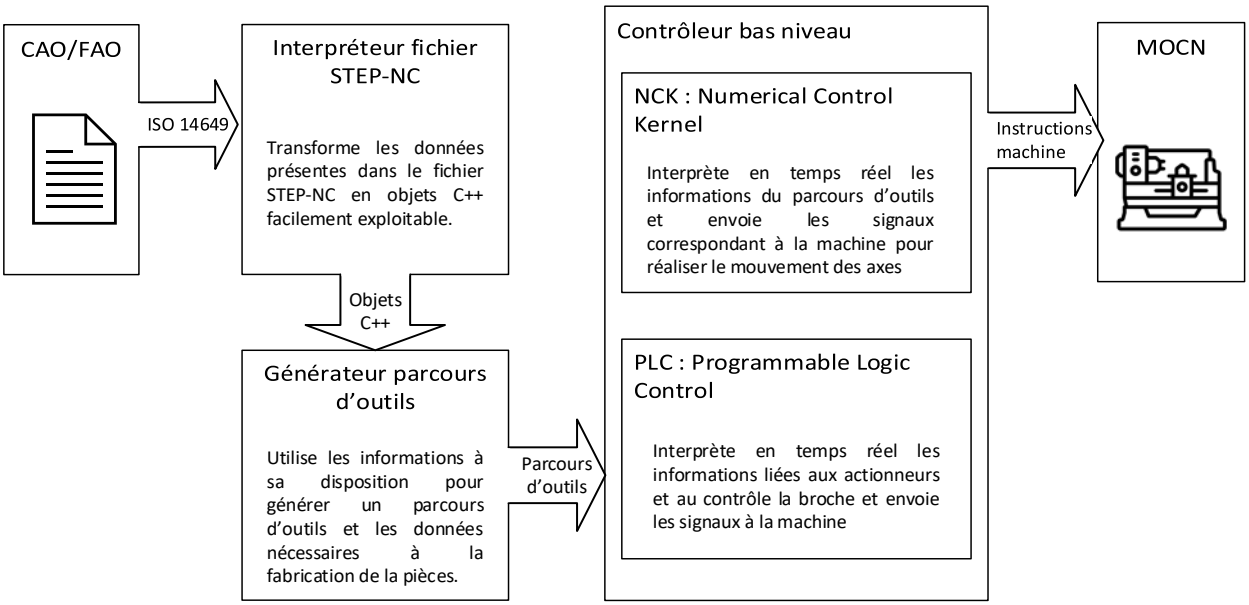


Figure 4.16 Structure générale de la partie logicielle du contrôleur

Les trois prochaines sous-parties permettront de présenter plus en détail le fonctionnement de ces modules.

4.3.3 Interprétation du langage STEP-NC

4.3.3.1 Création des classes

L'objectif de ce premier module est de lire et d'interpréter le fichier STEP-NC puis de stocker les entités présentes en instanciant des objets C++.

Pour cela, il faut créer une classe pour chaque entité que l'on souhaite instancier. La logique de création de classe est immédiate puisque STEP-NC est conçue sur une programmation par objet.

On peut prendre pour exemple l'entité *PROJECT* :

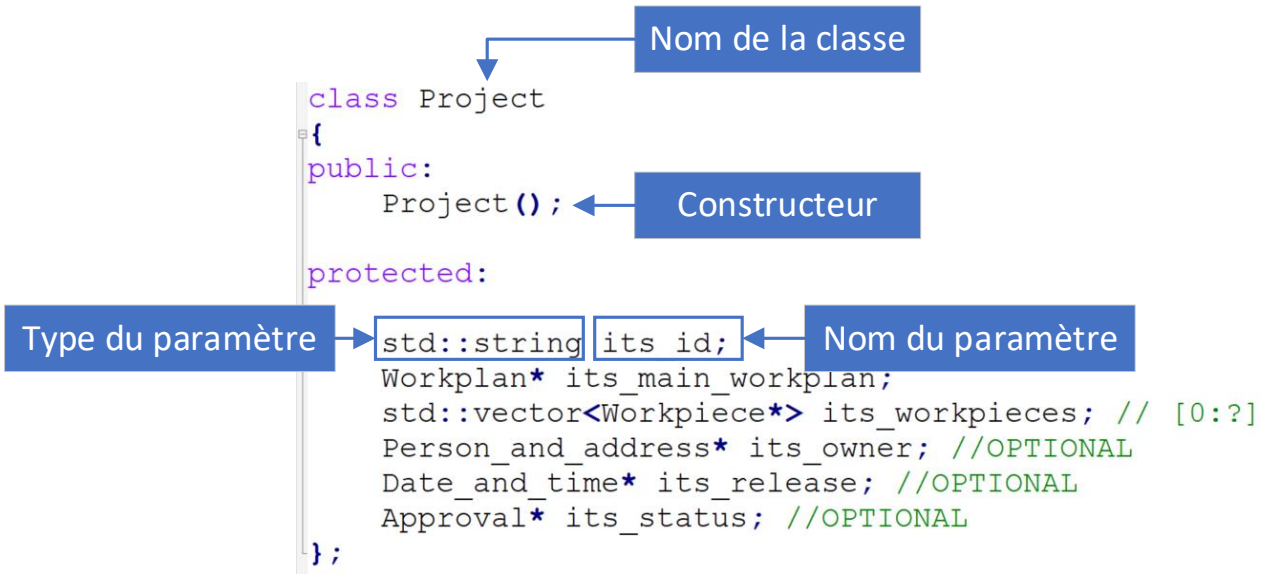
La norme ISO14649 spécifie la structure de l'entité *PROJECT* tel que présentée Figure 4.17.

```
ENTITY project;
   its_id:           identifier;
   main_workplan:    workplan;
   its_workpieces:   SET [0:?] OF workpiece;
   its_owner:        OPTIONAL person_and_address;
   its_release:      OPTIONAL date_and_time;
   its_status:       OPTIONAL approval;
   (*
   Informal proposition:
   its_id shall be unique within the part programme.
   *)
END_ENTITY;
```

its_id:	The project's identifier. It shall be unique within the part programme.
main_workplan:	The top-level workplan in this model.
its_workpieces:	The workpieces upon which actions are to be performed.
its_owner:	Optional information on the owner of the project.
its_release:	Optional date and time reference of the project.
its_status:	Optional attribute to indicate the current status of the project.

Figure 4.17 Structure de l'entité *PROJECT* (International Organization for Standardization, 2003a)

On peut ainsi créer de la même manière une classe *PROJECT* en langage C++ présentée Figure 4.18.




```
class Project
{
public:
    Project();

protected:
    std::string its_id;
    Workplan* its_main_workplan;
    std::vector<Workpiece*> its_workpieces; // [0:?]
    Person_and_address* its_owner; //OPTIONAL
    Date_and_time* its_release; //OPTIONAL
    Approval* its_status; //OPTIONAL
};
```

Figure 4.18 Implémentation de la classe *PROJECT*

Certaines entités sont définies comme des entités filles d'une entité mère, l'entité fille hérite alors des attributs de l'entité mère. On peut citer par exemple l'entité *MACHINING_WORKINGSTEP*

qui est une entité fille de *WORKINSTEP* qui est elle-même une entité fille de *EXECUTABLE* comme on peut voir Figure 4.19.

```
ENTITY executable
  ABSTRACT SUPERTYPE OF (ONEOF(workingstep, nc_function, program_structure));
  its_id:                identifier;
(*
Informal proposition:
its_id shall be unique within the part programme.
*)
END_ENTITY;

ENTITY workingstep
ABSTRACT SUPERTYPE OF (ONEOF (machining_workingstep, rapid_movement,
  touch_probing))
SUBTYPE OF (executable);
  its_secplane :          elementary_surface;
END_ENTITY;

ENTITY machining_workingstep
   SUBTYPE OF (workingstep);
   its_feature:           manufacturing_feature;
   its_operation:         machining_operation;
   its_effect:            OPTIONAL in_process_geometry;

END_ENTITY;
```

Figure 4.19 Entité *MACHINING _WORKINGSTEP* et ses classes mères

Ce principe d'hérédité est aussi implémenté dans notre programme afin de suivre la même structure que celle présente dans la norme et un exemple est visible Figure 4.20.

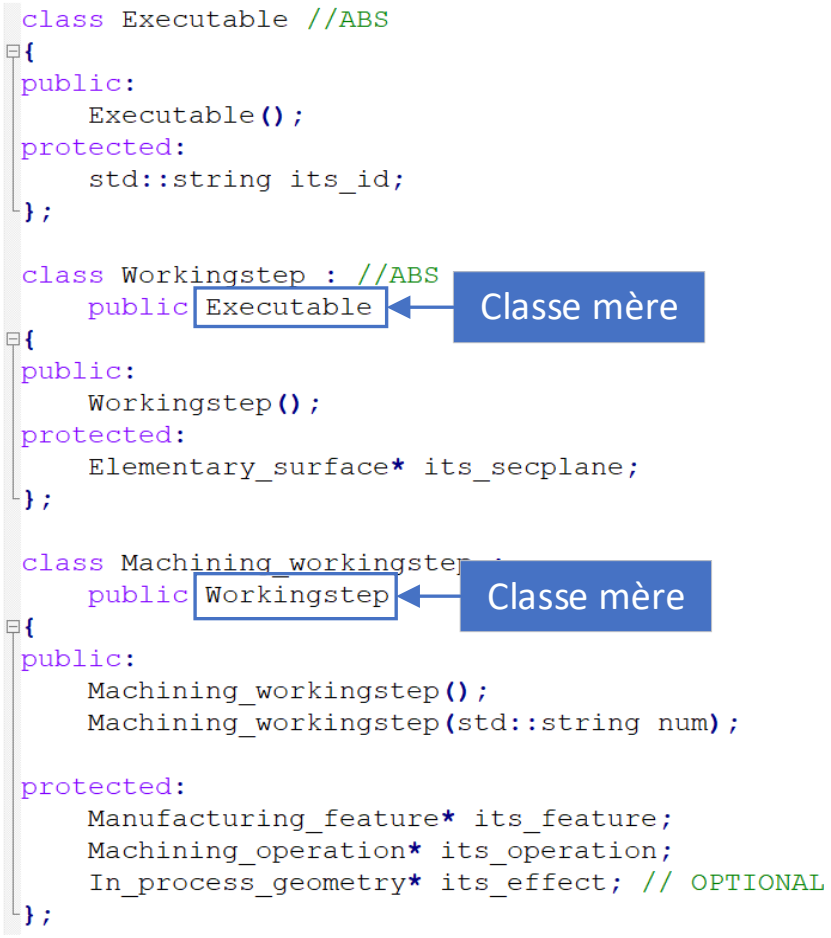


Figure 4.20 Classe *Machining_workingstep*

4.3.3.2 Instanciation des objets

Une fois les classes créées, on peut instancier les objets à partir des entités présentes dans le fichier STEP-NC. Pour cela, un outil a été développé pour lire chaque entité présente dans le fichier et instancier l'objet qui correspond avec les attributs donnés. Pour cela, on commence par récupérer l'entité *PROJECT* qui est unique dans chaque fichier et est la racine de toutes les autres entités. Si l'un de ces attributs fait référence à une entité (sous la forme #12 pour l'entité numéro 12 par exemple), on vient instancier cette entité et un pointeur vers cet objet sera stocké dans l'attribut correspondant. Ce processus est représenté Figure 4.22 et montre la méthode d'instanciation des objets présents dans l'exemple Figure 4.21.

```
#1=WORKPIECE('SIMPLE WORKPIECE',#2,0.010,#4,$,$,());
#2=MATERIAL('DIN EN 10027-1','E 295',(#3));
#3=NUMERIC_PARAMETER('ELASTIC MODULUS',2.E11,'pa');
#4=WORKPIECE('SIMPLE WORKPIECE',#2,0.010,$,$,$,());

#29=PROJECT('TURNING EXAMPLE 1',#30,(#1),$,$,$);

#30=WORKPLAN('MAIN WORKPLAN',(#31,#32,#33,#34),$,#37,$);

#31=MACHINING_WORKINGSTEP('WS ROUGH END FACE',#63,#10,#20,$);
#32=MACHINING_WORKINGSTEP('WS FINISH END FACE',#63,#10,#21,$);
#33=TURNING_WORKINGSTEP('WS ROUGH CONTOUR',#63,(#11,#12),#22,$);
#34=TURNING_WORKINGSTEP('WS FINISH CONTOUR',#63,(#11,#12),#23,$);
#37=SETUP('SETUP FOR TURNING EXAMPLE 1',$,#63,(#38));
```

Figure 4.21 Exemple de fichier STEP-NC

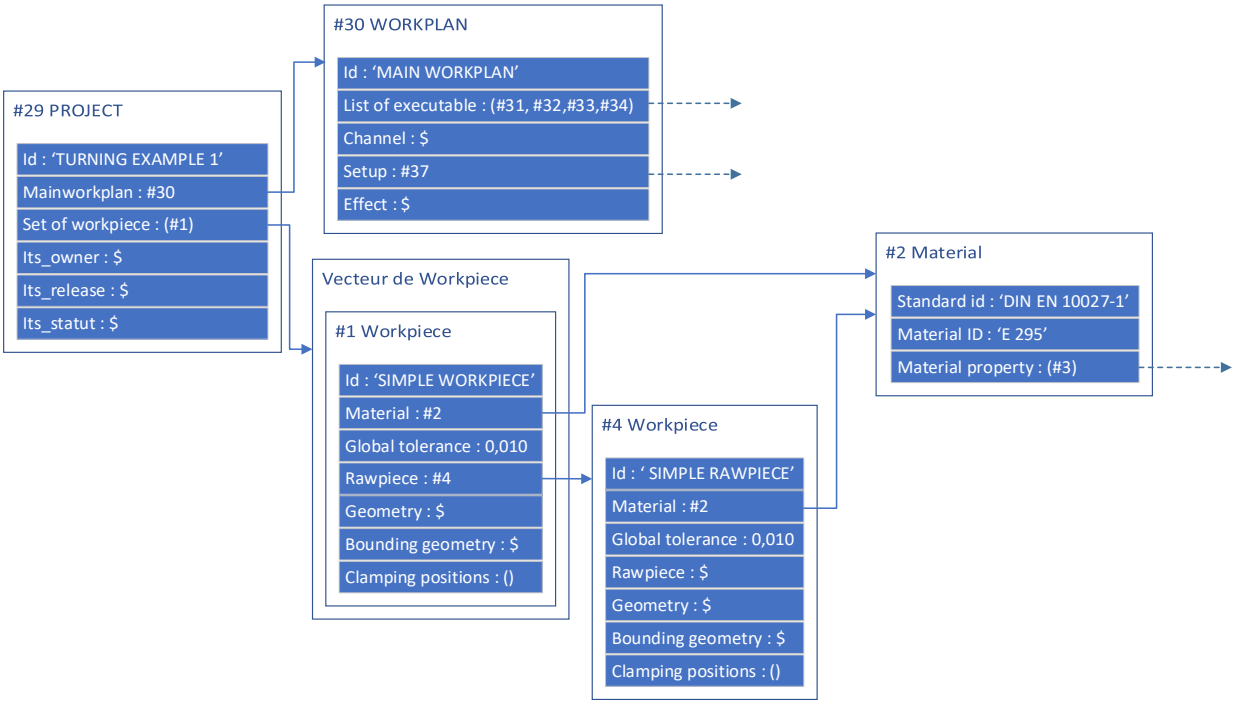


Figure 4.22 Processus d'instanciation des objets

Après l'instanciation de tous les objets, le programme possède en mémoire toutes les entités présentes dans le fichier STEP-NC avec leurs attributs. Il est maintenant possible d'utiliser ces objets très facilement pour la suite du programme, puisque les informations sont stockées de manière cohérente et donc accessibles facilement et rapidement.

4.3.4 Génération du parcours d'outils

On cherche maintenant à utiliser les informations précédemment stockées pour générer un parcours d'outils. En effet, contrairement au Code-G qui contient un parcours d'outils préétabli par le logiciel de FAO, le fichier STEP-NC ne contient pas de parcours d'outils explicite. Celui-ci doit être généré en fonction des différents paramètres présents dans le fichier STEP-NC, mais aussi en fonction de la machine (de la puissance ou du nombre d'axes par exemple), des outils disponibles, de la taille du brut, etc.

La génération du parcours d'outils est réalisée à l'aide d'une méthode contenue dans la classe *WORKINGSTEP*. Cette méthode est donc appelée pour tous les *WORKINGSTEPS* contenus dans le *WORKPLAN*. Pour chacun de ces *WORKINGSTEPS,* on vient récupérer les *FEATURES* et *l'OPERATION* qui correspond. Ces informations permettent de générer le parcours d'outils via la norme ISO1469 qui indique pour chaque *OPERATION* comment elle doit être exécutée en fonction de ces paramètres. Il faut donc implémenter une stratégie de génération de parcours d'outils pour chaque *feature*, c'est-à-dire développer un algorithme capable de générer le parcours d'outil en fonction des volumes à usiner et de la stratégie d'usinage. Dans notre exemple, ce sont les *features OUTER_DIAMETER* et *REVOLVED_FLAT* qui ont été implémentés avec les opérations de *FACING* et de *CONTOURING*. Un exemple de quelques entités qui sont utilisées pour gérer le parcours d'outils d'un WORKINGSTEP est présenté Figure 4.23 et sera stocké dans l'entité orange « *Generated machining toolpath* » de l'objet *MACHING_WORKINGSTEP*.

Le parcours d'outils ainsi généré est stocké dans un attribut de la classe WORKINGSTEP sous forme d'une liste de sections de parcours d'outils. Chaque section de parcours d'outils est composée de la coordonnée de départ (sous forme d'un vecteur à trois composantes), des trois coordonnées d'arrivée (sous forme d'un vecteur à trois composantes), du type de section (*linear* pour une section linéaire ou *circular* pour une section circulaire) et enfin du type de mouvement (*rapid_move, move, machining, approach, retract* respectivement pour un déplacement rapide, hors matière, en usinage, d'approche et de retrait). Ce dernier attribut permettra plus tard de déterminer la vitesse admissible des axes en fonction du type de mouvement.

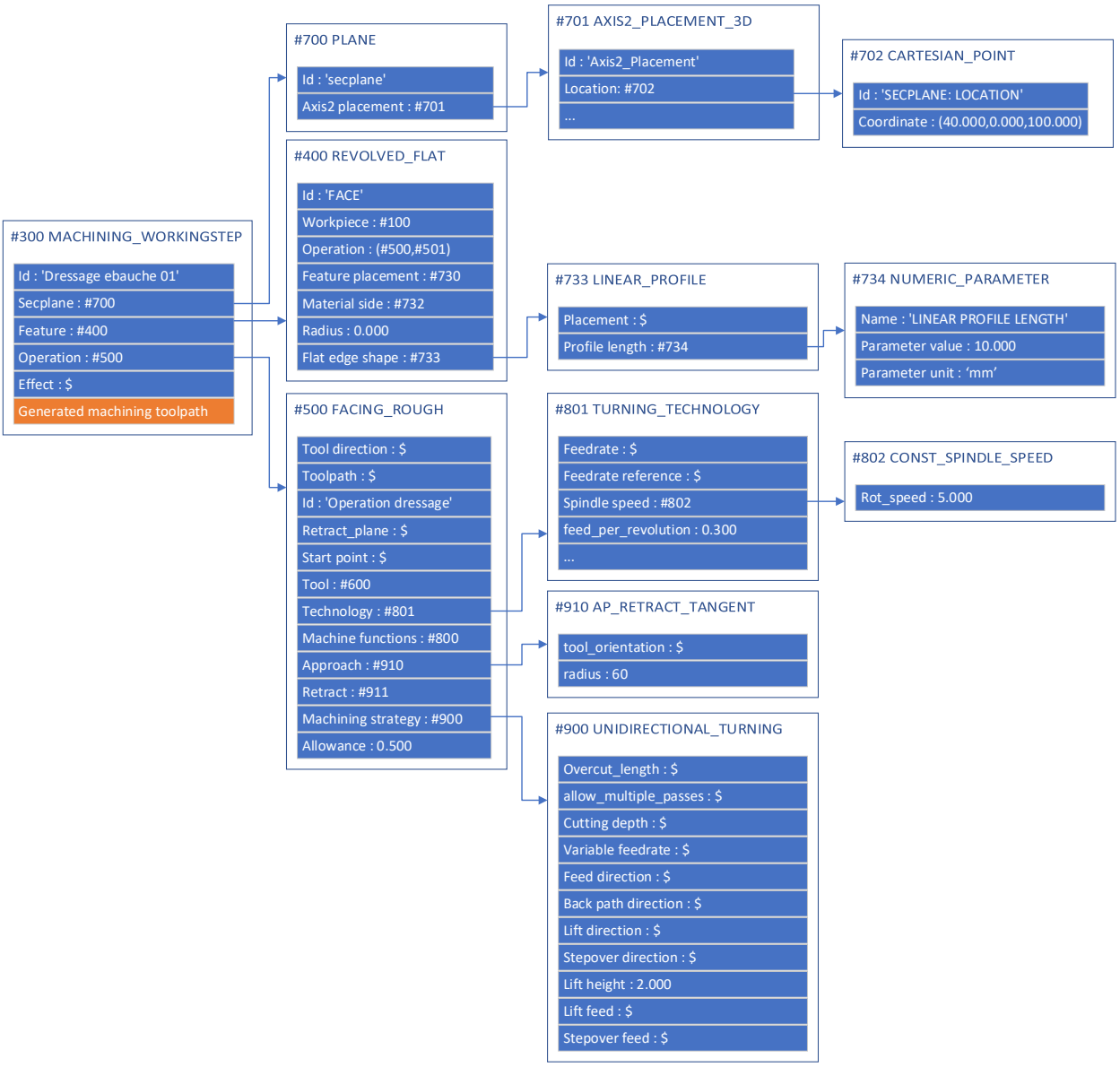


Figure 4.23 Exemple d'une partie des informations utilisées pour générer le parcours d'outils d'un *WORKINGSTEP*

L'ensemble des sections de parcours d'outils de chaque *WORKINGSTEP* forme le parcours d'outils total permettant d'usiner la pièce souhaitée.

4.3.5 NCK : Numerical Control Kernel

Le Numerical Control Kernel (NCK) ou en français : noyaux de commande numérique est un ensemble de modules qui ont pour but d'utiliser les données de parcours d'outils précédemment générées pour produire et envoyer des instructions pour les moteurs et actionneurs de la MOCN en

temps réel. Le NCK n'est pas unique à la structure d'un contrôleur STEP-NC, il est déjà présent dans un contrôleur Code-G traditionnel. La différence réside dans la quantité d'information présente qui est très faible en Code-G (uniquement le parcours d'outils) alors que pour STEP-NC on conserve toutes les données précédemment importées en plus du parcours d'outils généré. Ces informations pourront permettre par exemple de recalculer un nouveau parcours d'outils en cours d'usinage lors de la casse d'un outil en temps réel.

La structure générale du NCK pour un contrôleur OAC et la méthode pour implémenter les modules le constituant est traité dans le livre Theory and Design of CNC System (Suh, S.-H. et al., 2008) pour le cas d'un contrôleur Code-G. Une partie des connaissances sera réutilisée pour réaliser le développement du NCK de notre contrôleur. La structure générale des modules utilisés dans le NCK est résumée dans la Figure 4.24 et sera détaillée dans les parties suivantes.

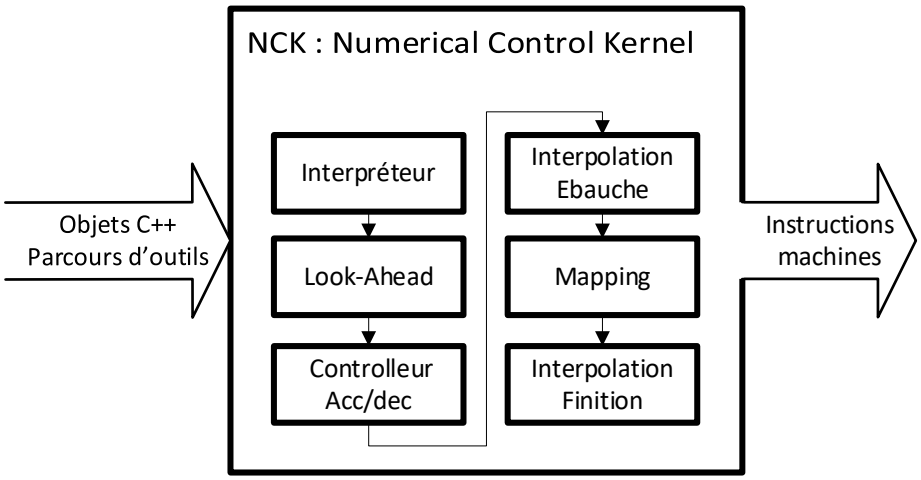


Figure 4.24 Structure du NCK

4.3.5.1 Interpréteur

Le module interpréteur consiste à récupérer les données nécessaires au mouvement des axes de la MOCN. Or dans notre cas, la lecture et l'interprétation des informations depuis le fichier STEP-NC a déjà été effectué auparavant. Le module doit donc uniquement récupérer les sections de parcours d'outils stockées dans chacun des *WORKINGSTEPS* dans l'ordre spécifié par le *WORKPLAN*.

4.3.5.2 Look-Ahead

Un parcours d'outils est souvent composé de nombreuses sections de parcours d'outil de courtes longueurs. Or si l'on réalise une interpolation de ces sections, il faut que la vitesse au début et à la fin de chaque section soit nulle pour pouvoir aborder la prochaine section, car on ne connait pas la liaison entre ces deux sections. Or cela nécessite un grand nombre d'accélérations et de décélérations, ce qui demande un temps d'usinage plus long et dégrade l'état de surface de la pièce.

Pour résoudre ce problème, on ajoute un module appelé *Look-Ahead* qui a pour but d'évaluer la vitesse admissible au début et à la fin de chaque section en fonction des sections suivantes.

Pour cela on utilise un algorithme issu de Theory and Design of CNC Systems (Suh, S.-H. et al., 2008) et présenté Figure 4.25 qui permet de déterminer la vitesse d'avance en début d'une section en fonction des N sections suivantes.

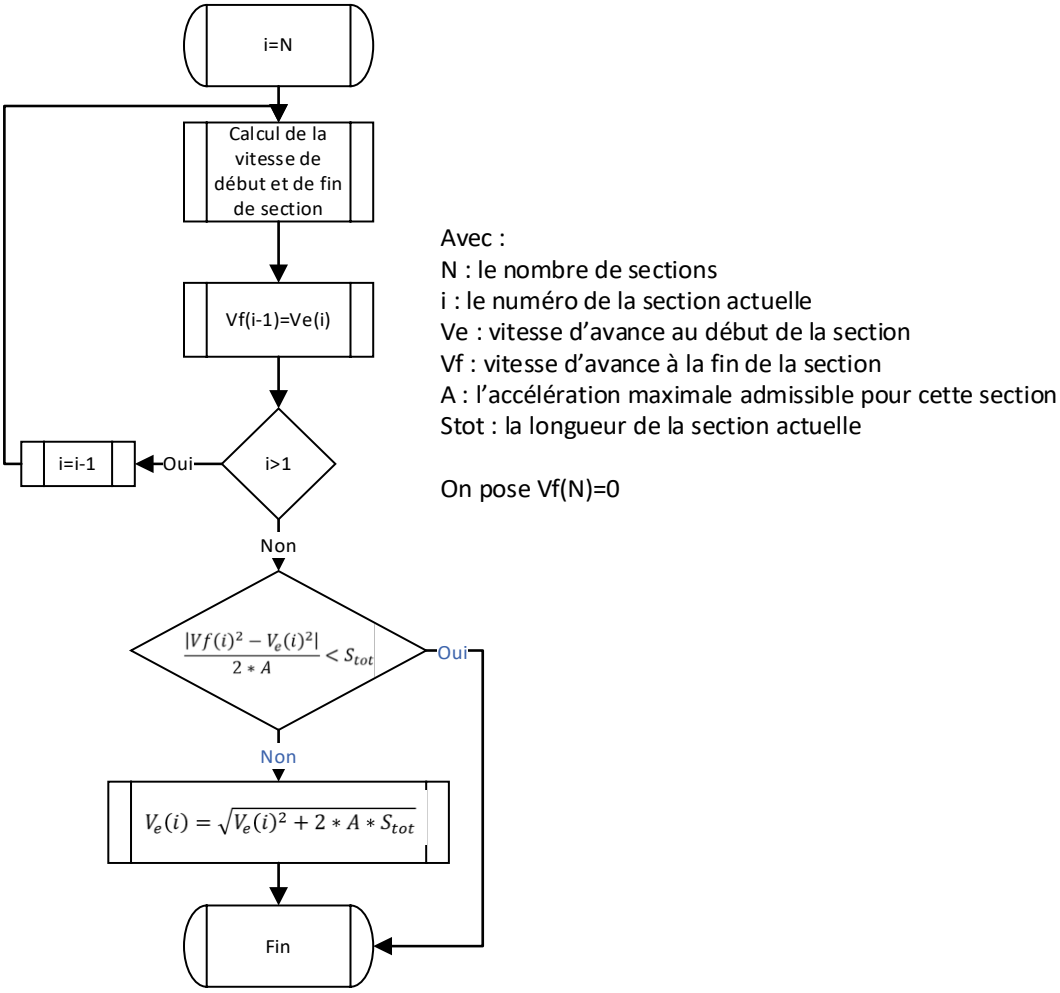


Figure 4.25 Organigramme du module *Look-Ahead* issue de (Suh, S.-H. et al., 2008)

On peut observer sur la Figure 4.26 l'avantage d'utiliser le module *Look-Ahead* pour optimiser la vitesse d'avance.

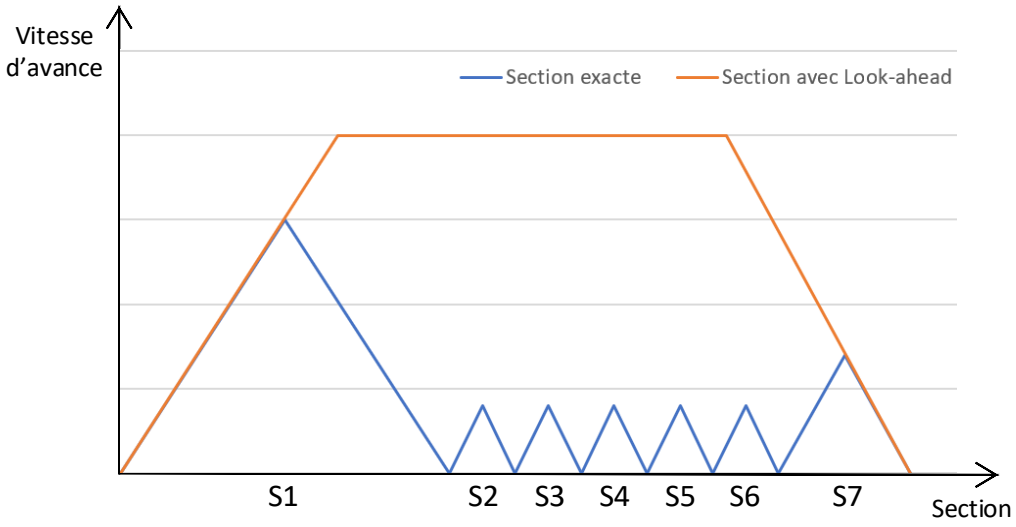


Figure 4.26 Comparaison de la vitesse d'avance avec et sans module Look-Ahead

4.3.5.3 Contrôle de l'accélération et la décélération

Une fois que la vitesse d'avance de début et de fin de chaque section du parcours d'outils a été déterminée, on cherche à déterminer la vitesse d'avance pour chaque intervalle d'échantillonnage, c'est-à-dire le profil de vitesse d'avance. Pour cela, on utilise la valeur de l'accélération et de décélération maximale admissible par la machine ainsi que la vitesse d'avance préconisée par le workingstep. On stocke ensuite une liste qui contient la vitesse d'avance pour chaque pas d'échantillonnage dans le *WORKINGSTEP* correspondant.

4.3.5.4 Interpolation ébauche

On possède maintenant la vitesse à chaque pas d'échantillonnage, on cherche maintenant à déterminer la position pour chacun des pas. On divise l'interpolation en deux modules, l'un pour les interpolations linéaires et l'autre pour les interpolations circulaires. Cette information est stockée dans chaque section du parcours d'outils. On peut alors calculer la position pour chaque intervalle avec la vitesse d'avance déterminée dans la partie précédente.

4.3.5.5 Mapping et Interpolation finition

Le module mapping permet de projeter les positions de l'outil précédemment calculées sur chacun des axes. Dans notre cas pour un tour, on projette la position sur l'axe Z et X de la MOCN.

On réalise ensuite une interpolation de finition qui permet de faire correspondre la fréquence d'échantillonnage du contrôleur avec celui du contrôleur de moteur de chaque axe de MOCN. En effet, on choisit une valeur de fréquence d'échantillonnage pour l'interpolation du parcours d'outils en fonction de la précision souhaitée et de la puissance de calcul disponible. On détermine à partir de la distance de déplacement entre chaque intervalle d'échantillonnage le nombre de pulsations nécessaire pour effectuer ce déplacement. Le contrôleur du moteur de la MOCN nous donne une valeur d'avance pour chaque pulsation envoyée. On expliquera dans la partie suivante le fonctionnement du driver.

La liste du nombre de pulsations par intervalle d'échantillonnage est ensuite stockée pour chaque axe dans l'objet *WORKINGSTEP* correspondant afin d'être utilisée par la suite pour déplacer les axes de la MOCN.

La Figure 4.27 synthétise les étapes qui ont été effectuées par les différents modules présentés précédemment permettant l'interprétation et le traitement des informations du fichier STEP-NC pour générer les informations nécessaires aux déplacements des axes de la MOCN.

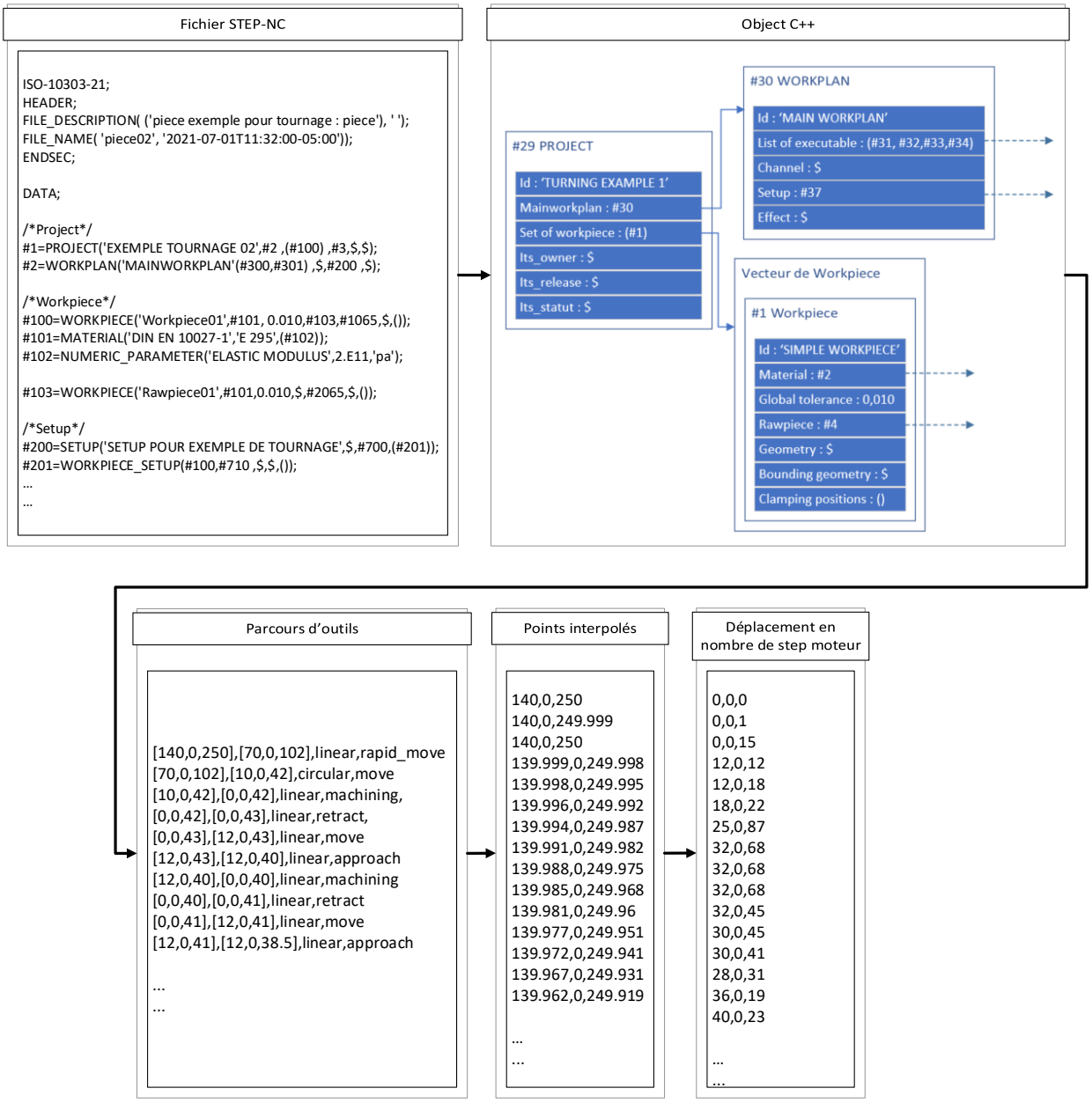


Figure 4.27 Schéma des étapes du traitement des informations d'un fichier STEP-NC

4.4 Conclusion

Nous avons développé dans cette partie un contrôleur STEP-NC de type 2, capable d'interpréter un fichier STEP-NC, de générer le parcours d'outils correspondant et de l'exécuter. Contrairement aux autres prototypes de contrôleurs de type 2 présents dans la littérature, notre prototype présente de nombreux avantages :

- Compacte : Le prototype est condensé dans un RPi dont les dimensions sont très compactes. Il est ainsi facilement intégrable dans une MOCN. L'ensemble du RPi et des cartes annexes mesure moins de 20*10*10 cm.
- Coût : Le prix de l'ensemble du matériel du projet est d'environ 200$ ce qui représente moins d'un dixième du coût des autres contrôleurs développés.
- Flexibilité : La structure matérielle permet d'ajouter de nombreux capteurs ou actionneurs qui peuvent être commandés par la RPi. De plus, le système d'exploitation de la RPi permet un développement de nouveaux modules aisément, dans plusieurs langages de programmation, notamment en Python et en C++. De plus, la connectivité de la RPi permet de communiquer des données et d'interagir avec d'autres systèmes de plusieurs manières : Ethernet, Wifi, USB, Bluetooth.
- Simplicité : Le faible nombre de composants utilisés assure une facilité d'installation de ce contrôleur dans une MOCN déjà existante.
- Reproductibilité : Au vu du coût et de la facilité à se le procurer, le prototype est facilement reproductible et pourra ainsi être testé et être développé. Aucun logiciel propriétaire n'est utilisé, toutes les licences utilisées sont open source, ce qui assure sa reproductibilité.

Dans le chapitre suivant, nous allons tester les capacités de notre prototype sur un cas réel en l'installant dans un tour à commande numérique. Nous présenterons aussi les fonctionnalités intégrées au logiciel du contrôleur.

CHAPITRE 5 VALIDATION EXPÉRIMENTALE

Ce chapitre a pour but de valider expérimentalement le fonctionnement du contrôleur précédemment développé. Nous présenterons tout d'abord le cas d'étude, puis nous détaillerons les modifications de la MOCN qui ont été effectuées afin d'installer le contrôleur STEP-NC. Finalement, les essais expérimentaux seront présentés ce qui nous permettra de conclure sur les performances du contrôleur développé.

5.1 Cas d'étude

Afin de valider notre proposition de contrôleur, nous allons usiner une pièce sur le tour à commande numérique qui sera modifié dans la section suivante afin d'accueillir le contrôleur STEP-NC. Pour cela, nous avons choisi de réaliser nos essais en usinant une pièce issue de la norme International Organization for Standardization (2003c). Son code STEP-NC est disponible Annexe A. Les caractéristiques du tour à commande numérique ainsi que de la pièce à usiner seront présentées. Une pièce étalon sera ensuite usinée à l'aide du contrôleur Code-G actuel, dans le but de vérifier la précision de l'usinage avec le nouveau contrôleur. Enfin, les fonctionnalités et l'interface du logiciel du contrôleur seront présentées.

5.1.1 Caractéristique de la MOCN utilisée pour la validation

Afin de réaliser les essais pour la validation, un tour à commande numérique deux axes EMCO PC TURN 55 sera utilisé. Il est équipé d'origine d'un contrôleur Code-G Fanuc 0-TC qui sera remplacé par notre contrôleur STEP-NC dans la section 5.2.

La vitesse maximale d'avance des axes est de 2 m/min et la taille maximale d'usinage est un cylindre de 52 mm de diamètre et de 215 mm de long. La machine possède une tourelle porte-outil comportant 8 emplacements. La broche est actionnée par un moteur asynchrone triphasé dont la vitesse varie entre 120 et 4000 tr/min.

Les axes de la machine sont actionnés par des moteurs pas-à-pas contrôlés par les contrôleurs de moteurs. Les contrôleurs de moteurs sont, quant à eux, contrôlés grâce à des signaux envoyés par le contrôleur. Chaque pulsation envoyée aux contrôleurs des moteurs permet de faire tourner le moteur d'un angle défini par les caractéristiques du moteur, ce qui correspond à une avance de

0,002 mm sur l'axe X et de 0,002 mm sur l'axe Z du tour EMCO. La fréquence des pulsations permet alors de gérer la vitesse d'avance.

La broche est actionnée par un moteur triphasé asynchrone et est contrôlée par un VFD (Variable Frequency Drive ou variateur électronique de fréquence) de la marque Lenze, lui aussi contrôlé via les informations envoyées par le contrôleur.

La Figure 5.1 illustre les composants qui sont présents à l'intérieur du carter arrière du tour EMCO.

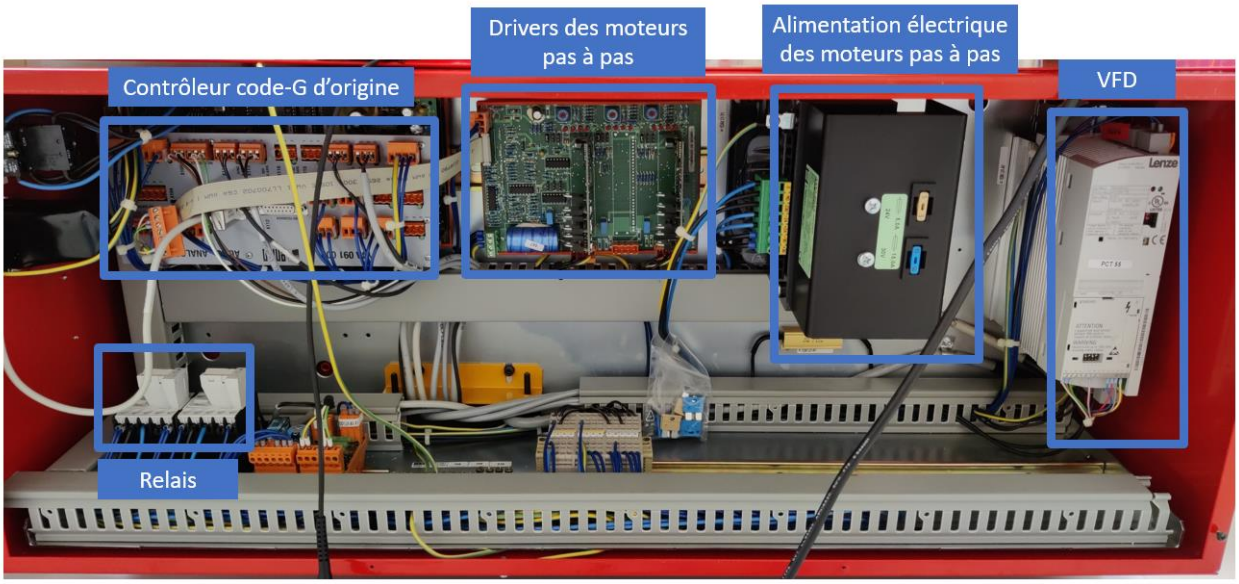
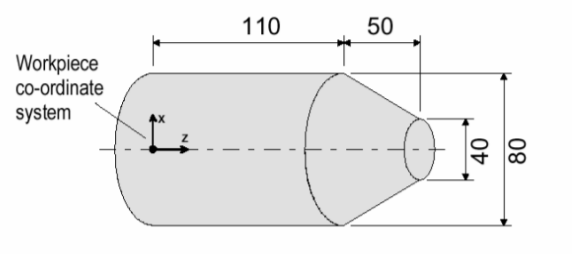


Figure 5.1 Composants du tour EMCO

5.1.2 Présentation de la pièce à usiner

La pièce a usiner est définie par la norme ISO14639. Elle est constituée d'une face, d'un cône et d'un cylindre dont les dimensions en millimètres sont données dans la Figure 5.2. Trois features ont été définis pour paramétrer le cône (OUTER_DIAMETER « CONE »), le cylindre (OUTER_DIAMETER « CYLINDRE ») et la face (REVOLVED_FLAT « END FACE ») et sont illustrées Figure 5.3.



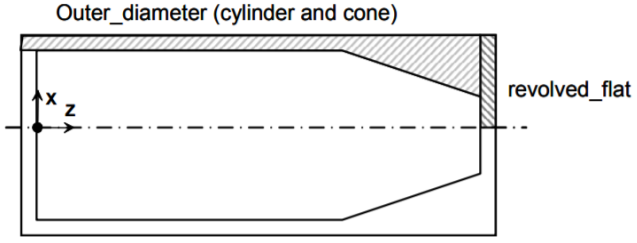


Figure 5.2 Schéma de la pièce Annexe D de la norme ISO14649 Part 12 (ISO, 2003c)

```
/* ***** Manufacturing features ***** */
#10=REVOLVED_FLAT('END FACE',#1,(#20,#21),#70,#80,0.000,#91);
#11=OUTER_DIAMETER('CONE',#1,(#22,#23),#76,#83,#93,#95);
#12=OUTER_DIAMETER('CYLINDER',#1,(#22,#23),#78,#72,#74,$);
```

Figure 5.3 *Features* de la pièce test

Afin de réaliser ses features, les opérations associées sont définies de la manière suivante et illustrées sur la Figure 5.4 :

- La feature REVOLVED_FLAT est usinée par une opération de dressage d'ébauche (FACING_ROUGH) puis de finition (FACING_FINISH).
- Les deux features OUTER_DIAMETER « CONE » et OUTER_DIAMETER « CYLINDRE » sont usinés à partir d'une seule opération de contournage d'ébauche (CONTOURING_ROUGH), puis d'une opération de contournage de finition (CONTOURING_FINISH)

```
/* ***** Turning operations ***** */
#20=FACING_ROUGH($,$,'ROUGH END FACE',$,$,#100,#41,#40,#52,#53,#50,0.500);
#21=FACING_FINISH($,$,'FINISH END FACE',$,$,#110,#42,#40,#52,#53,#51,0.000);
#22=CONTOURING_ROUGH($,$,'ROUGH CONTOUR',$,$,#100,#43,#40,#56,#56,#54,0.500);
#23=CONTOURING_FINISH($,$,'FINISH CONTOUR',$,$,#110,#44,#40,#56,#56,#55,0.000);
```

Figure 5.4 Operations de la pièce test

Les stratégies d'usinage sont ensuite définies Figure 5.5. On utilise un tournage unidirectionnel pour les opérations de dressage, avec une profondeur de passe de 3 mm pour l'ébauche et 0,5 mm pour l'opération de finition. L'approche est réalisée de manière tangentielle avec un rayon de 60mm et le retrait est effectué par un angle de 100° sur 2 mm. Pour l'opération de contournage, on utilise un tournage unidirectionnel avec une profondeur de passe de 3 mm puis une stratégie de contournage avec une passe de 0,5 mm. L'approche et le retrait sont effectués avec un angle de 45° sur une distance de 4 mm.

```
/* ***** Strategies ***** */
#50=UNIDIRECTIONAL_TURNING($,$,(3.000),$,$,#82,$,$,2.000,$,$);
#51=UNIDIRECTIONAL_TURNING($,$,(0.500),$,$,#82,$,$,2.000,$,$);

#52=AP_RETRACT_TANGENT($,60.000);
#53=AP_RETRACT_ANGLE($,100.000,2.000);

#54=UNIDIRECTIONAL_TURNING($,$,(3.000),$,$,$,$,$,2.000,$,$);
#55=CONTOUR_TURNING($,$,(0.500),$,$,#81,$,$,$,$,$);
#56=AP_RETRACT_ANGLE($,45.000,4.000);
```

Figure 5.5 Stratégies d'usinage de la pièce test

L'ordonnancement des opérations est défini dans le WORKPLAN qui donne l'ordre d'exécution des WORKINGSTEPS. On définit aussi le SETUP qui indique la mise en position du brut dans la machine. Enfin, l'entité Project est la racine du fichier STEP-NC. Ces entités sont illustrées Figure 5.6.

```
/* ***** Project ***** */
#29=PROJECT('TURNING EXAMPLE 1',#30,(#1),$,$,$);
#30=WORKPLAN('MAIN WORKPLAN',(#31,#32,#33,#34),$,#37,$);

#31=MACHINING_WORKINGSTEP('WS ROUGH END FACE',#63,#10,#20,$);
#32=MACHINING_WORKINGSTEP('WS FINISH END FACE',#63,#10,#21,$);
#33=TURNING_WORKINGSTEP('WS ROUGH CONTOUR',#63,(#11,#12),#22,$);
#34=TURNING_WORKINGSTEP('WS FINISH CONTOUR',#63,(#11,#12),#23,$);
#37=SETUP('SETUP FOR TURNING EXAMPLE 1',$,#63,(#38));
#38=WORKPIECE_SETUP(#1,#64,$,$,());
```

Figure 5.6 Section *Project* de la pièce test

D'autres entités sont définies pour caractériser les dimensions des outils qui doivent être utilisés ou encore des objets géométriques (point, vecteur, plan…) pour caractériser les différents placements dans l'espace.

5.1.3 Usinage d'une pièce étalon

Avant de réaliser les modifications du contrôleur sur la machine, la pièce présentée précédemment a été usinée à l'aide du contrôleur Code-G d'origine de la machine. Cela permettra de comparer avec la pièce obtenue via notre contrôleur STEP-NC. Au vu des performances limitées et de la taille de brut acceptable par le tour EMCO utilisé, nous avons choisi d'adapter les dimensions de

la pièce. De plus, la pièce est définie dans le standard en acier E295. Pour la même raison de limitation de puissance et de facilité d'usinage, un alliage de laiton sera utilisé. Le brut d'usinage a pour dimension : 25,4 mm de diamètre et de 76 mm de long. La Figure 5.7 présente les dimensions de la pièce usinée et du brut. On utilisera un seul outil à charioter-dresser pour l'ensemble des opérations.

Le programme d'usinage Code-G qui a été utilisé pour usiner la pièce est disponible Annexe B.

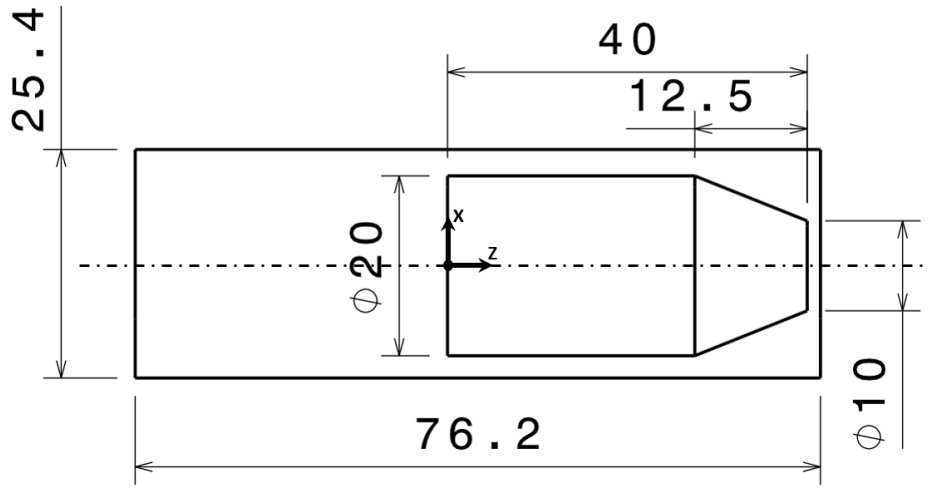


Figure 5.7 Mise en plan de la pièce pour tournage

La pièce obtenue est présentée sur la Figure 5.8. Plusieurs mesures ont été effectuées sur la pièce au niveau : de l'épaulement de 20 mm, du diamètre de 10 mm et des longueurs de 40 mm et de 12,5 mm. Les mesures ont été effectuées à l'aide d'un pied à coulisse et correspondent aux dimensions attendues avec une précision au dixième de millimètre. En revanche, on remarque des complexités dans le processus de transmission des informations qui ont lieu dans la chaine numérique d'industrialisation : le programme a été exporté via un post-processeur présent dans Catia V5 pour un contrôleur de type Fanuc 0-TC qui correspond à celui présent dans le tour EMCO. Or, certaines instructions en Code-G, n'étant plus compatible avec le contrôleur de machine, ont dû être modifiées manuellement. De plus, un changement de dimension de brut a été effectué avant l'usinage, ce qui a obligé la modification du programme dans le logiciel de CFAO Catia et un nouvel export du programme d'usinage. Le programme Code-G doit ensuite être transféré via une clé USB depuis le poste de FAO sur la machine. On retrouve ici un cas réel des inconvénients que peut apporter l'utilisation du Code-G dans la chaine numérique d'industrialisation.

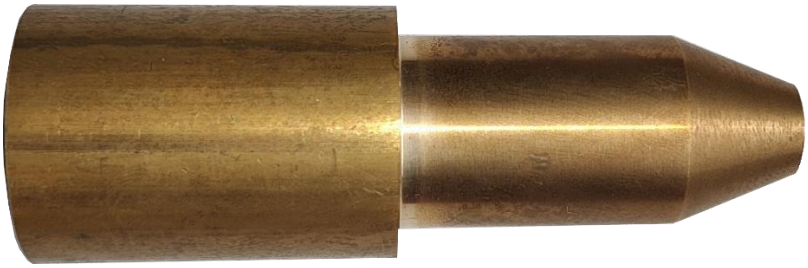


Figure 5.8 Pièce du standard, usinée avec le contrôleur Code-G d'origine du tour EMCO

5.2 Modification du contrôleur de la MOCN

5.2.1 Contrôleur Code-G actuel

Le contrôleur d'origine du tour EMCO PC TURN 55 est un contrôleur Code-G (EMCO WinNC GE Fanuc Series 0-TC), dont la partie matérielle est présentée Figure 5.9.

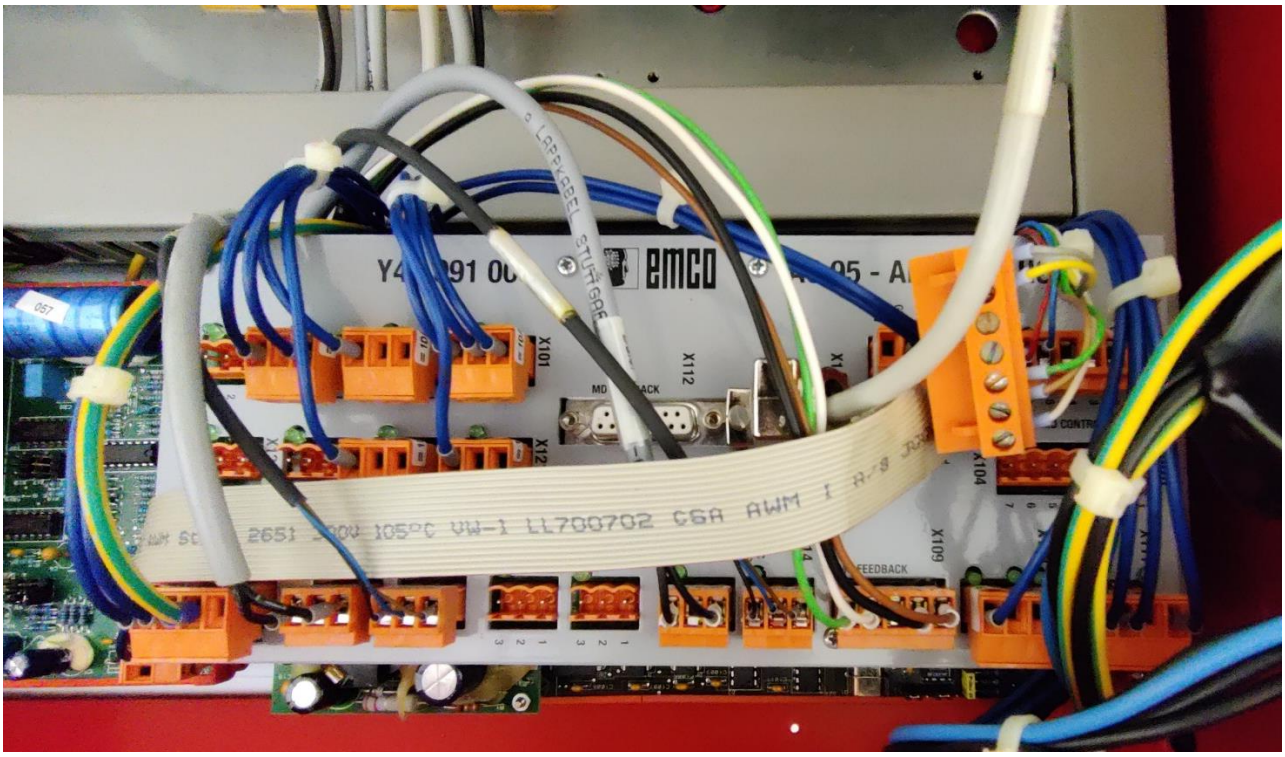


Figure 5.9 Contrôleur Code-G d'origine du tour EMCO PC TURN 55

Le tour est équipé d'un clavier de commande, visible Figure 5.10, qui permet à l'opérateur de réaliser les interactions avec la machine telle que le déplacement manuel des axes, le choix de la

vitesse d'avance, le lancement/arrêt du programme d'usinage, le choix de la vitesse de la broche, etc.

Figure 5.10 Clavier de commande et PC de contrôle

La particularité de ce tour, qui est destiné à un usage éducatif, est la présence d'un PC, visible Figure 5.10 qui permet de réaliser les mêmes actions que le clavier de contrôle, mais au travers du logiciel WinNC développé par EMCO et qui permet l'affichage visuel des informations d'usinages.

Le contrôleur Code-G présent actuellement dans la MOCN reprend les désavantages d'un contrôleur Code-G que nous avions cités dans le chapitre 2. De plus, la MOCN étant ancienne, on retrouve des fonctionnalités limitées et une impossibilité de faire évoluer ces fonctionnalités. De plus, le logiciel WinNC est uniquement compatible avec Windows 98. Le remplacement de l'ancien contrôleur Code-G par le nouveau contrôleur STEP-NC est donc bien approprié sur cette MOCN.

5.2.2 Installation du nouveau contrôleur STEP-NC

Le branchement du contrôleur STEP-NC s'effectue de la manière suivante :

- Contrôle des axes X et Z : Les signaux pour le contrôle des axes sont acheminés par un bornier IDC 16 qui est relié à la BOB, vers les drivers des moteurs pas à pas. La prise d'origine peut alors être directement branchée sans modification du câblage. Cette prise transmet les informations de pas et de direction pour les drivers des moteurs des axes X et Z, ainsi que l'information de l'activation des axes.
- Contrôle de la broche : La gestion de la vitesse et de l'allumage de la broche est effectuée par les signaux envoyés directement depuis la BOB vers le VFD, qui contrôle le moteur de la broche. Un signal en modulation de largeur d'impulsion (MLI) est envoyé avec une tension entre 0V et 10V afin de contrôler la vitesse de la broche. Un relais sur la BOB permet d'alimenter ou non le signal +24V pour commander l'allumage de la broche. Enfin un signal pour le sens de rotation peut être envoyé (0V pour le sens horaire, 24V sens antihoraire) au VFD. Dans notre cas, il n'est pas branché, car nous utiliserons uniquement le sens horaire pour nos essais d'usinage.
- Alimentation : Le RPi est alimenté par son alimentation USB-C, mais il pourra être alimenté directement en 5V par l'alimentation disponible dans la machine. La BOB est alimentée en 5V par USB branché sur le RPi.

Le schéma de branchement Figure 5.11 résume les branchements effectués.

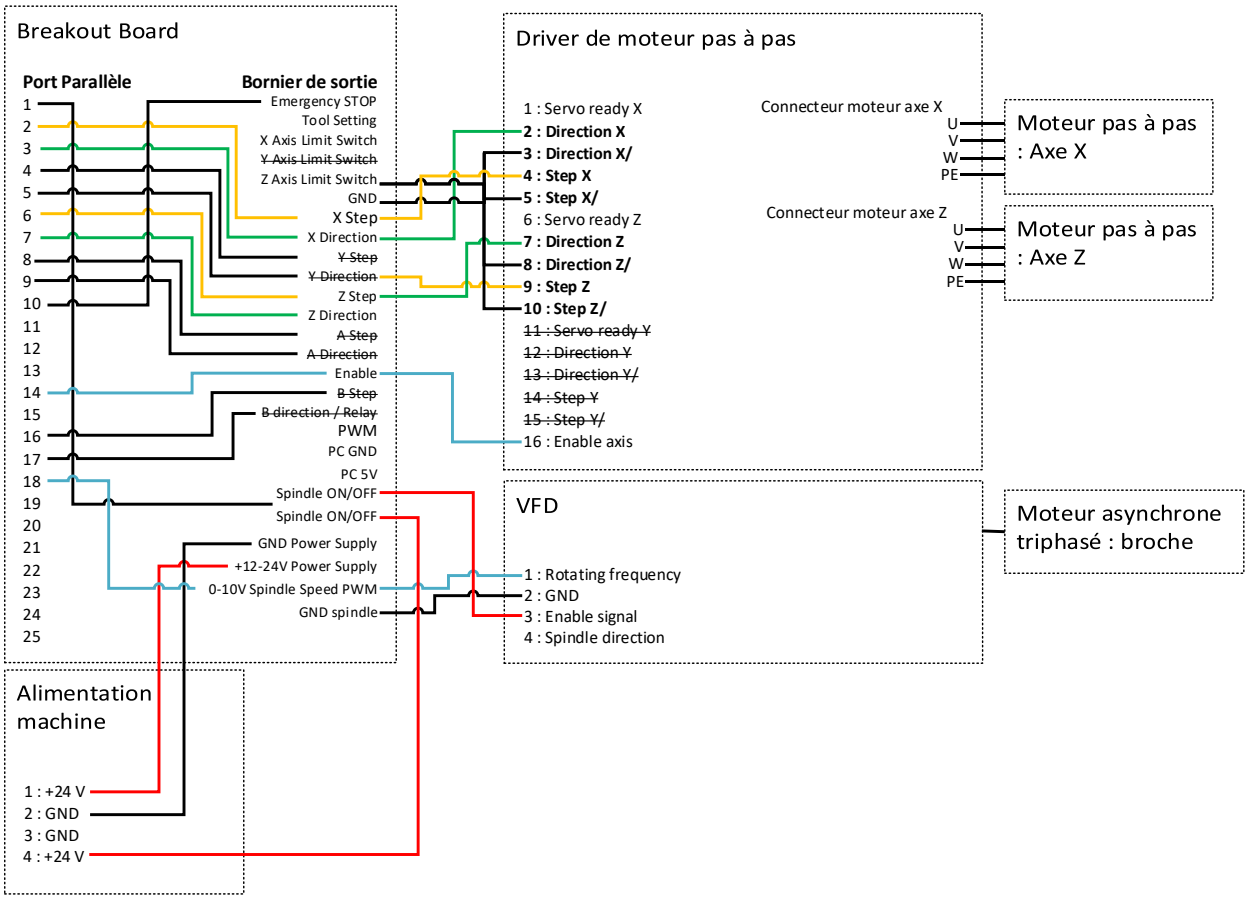


Figure 5.11 Schéma de branchement du contrôleur sur la MOCN

La Figure 5.12 permet de visualiser les branchements effectués dans la MOCN.

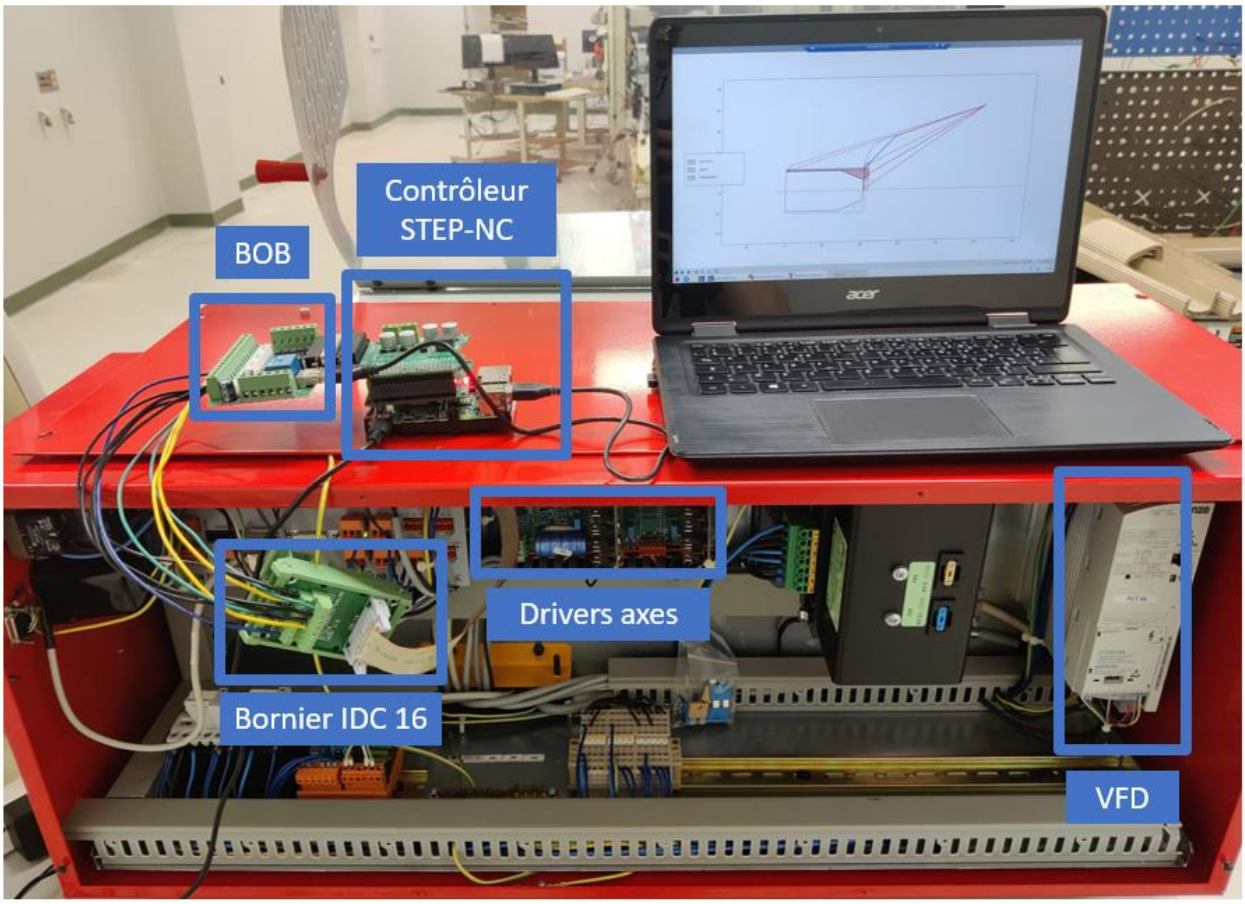


Figure 5.12 Branchement du contrôleur STEP-NC sur le tour EMCO PC TURN 55

Par simplicité d'utilisation, un PC portable est utilisé pour se connecter en bureau à distance au RPi via le réseau wifi local. On pourrait aussi brancher des périphériques (écran, clavier, souris) directement sur le RPi.

5.2.3 Interface et utilisation du logiciel du contrôleur STEP-NC

Cette partie va permettre de présenter le fonctionnement du logiciel du contrôleur et son interface.

Après avoir sélectionné le fichier STEP-NC, une interface représentée Figure 5.12, permet d'accéder aux différentes fonctionnalités présentes sur le logiciel et qui vont être développées dans les sous-parties suivantes.

Figure 5.13 Menu principal du logiciel

5.2.3.1 Interprétation

La fonction « Interprétation » permet d'interpréter les informations présentes dans le fichier STEP-NC en instanciant les objets correspondant aux entités présentent. Elle génère ensuite le parcours d'outils et toutes les informations nécessaires aux déplacements des axes et à la gestion des fonctionnalités de la machine. Comment on peut le voir Figure 5.14, le temps pour effectuer toutes les actions citées précédemment est indiqué. Pour le fichier de la norme présenté précédemment cela est effectué en 500 ms environ.

Figure 5.14 Fonction interprétation

5.2.3.2 Automatique

La fonction « Automatique » permet de réaliser la pièce présente dans le fichier STEP-NC, en envoyant aux actionneurs MOCN les signaux correspondants. On peut observer Figure 5.15, en haut à gauche l'affichage en temps réel des différents *workingstep* qui ont été effectués et l'affichage de la position de la broche en bas à droite.

Figure 5.15 Fonction automatique

5.2.3.3 Manuel

La fonction « Manuel » permet de réaliser le déplacement manuel des axes de la MOCN via le clavier branché sur la RPi. Il permet aussi de modifier la vitesse de déplacement et de modifier la vitesse et l'activation de la broche.

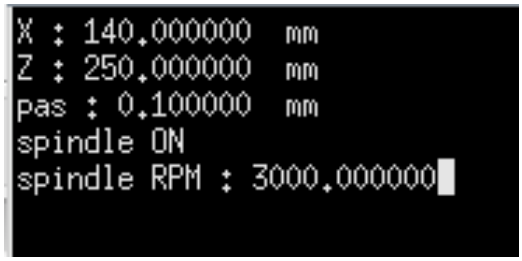


Figure 5.16 Fonction manuelle

5.2.3.4 Visualisation

La fonction « Visualisation » permet d'afficher la visualisation de la pièce et son parcours d'outils. La pièce est présentée Figure 5.17, en bleu foncé la pièce finie et en cyan le brut de la pièce.

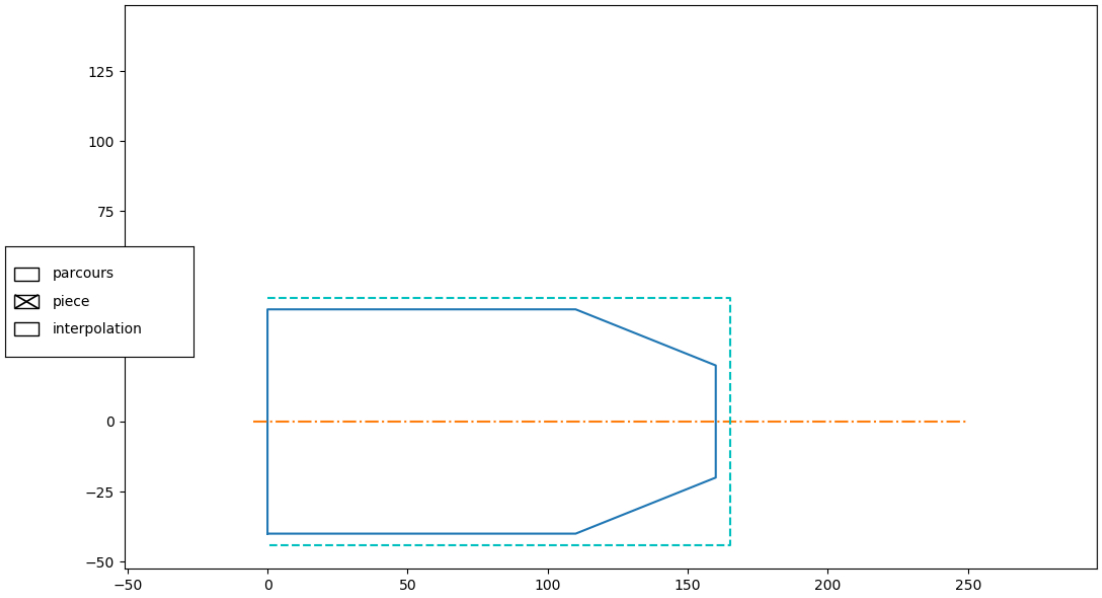


Figure 5.17 Fonction visualisation de la pièce

On peut aussi visualiser le parcours d'outils qui a été généré. En rouge, on retrouve les déplacements linéaires et en bleu les déplacements circulaires présentés Figure 5.18.

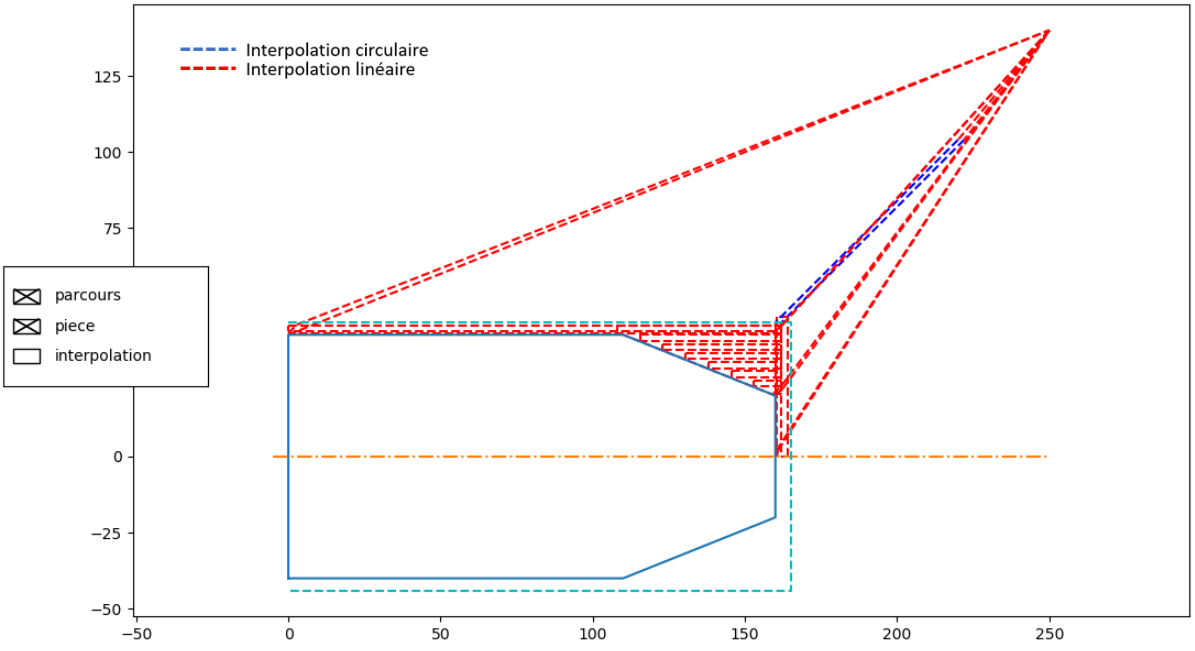


Figure 5.18 Fonction visualisation du parcours d'outils

On peut ensuite visualiser les points qui ont été interpolés grâce aux parcours d'outils présentés Figure 5.19. On affiche uniquement un point sur 300 pour des raisons de fluidité d'affichage.

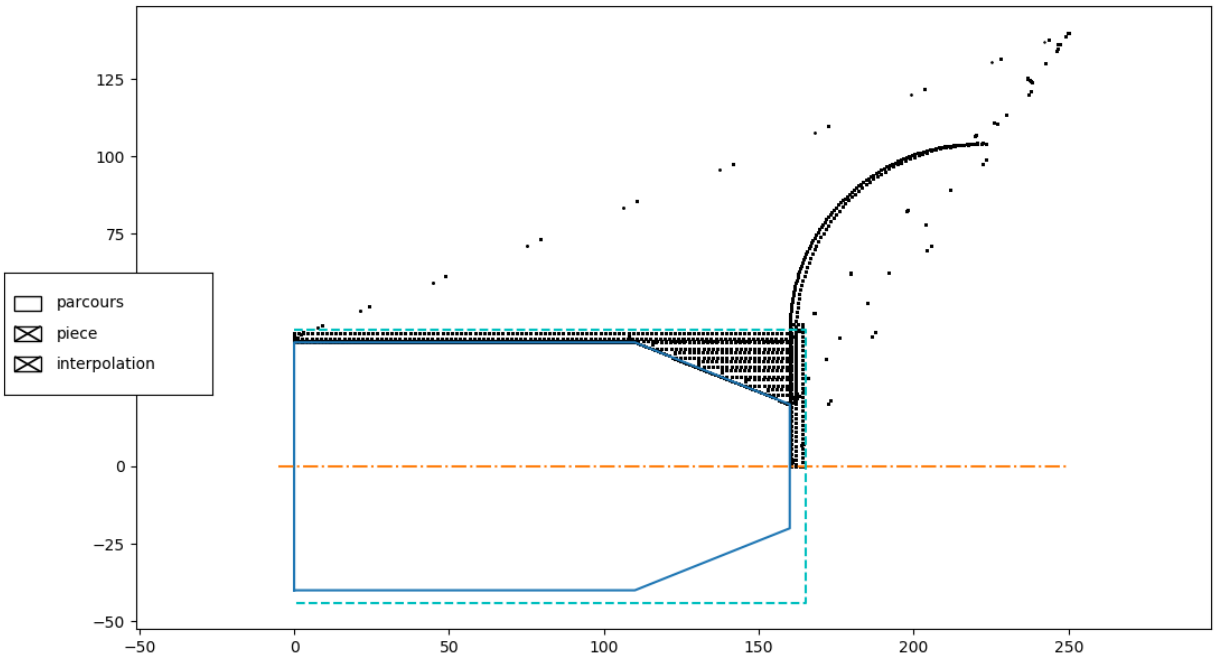


Figure 5.19 Fonction visualisation de l'interpolation

5.2.3.5 Animation

La fonction « Animation » génère une vidéo de simulation du déplacement des axes en temps réel en fonction des points d'interpolations qui ont été générés par l'interprétation. On peut voir un exemple de l'animation sur la pièce de la norme Figure 5.20.

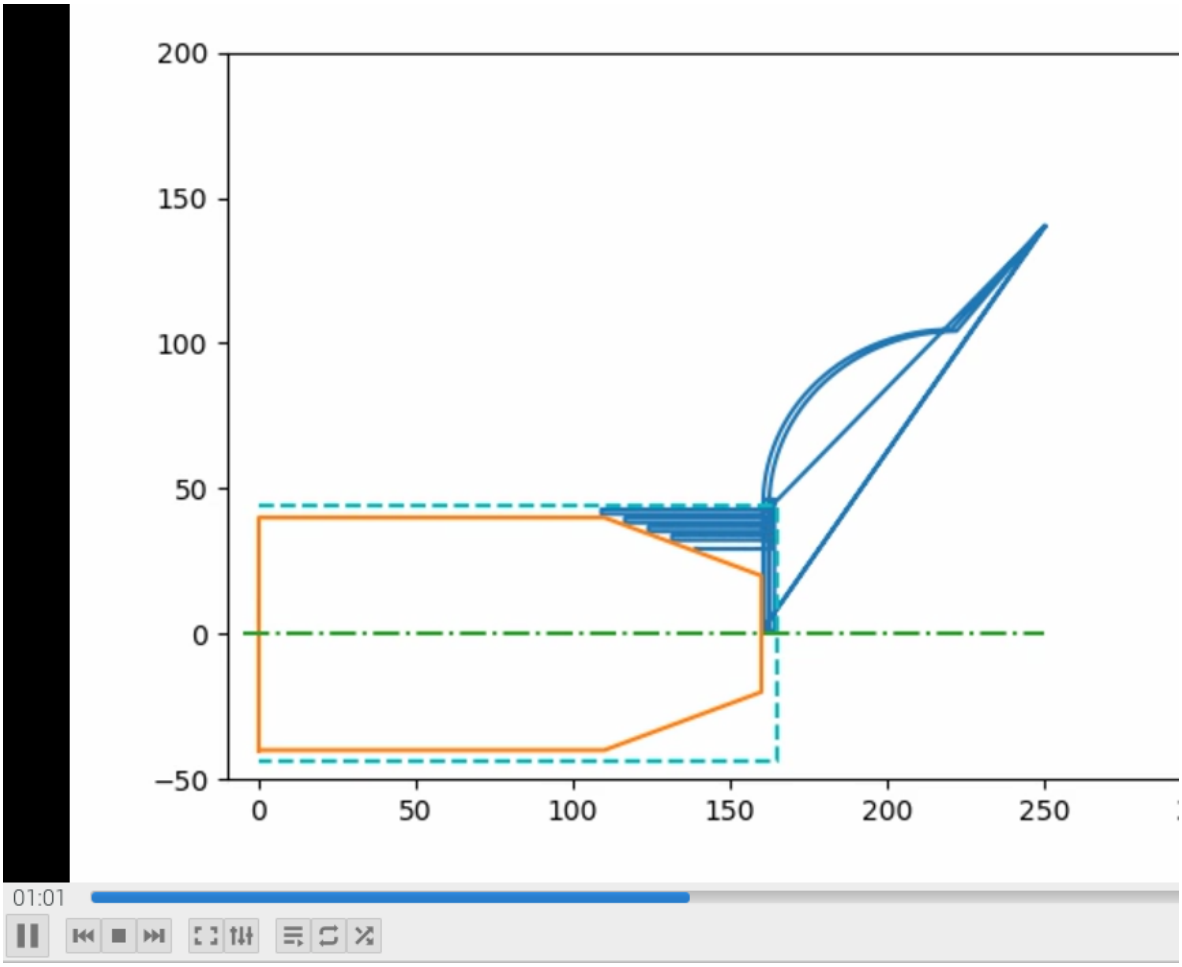


Figure 5.20 Fonction animation

5.2.3.6 Edit

La fonction « Edit » permet de modifier certains paramètres, par exemple les dimensions du brut qui va être utilisé pour usiner la pièce ou encore de donnée la position de départ de l'outil. Certain de ces paramètres sont présentés Figure 5.21. Pour exemple, mais de nombreux autres paramètres pourront être ajoutés.

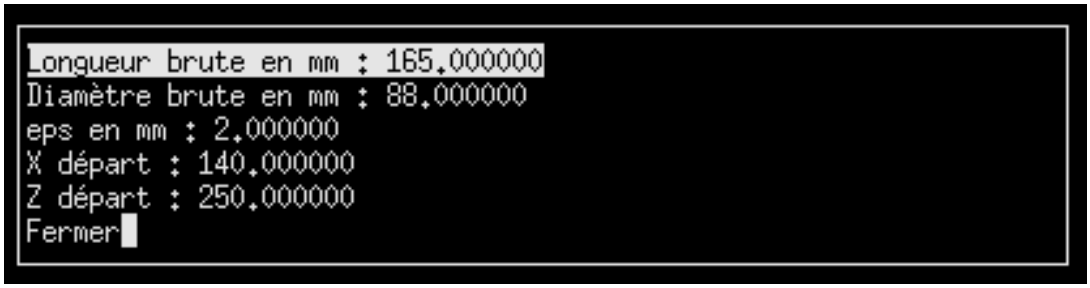


Figure 5.21 Fonction edit

5.2.3.7 Informations

La fonction « Informations » permet d'afficher des informations qui sont présentes dans le fichier STEP-NC comme le nom des *WORKINGSTEPS* qui vont être usinés ou encore les paramètres du *WORKPIECE.* Un exemple de quelques informations que l'on peut récupérer depuis le fichier STEP-NC est présenté Figure 5.22.

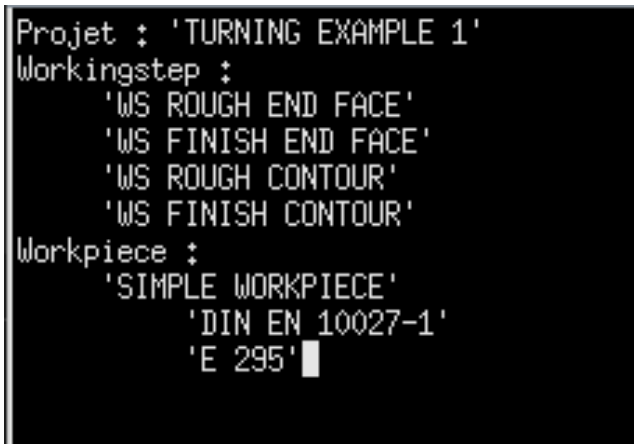


Figure 5.22 Fonction informations

5.3 Essais expérimentaux

L'objectif de cette partie est de présenter l'usinage de la pièce présentée dans le cas d'étude section 5.1 afin de valider le fonctionnement du contrôleur STEP-NC développé. On réalisera les essais sur le tour EMCO PC TURN 55 modifié dans la section précédente.

5.3.1 Protocole de validation

On réalise le protocole suivant afin de valider le fonctionnement du contrôleur :

- Le fichier STEP-NC contenant les informations de la pièce issue de l'annexe D du standard iso14640 part 12, est importé dans le logiciel du contrôleur.
- À l'aide du menu « *edit* », il est possible de modifier certains paramètres comme l'origine machine de l'outil ou les dimensions de la pièce brute utilisée.
- L'interprétation du fichier peut ensuite être effectuée à l'aide du menu « Interprétation ». L'opération est effectuée en 800 ms environ. Cela permet de générer le parcours d'outils en fonction des informations présentes dans le fichier STEP-NC et des paramètres de la machine.
- Le parcours d'outils peut être visualisé afin de vérifier sa validité à l'aide du menu « Visualisation » et une animation de l'usinage est disponible dans le menu « Animation » afin de visualiser une simulation de l'usinage en temps réel.
- Une fois les vérifications effectuées, l'usinage de la pièce peut être effectué par le menu « Automatique » qui permet de lancer l'usinage de la pièce.
- Une fois la pièce usinée, une vérification visuelle et une prise de mesure seront effectuées afin de valider le bon usinage de la pièce. Cependant, l'objectif n'est pas dans la précision de l'usinage qui pourra être amélioré par la suite, mais dans la bonne interprétation des informations du fichier STEP-NC et de son exécution.

5.3.2 Résultats

5.3.2.1 Simulation

Dans un premier temps une simulation est effectuée pour vérifier le parcours d'outils qui a été généré.

On commence par choisir le fichier STEP-NC qui correspond à la pièce du standard. On réalise ensuite l'interprétation via le menu « Interprétation ». Le parcours d'outil est alors généré, on peut ensuite visualiser le parcours d'outils et son interpolation via le menu « Visualisation ». Cela nous permet de vérifier la cohérence du programme qui va être exécutée. La Figure 5.23 est la visualisation du parcours d'outils et de son interpolation pour la pièce à usiner.

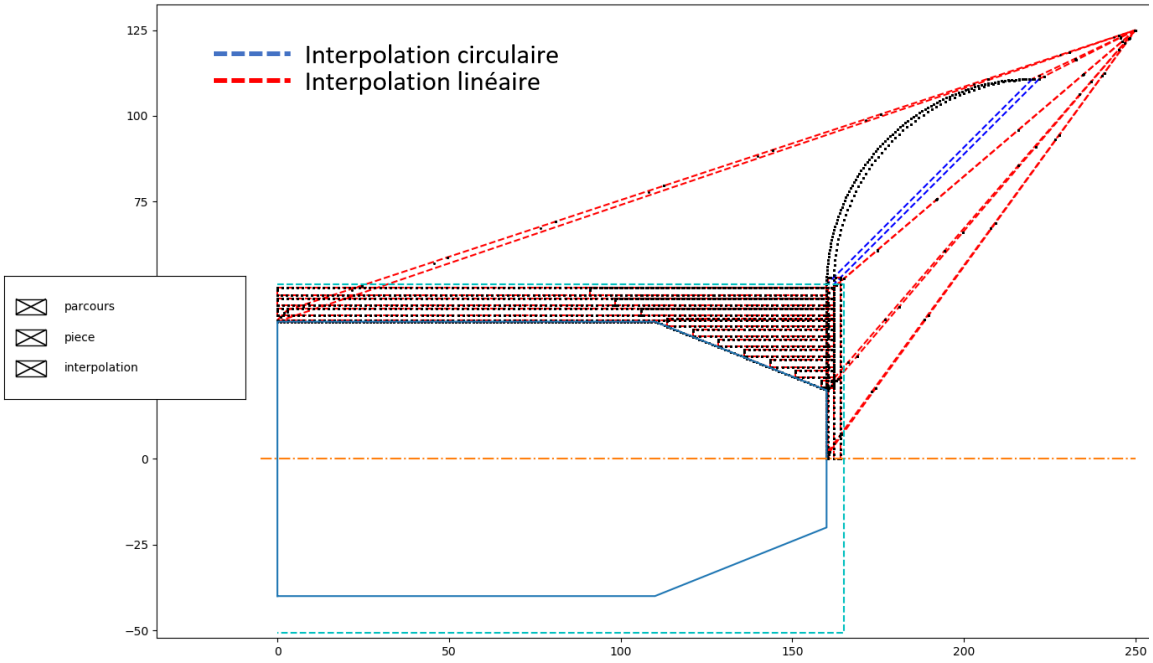


Figure 5.23 Visualisation du parcours d'outils et de son interpolation pour la pièce usinée

Cela nous permet aussi de mettre en avant l'un des avantages du contrôleur STEP-NC. Si l'on souhaite changer la taille du brut, comme cela a été le cas pour l'usinage de la pièce étalon en Code-G, il suffit de modifier les dimensions du brut dans le logiciel du contrôleur, via le menu « edit », directement depuis le plancher de l'atelier. Le parcours d'outils est alors adapté. La Figure 5.24 permet de visualiser deux exemples de parcours d'outils pour deux dimensions de bruts d'usinage différentes.

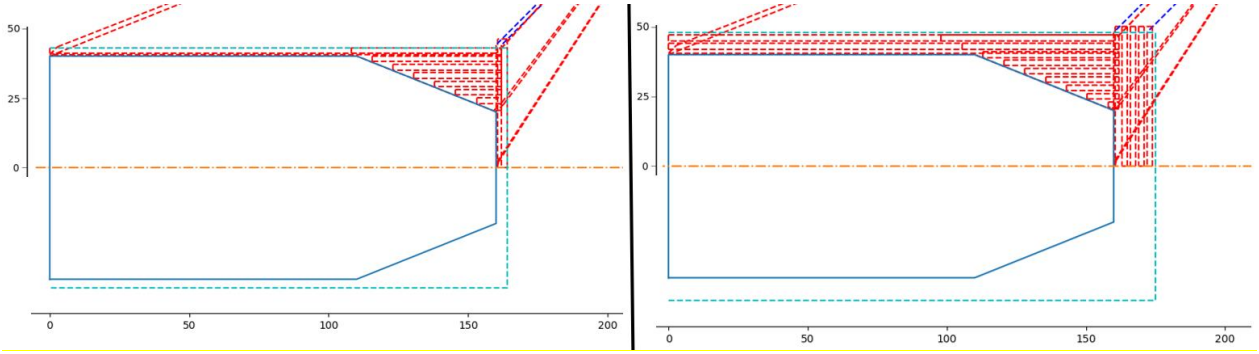


Figure 5.24 Exemple de visualisation du parcours d'outils pour deux dimensions de bruts différentes : gauche (longueur 165 mm, diamètre 88mm), droite (longueur 175mm, diamètre 96mm)

L’animation permet ensuite de vérifier les vitesses et le parcours d’usinage qui seront effectués. On peut alors valider le parcours d’outils par simulation et procéder à l’usinage.

5.3.2.2 Essai de tournage

Il faut rappeler qu’au vu des dimensions de notre tour d’expérimentation, l’usinage de la pièce du standard sera effectué avec une échelle un quart, qui est appliquée dans le logiciel du contrôleur. Cette mise à l’échelle permet d’utiliser le fichier présent dans le standard STEP-NC sans avoir à le modifier afin d’assurer la reproductibilité de l’expérimentation. C’est uniquement une fonction logiciel qui permet de réduire les dimensions des déplacements effectués.

Après avoir placé le brut d’une dimension de 76 mm de long et de 25,4 mm de diamètre dans les mors du tour. L’outil est ensuite déplacé via le mode de déplacement manuel comme visible sur la Figure 5.25, afin de définir l’origine de la pièce via le menu « edit ».

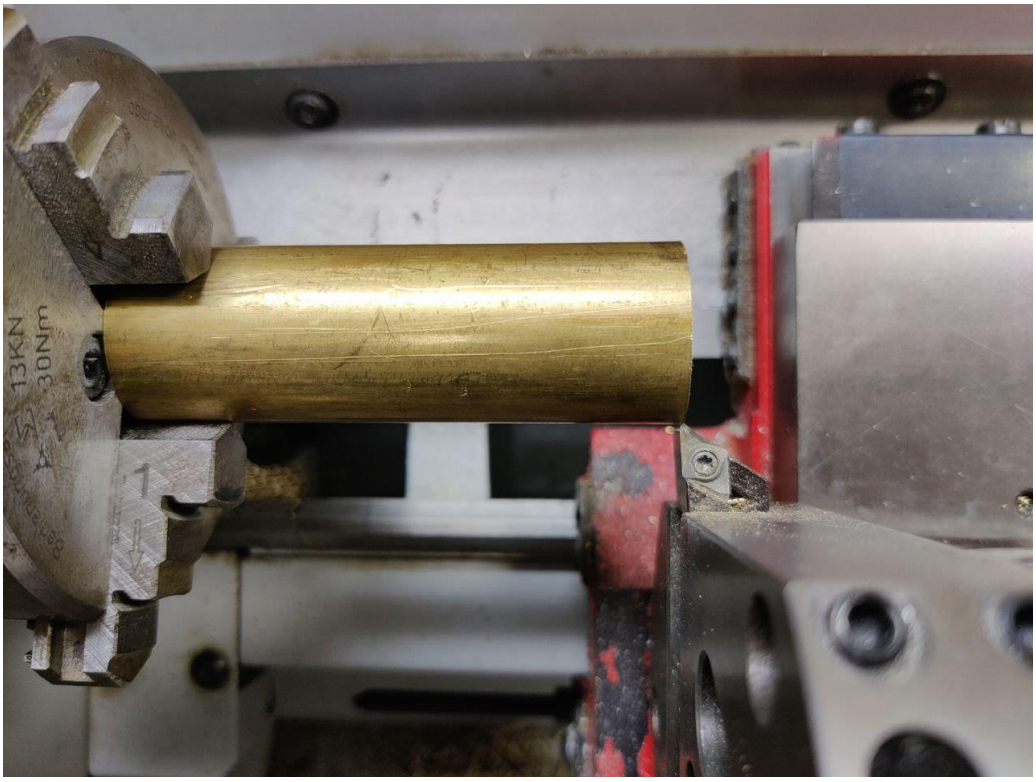


Figure 5.25 Prise d'origine pièce

L’usinage est lancé via le menu « Automatique ». Le programme réalise l’activation du moteur de la broche dans un premier temps, puis les axes X et Z sont mis en mouvement via les informations

envoyées par le contrôleur STEP-NC. On peut ainsi observer que le contrôleur STEP-NC développé permet bien le contrôle de la position et de la vitesse des deux axes du tour selon l'interprétation des informations présentes dans le fichier STEP-NC.

Les figures suivantes illustrent certaines des phases d'usinage réalisées :

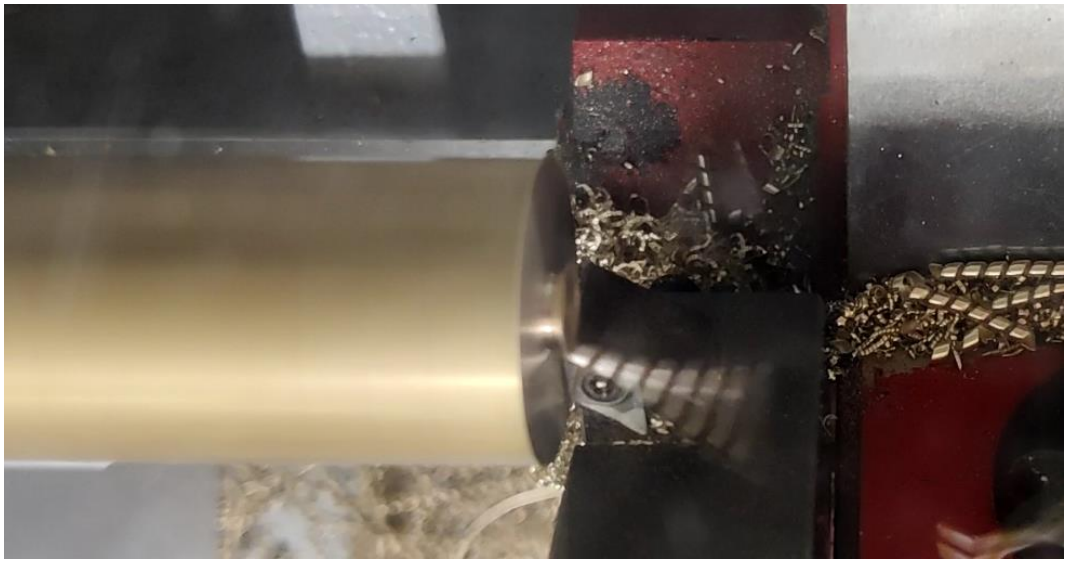


Figure 5.26 Dressage d'ébauche correspondant au *Workingstep* : Rough end face

Figure 5.27 Contournage d'ébauche correspondant au *Workingstep* : Rough contouring

Figure 5.28 Contournage d'ébauche correspondant au Workingstep : Rough contouring

On remarque un problème lors de la phase d'usinage du contournage de finition. En effet, lors de l'usinage du chanfrein, les mouvements de la machine n'ont pas suivi les mouvements souhaités et l'usinage de la pièce a dû être arrêté. Le problème étant que le déplacement de l'axe X et Z doit être effectué simultanément. Dans le logiciel, cela se traduit par deux threads qui traitent les informations de déplacements des deux axes. C'est alors le système d'exploitation du RPi qui gère l'allocation des temps d'exécution de chaque thread. On remarque qu'ici le temps d'exécution est mal géré, puisqu'il privilégie l'axe Z à l'axe X dans l'exécution, ce qui implique un déplacement plus important sur l'axe Z, alors que l'axe X doit attendre les moments où les instructions de l'axe Z ont été traitées pour être exécutées. Bien que l'outil atteigne le point d'arrivée souhaiter, le mouvement pour y arriver ne correspond pas au mouvement souhaité. L'allocation manuelle de l'ordre d'exécution permettrait de résoudre le problème, mais est complexe à mettre en œuvre et demande des compétences avancées en programmation. Une deuxième possibilité est d'exécuter le programme sur plusieurs cœurs du RPi puisqu'il en possède quatre. Cette solution permettra en plus d'améliorer les performances du logiciel tout en permettant l'exécution simultanée des mouvements de l'axe X et Z sur deux cœurs séparés. Cette méthode de programmation est elle aussi complexe et demande des compétences avancées en programmation.

La Figure 5.29 présente en vert le déplacement souhaité pour l'opération de contournage et en rouge le déplacement réellement effectué.

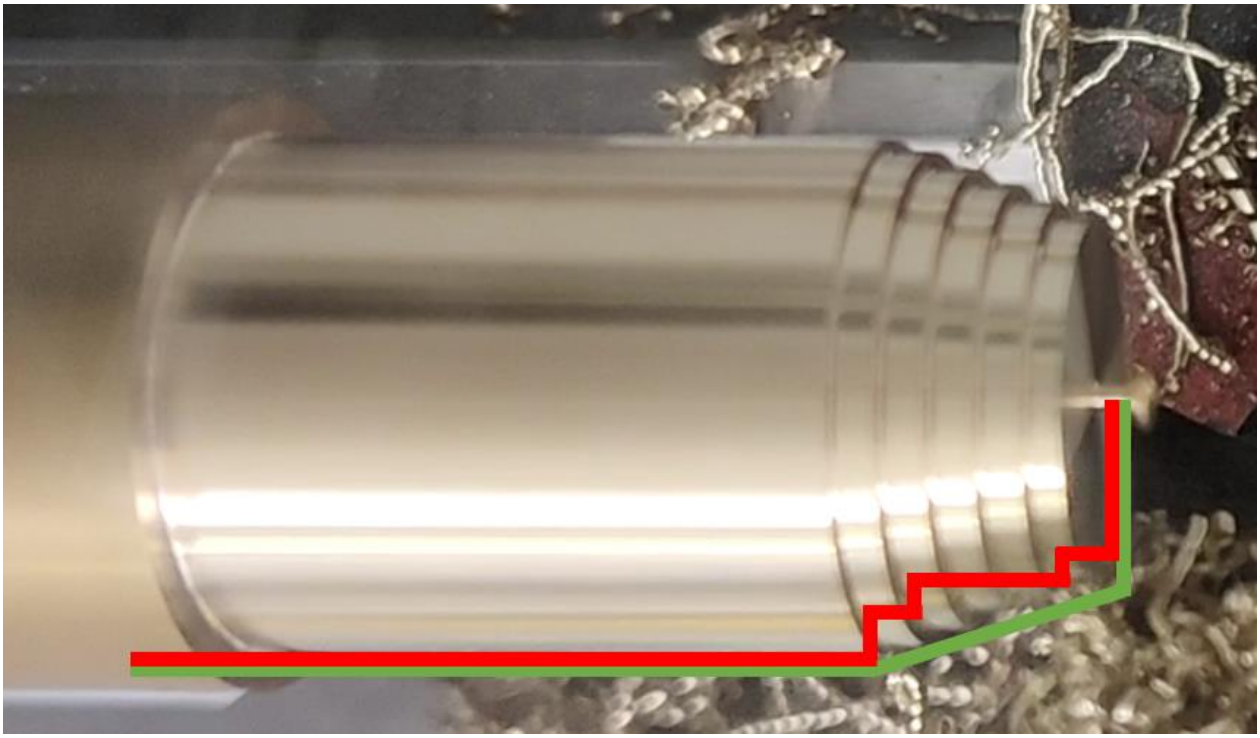


Figure 5.29 Déplacement souhaité (en vert) et déplacement réellement effectué (en rouge) lors de l'opération de contournage

Une prise de mesure a été effectuée de la même manière que pour la pièce usinée à partir du contrôleur Code-G (au niveau de l'épaulement de 20 mm, du diamètre de 10 mm et les longueurs de 40 mm de l'épaulement et de 12,5 mm du chanfrein). On retrouve la même précision que pour la pièce usinée à partir du Code-G, c'est-à-dire au dixième de millimètre.

5.3.3 Conclusion des expérimentations

Les expérimentations ont permis de valider partiellement le fonctionnement du contrôleur STEP-NC développé.

Il est capable d'interpréter correctement les informations présentes dans le fichier STEP-NC, en générant le parcours d'outils et son interpolation dont la cohérence peut être validée par la visualisation de la simulation. Ensuite l'animation de la simulation nous a permis aussi de vérifier la cohérence des vitesses d'usinage.

Enfin, le tour EMCO PC TURN 55 a été modifié afin de valider le fonctionnement du matériel. Cette étape est partiellement validée. Sur les quatre *WORKINGSTEPS*, uniquement trois ont été correctement exécutés. Le dernier n'a pas pu être entièrement effectué à la suite du problème

d’exécution des déplacements des axes. Ce problème devra être réglé de manière logicielle. La viabilité du matériel utilisé peut alors être validée puisqu’il permet la bonne exécution des différents organes de la MOCN, en se connectant facilement à ces derniers.

CHAPITRE 6 CONCLUSION ET RECOMMANDATIONS

Ce chapitre présente dans un premier temps les contributions de ce projet de recherche, puis dans un deuxième temps, nous aborderons ses limitations pour finir sur les opportunités de recherches qui résultent de ce projet.

La revue de littérature sur les contrôleurs STEP-NC actuellement existants a mis en avant la complexité, le coût et l'encombrement de ces derniers. C'est pourquoi nous avons décidé d'axer notre recherche sur le développement d'un contrôleur STEP-NC interprété qui soit peu coûteux, facilement reproductible et compact.

Pour mener à bien ce projet, nous avons défini une méthodologie basée sur les projets de contrôleurs STEP-NC présent dans la littérature. Après avoir effectué une revue systématique de littérature dans le chapitre 2, nous avons recensé et spécifié les requis pour notre contrôleur, afin qu'il réponde aux exigences fixées ce qui nous a permis de répondre au SO-1 : Recenser et spécifier les requis pour le contrôleur afin qu'il réponde aux exigences fixées : peu coûteux, adaptable, évolutif et compact. Ensuite, nous avons réalisé les choix technologiques dans le chapitre 4 qui permettent de répondre à notre problématique, d'un point de vue matériel et d'un point de vue du langage de programmation. Notre choix de matériel s'est porté sur un RPi qui respecte nos critères de coût, d'encombrement et d'interopérabilité. Le logiciel a été codé en C++, qui est un langage déjà très utilisé dans le domaine et répond aux exigences de rapidité et d'ouverture qui était demandée. Le développement logiciel a ensuite été présenté. Ces deux parties permettent de répondre au SO-2 : Développer le contrôleur d'un point de vue du logiciel et d'un point de vue de l'architecture matériel. Le chapitre 5 a ensuite permis de valider partiellement notre prototype par l'usinage d'une pièce présente dans le standard sur un tour à commande numérique dont l'ancien contrôleur Code-G a été remplacé par notre prototype. Ce dernier chapitre nous a permis de répondre au SO-3 : Valider le contrôleur sur un cas réel bien qu'il y ait certains problèmes que nous avons détaillés dans le chapitre précédent, notre objectif général de recherche peut être validé en montrant qu'il est possible de développer un contrôleur STEP-NC interprété, peu cher (environ un dixième du prix des contrôleurs précédemment développé), compact et facilement reproductible.

Pendant notre développement, un article de Dharmawardhana, Mahanama et al. (2021) a été publié et traite du développement d'un contrôleur basé sur une RPi ainsi qu'une carte microcontrôleur

Arduino Uno. Ce travail nous conforte sur le choix d'utiliser une RPi comme support matériel. En revanche, le contrôleur développé ne présente pas de systèmes de visualisation du parcours d'outils. De plus, il a été installé sur une MOCN développée spécialement pour le projet et non sur une MOCN commerciale existante. Enfin, son architecture est légèrement plus complexe et coûteuse puisqu'elle utilise en plus d'un RPi, une carte microcontrôleur Arduino Uno afin de réaliser, entre autres, le contrôle des moteurs.

Notre projet a donc apporté une contribution scientifique, en proposant un prototype de contrôleur STEP-NC interprété plus compact, moins cher et moins complexe que ceux précédemment développés dans la littérature.

Bien que notre prototype soit fonctionnel, il existe de nombreuses limitations à notre projet :

- Dans un premier, il sera nécessaire de continuer son développement afin de pouvoir envisager son utilisation sur d'autres pièces. En effet, comme nous l'avons vu dans la partie expérimentation, des problèmes subsistent encore quant à l'exécution des mouvements des axes. Des pistes d'amélioration ont été citées afin de corriger ces problèmes.
- Notre projet s'est concentré sur l'interprétation des informations présentes dans le fichier STEP-NC et son exécution et non sur l'optimisation du parcours d'usinage et la précision de l'usinage. De nombreuses avancées doivent être faites dans cet axe afin de rendre le contrôleur viable pour un usage commercial. On pourra par exemple optimiser le déplacement lors des transitions entre les phases d'usinage, ou encore les vitesses de déplacement entre les phases de coupes.
- Le langage STEP-NC étant très riche et couvrant une large gamme d'opérations, de features, d'outils, et de fonctionnalités, à travers plusieurs procédés de fabrication, uniquement une partie des entités a été implémentée.
- De plus, c'est uniquement le procédé de tournage qui a été implémenté dans un souci de simplification. Il faudrait continuer le développement pour ajouter d'autres procédés comme le fraisage par exemple qui serait compatible sans modifications du matériel utilisé.
- Pour assurer un usage commercial, de nombreuses autres fonctionnalités liées à l'interface avec l'utilisateur ou à la sécurité devraient être implémentées.

Ces limitations peuvent en revanche être levées en poursuivant le développement du projet, le cadre de travail étant posé, l'implémentation de nouvelles entités et de nouvelles fonctionnalités sont alors aisées.

La plateforme matérielle et logicielle choisie étant ouverte et interopérable, il est possible d'ajouter de nombreuses fonctionnalités. On peut citer par exemple :

- L'ajout de fonctionnalité logiciel (possibilité de modification du fichier STEP-NC, de stocker des informations mesurées dans le fichier STEP-NC par exemple)
- L'ajout de capteurs pour analyser l'usinage comme un capteur de vibration par exemple. Cela permettra de prendre des mesures pendant l'usinage et d'avoir des données qui peuvent être analysées pour améliorer le programme, voire d'adapter le programme en temps réel en fonction de l'acquisition des données (Closed Loop Manufacturing).
- L'ajout d'un palpeur permettant la mesure automatique du brut d'usinage afin d'adapter la dimension du brut automatiquement dans le logiciel.
- L'ajout d'une caméra afin de pouvoir suivre le déroulement de l'usinage depuis n'importe où.

En conclusion, un prototype de contrôleur STEP-NC peu cher, compact, facilement reproductible et interopérable a été développé et ouvre la voie à de nombreux projets d'amélioration dans le but de démocratiser l'usage de STEP-NC afin d'optimiser la chaine numérique d'industrialisation.

RÉFÉRENCES

Brecher, C., Verl, A., Lechler, A., & Servos, M. (2010). Open control systems: state of the art. *Production Engineering, 4*(2-3), 247-254. https://doi.org/10.1007/s11740-010-0218-5

Calabrese, F., & Celentano, G. (25-28 Sept. 2007). *Design and realization of a STEP-NC compliant CNC embedded controller.* IEEE Conference on Emerging Technologies and Factory Automation, UNITED STATES (p. 1010-1017). https://doi.org/10.1109/EFTA.2007.4416894

Cha, J. M., Suh, S. H., Hascoet, J. Y., & Stroud, I. (2016). A roadmap for implementing new manufacturing technology based on STEP-NC. *Journal of Intelligent Manufacturing, 27*(5), 959-973. https://doi.org/10.1007/s10845-014-0927-2

Choi, I., Suh, S. H., Kim, K., Song, M., Jang, M., & Lee, B. E. (2006). Development process and data management of TurnSTEP: a STEP-compliant CNC system for turning. *International Journal of Computer Integrated Manufacturing, 19*(6), 546-558. https://doi.org/10.1080/09511920600622072

Danjou, C. (2015). *Ingénierie de la chaîne numérique d'industrialisation : proposition d'un modèle d'interopérabilité pour la conception-fabrication intégrées* [thèse doctorale, Université de Technologie de Compiègne].

Danjou, C., Le Duigou, J., & Eynard, B. (2016). Closed-loop Manufacturing, a STEP-NC Process for Data Feedback: A Case Study. *Procedia CIRP, 41*, 852-857. https://doi.org/10.1016/j.procir.2015.12.034

Dharmawardhana, M., Oancea, G., & Ratnaweera, A. (2018). A review of STEP-NC compliant CNC systems and possibilities of closed loop manufacturing. *IOP Conference Series: Materials Science and Engineering, 399*. https://doi.org/10.1088/1757-899x/399/1/012014

Dharmawardhana, M., Ratnaweera, A., & Oancea, G. (2021). STEP-NC Compliant Intelligent CNC Milling Machine with an Open Architecture Controller. *Applied Sciences, 11*(13). https://doi.org/10.3390/app11136223

Elias, D. M., Yusof, Y., & Minhat, M. (Dec 02-04 2013). *CNC Machine System Via STEP-NC Data Model and Lab VIEW Platform for Milling Operation.* IEEE Conference on Open Systems (ICOS), Kuching, MALAYSIA (p. 27-+).

Facts&Factors. (2020). *Global CNC Machine Market Size Expected to Reach USD 115 Billion by 2026, Increasing Growth of CNC Automation Propel the Global Market*. Globenewswire. https://www.globenewswire.com/news-release/2020/11/24/2132555/0/en/Global-CNC-Machine-Market-Size-Expected-to-Reach-USD-115-Billion-by-2026-Increasing-Growth-of-CNC-Automation-Propel-the-Global-Market-Facts-Factors.html

Grigoriev, S. N., & Martinov, G. M. (2016). An ARM-based Multi-channel CNC Solution for Multi-tasking Turning and Milling Machines. *Procedia CIRP, 46*, 525-528. https://doi.org/10.1016/j.procir.2016.04.036

Hamilton, K., Hascoet, J.-Y., & Rauch, M. (2014). Implementing STEP-NC: Exploring Possibilities for the Future of Advanced Manufacturing. Dans *Modern Mechanical Engineering* (p. 199-239). ISBN : 978-3-642-45175-1

Hascoet, J.-Y., & Rauch, M. (2016). Enabling Advanced CNC Programming with openNC Controllers for HSM Machines Tools. *High Speed Machining, 2*(1). https://doi.org/10.1515/hsm-2016-0001

Hu, P., Fu, H., Fu, Y., & Han, D. (1-3 Aug. 2016). *STEP-NC interpreter for intelligent and open CNC.* International Symposium on Flexible Automation (ISFA), Cleveland, Ohio, U.S.A (p. 41-44). https://doi.org/10.1109/ISFA.2016.7790133

Hu, P., Han, Z. Y., Fu, H. Y., & Han, D. D. (2016). Architecture and implementation of closed-loop machining system based on open STEP-NC controller. *International Journal of Advanced Manufacturing Technology, 83*(5-8), 1361-1375. https://doi.org/10.1007/s00170-015-7631-z

International Organization for Standardization. (1982). *Commande numérique des machines — Format de programme et définition des mots adresses* (6983-1:1982).

International Organization for Standardization. (1994). *Systèmes d'automatisation industrielle et intégration — Représentation et échange de données de produits* (10303:1994).

International Organization for Standardization. (2003a). *General Process Data* (14649-10:2003).

International Organization for Standardization. (2003b). *Overview and fundamental principles* (14649-1:2003).

International Organization for Standardization. (2003c). *Process date for turning* (14649-12:2003).

International Organization for Standardization. (2007). *Systèmes d'automatisation industrielle et intégration — Représentation et échange de données de produits — Partie 238: Protocole d'application: Modèle d'application interprété pour des contrôleurs numériques informatisés* (10303-238:2007).

Laguionie, R., Rauch, M., Hascoët, J. Y., & Suh, S. H. (2011). An eXtended Manufacturing Integrated System for feature-based manufacturing with STEP-NC. *International Journal of Computer Integrated Manufacturing, 24*(9), 785-799. https://doi.org/10.1080/0951192x.2011.592992

Lan, H., Liu, R., & Zhang, C. (2008). A multi-agent-based intelligent STEP-NC controller for CNC machine tools. *International Journal of Production Research, 46*(14), 3887-3907. https://doi.org/10.1080/00207540701213494

Latif, K., Adam, A., Yusof, Y., & Kadir, A. Z. A. (2021). A review of G code, STEP, STEP-NC, and open architecture control technologies based embedded CNC systems. *The International Journal of Advanced Manufacturing Technology*. https://doi.org/10.1007/s00170-021-06741-z

Latif, K., & Yusof, Y. (Sep 16-18 2015). *New Method for the Development of Sustainable STEP-Compliant Open CNC System.* 13th Global Conference on Sustainable Manufacturing - Decoupling Growth from Resource Use, Binh Dong New City, VIETNAM (vol. 40, p. 230-235). https://doi.org/10.1016/j.procir.2016.01.110

Latif, K., Yusof, Y., Latif, Q., Jamaludin, S. N. S., & Zaki, W. M. (2017). New Open CNC Machine Motion Control System for ISO 14649 and ISO 6983. *Advanced Science Letters, 23*(6), 5024-5028. Proceedings Paper. https://doi.org/10.1166/asl.2017.7302

Latif, K., Yusof, Y., Nassehi, A., & Alias Imran Latif, Q. B. (2016). Development of a feature-based open soft-CNC system. *The International Journal of Advanced Manufacturing Technology, 89*(1-4), 1013-1024. https://doi.org/10.1007/s00170-016-9124-0

Lee, W., & Bang, Y.-B. (2010). Design and implementation of an ISO14649-compliant CNC milling machine. *International Journal of Production Research, 41*(13), 3007-3017. https://doi.org/10.1080/00207540310001064 34

Lee, W., Bang, Y. B., Ryou, M. S., Kwon, W. H., & Jee, H. S. (2006). Development of a PC-based milling machine operated by STEP-NC in XML format. *International Journal of Computer Integrated Manufacturing, 19*(6), 593-602. https://doi.org/10.1080/09511920600623674

Li, X., & Liang, H. B. (Sep 16-18 2011). *Development of a STEP-NC controller for 5-axis machining of NURBS surfaces.* International Conference on Advanced Design and Manufacturing Engineering (ADME 2011), Guangzhou, PEOPLES R CHINA (vol. 317-319, p. 1940-+). https://doi.org/10.4028/www.scientific.net/AMR.317-319.1940

Li, X., Lin, L., & Liang, H. B. (Aug 09-12 2009). *An Open CNC for Machining NURBS Surfaces Based on STEP-NC.* IEEE International Conference on Mechatronics and Automation, Changchum, PEOPLES R CHINA (p. 1268-+). https://doi.org/10.1109/icma.2009.5246645

Liang, H. B., & Li, X. (2013). Five-axis STEP-NC controller for machining of surfaces. *International Journal of Advanced Manufacturing Technology, 68*(9-12), 2791-2800. Article. https://doi.org/10.1007/s00170-013-4871-7

Mohamed, S. B., Jameel, A., & Minhat, M. (Dec 04-06 2013). *A Review on Intelligence STEP-NC Data Model and Function Blocks CNC Machining Protocol.* 1st International Materials, Industrial, and Manufacturing Engineering Conference, Johor Bahru, MALAYSIA (vol. 845, p. 779-785). https://doi.org/10.4028/www.scientific.net/AMR.845.779

Monkova, K., Monka, P., Ungureanu, M., Ungureanu, N., Gusak, O., & Edl, M. (2019). Data network related to an object manufacturing inside of exerted Intelligent System. *EAI Endorsed Transactions on Industrial Networks and Intelligent Systems, 6*(18). https://doi.org/10.4108/eai.28-3-2019.157123

Mueller, P., & Hyu, Y. T. (2001). *ESPRIT Project EP 29708 STEP-Compliant Data Interface for Numerical Controls (STEP-NC).* STEP-NC consortium.

Othman, M. A., Minhat, M., & Jamaludin, Z. (Aug 01-02 2017). *An overview on STEP-NC compliant controller development.* 4th International Conference on Mechanical Engineering Research (ICMER), Kuantan, MALAYSIA (vol. 257). https://doi.org/10.1088/1757-899x/257/1/012048

Po, H., Hongya, F., Zhenyu, H., & Dedong, H. (8-11 July 2014). *A closed-loop and self-learning STEP-NC machining system.* IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Besançon, FRANCE (p. 1598-1603). https://doi.org/10.1109/AIM.2014.6878312

Rauch, M., & Hascoet, J. Y. (Jul 02-04 2012). *A manufacturing system for advanced multi-process manufacturing based on step-nc.* 11th ASME Biennial Conference on Engineering Systems Design and Analysis, (ESDA 2012), Nantes, FRANCE (p. 71-79).

Rauch, M., Hascoët, J. Y., Simoes, V., & Hamilton, K. (2014). Advanced programming of machine tools: interests of an open CNC controller within a STEP-NC environment. *International Journal of Machining and Machinability of Materials, 15*(1/2). https://doi.org/10.1504/ijmmm.2014.059184

Rauch, M., Laguionie, R., Hascoet, J.-Y., & Suh, S.-H. (2012). An advanced STEP-NC controller for intelligent machining processes. *Robotics and Computer-Integrated Manufacturing, 28*(3), 375-384. https://doi.org/10.1016/j.rcim.2011.11.001

Rauch, M., Laguionie, R., Hascoet, J., & Xu, X. (2009). Enhancing CNC Manufacturing Interoperability with STEP-NC. *Journal of Machine Engineering*. https://doi.org/hal-00501489f

Sang, Z. Q., & Xu, X. (Nov 15-21 2013). *development of a smart computer numerical control system.* ASME International Mechanical Engineering Congress and Exposition (IMECE2013), San Diego, CA.

Suh, S.-H., Cho, J.-H., & Hong, H.-D. (2002). On the architecture of intelligent STEP-compliant CNC. *International Journal of Computer Integrated Manufacturing, 15*(2), 168-177. https://doi.org/10.1080/09511920110056541

Suh, S.-H., Chung, D.-H., Lee, B.-E., Shin, S., Choi, I., & Kim, K.-M. (2006). STEP-compliant CNC system for turning: Data model, architecture, and implementation. *Computer-Aided Design, 38*(6), 677-688. https://doi.org/10.1016/j.cad.2006.02.006

Suh, S.-H., Kang, S., Chung, D.-H., & Stroud, I. (2008). *Theory and Design of CNC Systems*. Spinger. ISBN : 978-1-84800-335-4

Suh, S. H., & Cheon, S. U. (2002). A Framework for an Intelligent CNC and Data Model. *The International Journal of Advanced Manufacturing Technology, 19*(10), 727-735. https://doi.org/10.1007/s001700200083

Suh, S. H., Chung, D. H., Lee, B. E., Cho, J. H., Cheon, S. U., Hong, H. D., & Lee, H. S. (2002). Developing an integrated STEP-compliant CNC prototype. *Journal of Manufacturing Systems, 21*(5), 350-362. https://doi.org/10.1016/s0278-6125(02)80034-6

Suh, S. H., Lee, B. E., Chung, D. H., & Cheon, S. U. (2003). Architecture and implementation of a shop-floor programming system for STEP-compliant CNC. *Computer-Aided Design, 35*(12), 1069-1083. https://doi.org/10.1016/s0010-4485(02)00179-3

Suk-Hwan Suh, S.-U. C. (2005). *Intelligent STEP-NC Controller*. U. S. P. A. Publication. Patent No : 10/506,685

Tao, L., Yongzhang, W., & Hongya, F. (21-23 June 2006 2006). *The Open Architecture CNC System HITCNC Based on STEP-NC.* 2006 6th World Congress on Intelligent Control and Automation (vol. 2, p. 7983-7987). https://doi.org/10.1109/WCICA.2006.1713526

Wang, G. X., Shu, Q. L., Wang, J., & Wang, W. S. (2012). Open Architecture of CNC System Based on STEP-NC Data Model. *Applied Mechanics and Materials, 220-223*, 422-425. https://doi.org/10.4028/www.scientific.net/AMM.220-223.422

Wang, K., Liu, R. L., Xu, X., Zhang, C. R., & Yang, L. (2012). A STEP-compliant computer numerical control based on real-time Ethernet for circuit board milling. *International*

Journal of Computer Integrated Manufacturing, 25(12), 1151-1164. Article. https://doi.org/10.1080/0951192x.2012.684720

Xu, X. W. (2006). Realization of STEP-NC enabled machining. *Robotics and Computer-Integrated Manufacturing, 22*(2), 144-153. https://doi.org/10.1016/j.rcim.2005.02.009

Xu, X. W., Wang, H., Mao, J., Newman, S. T., Kramer, T. R., Proctor, F. M., & Michaloski, J. L. (2005). STEP-compliant NC research: the search for intelligent CAD/CAPP/CAM/CNC integration. *International Journal of Production Research, 43*(17), 3703-3743. https://doi.org/10.1080/00207540500137530

Yusof, Y., & Latif, K. (2015a). New Interpretation Module for Open Architecture Control Based CNC Systems. *Procedia CIRP, 26*, 729-734. https://doi.org/10.1016/j.procir.2014.07.051

Yusof, Y., & Latif, K. (2015b). New technique for the interpretation of ISO 14649 and 6983 based on open CNC technology. *International Journal of Computer Integrated Manufacturing*, 1-13. https://doi.org/10.1080/0951192x.2015.1030698

Zhang, Y., Zeng, Q. F., Mu, G. D., Yang, Y. F., Yan, Y. T., Song, W. L., & Gong, Y. D. (2018). A Design for a Novel Open, Intelligent and Integrated CNC System Based on ISO 10303-238 and PMAC. *Tehnicki Vjesnik-Technical Gazette, 25*(2), 470-478. https://doi.org/10.17559/tv-20170419111243

Zhao, Y. F., Habeeb, S., & Xu, X. (2008). Research into integrated design and manufacturing based on STEP. *The International Journal of Advanced Manufacturing Technology*, *44*(5-6), 606-624. https://doi.org/10.1007/s00170-008-1841-6

ANNEXE A PROGRAMME STEP-NC DE LA PIÉCE POUR TOURNAGE, ISSUE DE LA NORME ISO14649 PARTIE 12

```
/* ***** Manufacturing features ***** */
#10=REVOLVED_FLAT('END FACE',#1,(#20,#21),#70,#80,0.000,#91);
#11=OUTER_DIAMETER('CONE',#1,(#22,#23),#76,#83,#93,#95);
#12=OUTER_DIAMETER('CYLINDER',#1,(#22,#23),#78,#72,#74,$);
/* ***** Turning operations ***** */
#20=FACING_ROUGH($,$,'ROUGH END FACE',$,$,#100,#41,#40,#52,#53,#50,0.500);
#21=FACING_FINISH($,$,'FINISH END FACE',$,$,#110,#42,#40,#52,#53,#51,0.000);
#22=CONTOURING_ROUGH($,$,'ROUGH CONTOUR',$,$,#100,#43,#40,#56,#56,#54,0.500);
#23=CONTOURING_FINISH($,$,'FINISH CONTOUR',$,$,#110,#44,#40,#56,#56,#55,0.000);
/* ***** Project ***** */
#29=PROJECT('TURNING EXAMPLE 1',#30,(#1),$,$,$);
#30=WORKPLAN('MAIN WORKPLAN',(#31,#32,#33,#34),$,#37,$);
#31=MACHINING_WORKINGSTEP('WS ROUGH END FACE',#63,#10,#20,$);
#32=MACHINING_WORKINGSTEP('WS FINISH END FACE',#63,#10,#21,$);
#33=TURNING_WORKINGSTEP('WS ROUGH CONTOUR',#63,(#11,#12),#22,$);
#34=TURNING_WORKINGSTEP('WS FINISH CONTOUR',#63,(#11,#12),#23,$);
#37=SETUP('SETUP FOR TURNING EXAMPLE 1',$,#63,(#38));
#38=WORKPIECE_SETUP(#1,#64,$,$,());
/* ***** Functions / Technology ***** */
#40=TURNING_MACHINE_FUNCTIONS(.T.,$,$,(),.F.,$,$,(),$,$,$);
#41=TURNING_TECHNOLOGY($,.TCP.,#45,0.300,.F.,.F.,.F.,$);
#42=TURNING_TECHNOLOGY($,.TCP.,#46,0.200,.F.,.F.,.F.,$);
#43=TURNING_TECHNOLOGY($,.TCP.,#47,0.300,.F.,.F.,.F.,$);
#44=TURNING_TECHNOLOGY($,.TCP.,#48,0.200,.F.,.F.,.F.,$);
#45=CONST_SPINDLE_SPEED(5.000); #46=CONST_CUTTING_SPEED(2.500,10.000);
#47=CONST_CUTTING_SPEED(2.500,10.000);
#48=CONST_CUTTING_SPEED(2.200,10.000);
***** Strategies ***** */
#50=UNIDIRECTIONAL_TURNING($,$,(3.000),$,$,#82,$,$,2.000,$,$);
#51=UNIDIRECTIONAL_TURNING($,$,(0.500),$,$,#82,$,$,2.000,$,$);
#52=AP_RETRACT_TANGENT($,60.000); #53=AP_RETRACT_ANGLE($,100.000,2.000);
#54=UNIDIRECTIONAL_TURNING($,$,(3.000),$,$,$,$,$,2.000,$,$);
#55=CONTOUR_TURNING($,$,(0.500),$,$,#81,$,$,$,$,$);
#56=AP_RETRACT_ANGLE($,45.000,4.000);
/* ***** Placements / Lengths / Planes ***** */
#63=PLANE('SECURITY PLANE',#68);
#64=AXIS2_PLACEMENT_3D('WORKPIECE',#65,#66,#67);
#65=CARTESIAN_POINT('WORKPIECE: LOCATION',(0.000,0.000,0.000));
#66=DIRECTION('WORKPIECE: AXIS',(0.000,0.000,1.000));
#67=DIRECTION('WORKPIECE: REF_DIRECTION',(1.000,0.000,0.000));
#68=AXIS2_PLACEMENT_3D('SECURITY PLANE',#69,$,$);
#69=CARTESIAN_POINT('SECPLANE: LOCATION',(90.000,0.000,200.000));
```

#70=AXIS2_PLACEMENT_3D('PLACEMENT END FACE',#71,$,$);
#71=CARTESIAN_POINT('END FACE: LOCATION',(0.000,0.000,160.000));
#72=TOLERANCED_LENGTH_MEASURE(80.000,#73);
#73=PLUS_MINUS_VALUE(0.100,0.100,1);
#74=TOLERANCED_LENGTH_MEASURE(110.000,#75);
#75=PLUS_MINUS_VALUE(0.100,0.100,1);
#76=AXIS2_PLACEMENT_3D('PLACEMENT CONE',#77,$,$);
#77=CARTESIAN_POINT('CONE: LOCATION',(0.000,0.000,160.000));
#78=AXIS2_PLACEMENT_3D('PLACEMENT CYLINDER',#79,$,$);
#79=CARTESIAN_POINT('CYLINDER: LOCATION',(0.000,0.000,110.000));
#80=DIRECTION('END FACE: FRONT',(0.000,0.000,-1.000));
#81=DIRECTION('STEPOVER DIRECTION FOR CONTOUR',(1.,0.,0.));
#82=DIRECTION('FACING DIRECTION',(-1.000,0.000,0.000));
#83=TOLERANCED_LENGTH_MEASURE(40.000,#90);
#89=PLUS_MINUS_VALUE(0.000,0.200,1);
#91=LINEAR_PROFILE($,#92); #92=NUMERIC_PARAMETER('LINEAR PROFILE
LENGTH',20.000,'mm'); #93=TOLERANCED_LENGTH_MEASURE(50.000,#94);
#94=PLUS_MINUS_VALUE(0.100,0.100,1);
#95=DIAMETER_TAPER(#96); #96=TOLERANCED_LENGTH_MEASURE(80.000,#97);
97=PLUS_MINUS_VALUE(0.100,0.100,1);
/* ***** Tools ***** */
#100=TURNING_MACHINE_TOOL('ROUGHING TOOL',#101,(#102),$,$,$);
#101=GENERAL_TURNING_TOOL(#103,.LEFT.,$,$,$,$);
#102=CUTTING_COMPONENT(50.,#104,$,$,$);
#103=TURNING_TOOL_DIMENSION($,$,$,10.000,$,20.000,$,0.300,$);
#104=MATERIAL('T15K6','CEMENT CARBIDE',(#105));
#105=NUMERIC_PARAMETER('ELASTIC MODULUS',3.E11,'pa');
#110=TURNING_MACHINE_TOOL('FINISHING TOOL',#111,(#112),$,$,$);
#111=GENERAL_TURNING_TOOL(#113,.LEFT.,$,$,$,$);
#112=CUTTING_COMPONENT(50.,#114,$,$,$);
#113=TURNING_TOOL_DIMENSION($,$,$,35.000,$,25.000,$,0.300,$);
#114=MATERIAL('T15K6','CEMENT CARBIDE',(#115));
#115=NUMERIC_PARAMETER('ELASTIC MODULUS',3.E11,'pa'); ENDSEC;
END-ISO-10303-21;

ANNEXE B PROGRAMME D'USINAGE CODE-G DE LA PIECE POUR TOURNAGE ISSUE DE LA NORME ISO14649 PARTIE 12

```
O1000
N1 G71 G00 G40
N2 T0101 M6
N2 G0 G41 X30 Z45
N3 X25.406 S1000 M4
N4 Z43.5
N5 G1 G95 Z41.5 F.5
N6 Z.75
N7 X25.83 Z.962
N8 G0 Z43.5
N9 X24.414
N10 G1 Z41.5
N11 Z.75
N12 X24.838 Z.962
N13 G0 Z43.5
N14 X23.42
N15 G1 Z41.5
N16 Z.75
N17 X23.844 Z.962
N18 G0 Z43.5
N19 X22.426
N20 G1 Z41.5
N21 Z.75
N22 X22.85 Z.962
N23 G0 Z43.5
N24 X21.434
N25 G1 Z41.5
N26 Z27.728
N27 X21.858 Z27.94
N28 G0 Z43.5
N29 X20.44
N30 G1 Z41.5
N31 Z28.969
N32 X20.864 Z29.182
N33 G0 Z43.5
N34 X19.446
N35 G1 Z41.5
N36 Z30.211
N37 X19.87 Z30.423
N38 G0 Z43.5
N39 X18.454
N40 G1 Z41.5
N41 Z31.453
```

N42 X18.878 Z31.665
N43 G0 Z43.5
N44 X17.46
N45 G1 Z41.5
N46 Z32.694
N47 X8.17.884 Z32.907
N48 G0 Z43.5
N49 X16.466
N50 G1 Z41.5
N51 Z33.936
N52 X16.89 Z34.148
N53 G0 Z43.5
N54 X15.44
N55 G1 Z41.5
N56 Z35.178
N57 X15.898 Z35.39
N58 G0 Z43.5
N59 X14.4
N60 G1 Z41.5
N61 Z36.419
N62 X14.904 Z36.632
N63 G0 Z43.5
N64 X13.46
N65 G1 Z41.5
N66 Z37.661
N67 X13.91 Z37.873
N68 G0 Z43.5
N69 X12.494
N70 G1 Z41.5
N71 Z38.903
N72 X12.918 Z39.115
N73 G0 Z43.5
N74 X11.5
N75 G1 Z41.5
N76 Z40.144
N77 X11.924 Z40.357
N78 X11 Z43.5 F.2
N79 Z41.5
N80 Z40.096
N81 X21 Z27.596
N82 Z.5
N83 X26.4
N84 X26.824 Z.712
N84 G0 G40 X30 Z50
N85 M30