Concepts et illustrations de logique programmée surautomates programmables industriels pour la commande de process

Baptiste Trajin, Stéphane Caux, Marcel Grandpierre, Bruno Dagues, Bruno Sareni, Guillaume Gateau

▶ **To cite this version:**

Baptiste Trajin, Stéphane Caux, Marcel Grandpierre, Bruno Dagues, Bruno Sareni, et al.. Concepts et illustrations de logique programmée surautomates programmables industriels pour la commande de process. Colloque sur l'Enseignement des Technologies et des Sciences de l'Information et des Systèmes (CETSIS 2011), Oct 2011, Trois-Rivières, Canada. pp.0. hal-04104823

HAL Id: hal-04104823
https://hal.science/hal-04104823

Submitted on 24 May 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Open Archive Toulouse Archive Ouverte

OATAO is an open access repository that collects the work of Toulouse researchers and makes it freely available over the web where possible

This is an author's version published in: http://oatao.univ-toulouse.fr/21665

To cite this version:

Trajin, Baptiste and Caux, Stéphane and Grandpierre, Marcel and Dagues, Bruno and Sareni, Bruno and Gateau, Guillaume Concepts et illustrations de logique programmée sur automates programmables industriels pour la commande de process. (2011) In: Colloque sur l'Enseignement des Technologies et des Sciences de l'Information et des Systèmes (CETSIS 2011), 23 October 2011 - 26 October 2011 (Trois-Rivières, Canada). (Unpublished)

Any correspondence concerning this service should be sent to the repository administrator: tech-oatao@listes-diff.inp-toulouse.fr

Concepts et illustrations de logique programmée sur automates programmables industriels pour la commande de process

Baptiste Trajin, Stéphane Caux, Marcel Grandpierre, Bruno Dagues, Bruno Sareni, Guillaume Gateau
{prénom.nom}@enseeiht.fr
INPT-ENSEEIHT
Département Génie Electrique et Automatique
2 rue Camichel,
31 071 Toulouse Cedex, France

RESUME : Cet article présente la mise en place d'un réseau d'automates programmables industriels (API) pour la gestion et la commande de maquettes pédagogiques. Ces API sont utilisés comme support de formation à l'automatique et l'informatique industrielle. L'utilisation d'API couplés en réseaux, permet de se rapprocher de situations industrielles réelles, tout en abordant la problématique fondamentale de la logique programmée. L'implantation de commandes de process dans les API, sous différents langages (langage à contacts-LD, GRAFCET-SFC, statement list-SL), permet d'asseoir ou d'aborder différentes notions essentielles de l'informatique industrielle (gestion des mémoires, structure des programmes, synchronisme). Les manipulations associées aux API sont utilisées en première et deuxième années de la formation des élèves ingénieurs du département de formation Génie Electrique et Automatique de l'ENSEEIHT (Ecole Supérieure d'Electrotechnique, d'Electronique, d'Hydraulique et de Télécommunication de Toulouse).

Mots clés : Automates Programmables Industriels, Logique Programmée, GRAFCET implanté.

1 INTRODUCTION

L'enseignement de l'informatique industrielle commence généralement par la présentation de l'aspect théorique (logique combinatoire, séquentielle) et des fonctions logiques (composants, technologie), pour ensuite se concrétiser par des enseignements pratiques dont l'évolution suit également une progression dans la complexité. Dans le département INPT-ENSEEIHT/GEA les 1ere année câblent des fonctions réalisant des cahiers des charges et programment également sur FPGA ce genre de fonction.

Très vite, en fin de 1ere année et en 2e année, les élèves voient les difficultés de la programmation de ces fonctions logiques dans un environnement informatique notamment sur différents matériels et logiciels (Schneider Electric, Leroy Automation, Festo, PL7Pro, Unity, Isagraph…). Cette diversité de matériel et logiciel n'est pas seulement là d'un point de vue technique, mais permet de mettre l'accent sur des concepts plus ou moins bien vérifiés par ces différents matériels et logiciels. Nous pouvons citer les cas traités ici : la gestion d'entrées/sorties déportées, la recherche de stabilité de franchissement, la programmation structurée des règles de synchronisme, la gestion du temps et des temporisations dans un cycle d'automate…

Les supports pédagogiques présentés dans cet article permettent de mettre en place un ensemble de TP répartis sur la première et la deuxième année du cycle de formation des ingénieurs. Cet article se focalise sur le mur d'automate qui est le support des enseignements de l'informatique industrielle programmée sur API en réseau. Le paragraphe 2 décrit le matériel mis en jeu, le paragraphe 3 décrit les implantations mises en place pour valider et illustrer les concepts de réalisation. Le paragraphe 4 présente le lien entre la logique 'de base' et la logique programmée, pour arriver aux réseaux de Pétri et les systèmes multi-tâches temps réels. Un bilan est proposé en conclusion.

2 DISPOSITIF PEDAGOGIQUE

2.1 Réseau d'automates

Le réseau d'automates se compose d'un API *LT160* avec 7 modules d'entrées-sorties (E/S) déportées *Alto* de *Leroy-Automation (fig1)*, de 2 API *Premium P57*, 3 API *Micro TSX3721* et d'un API *M340* de *Schneider-Electric (fig2)*, ainsi que de 3 postes informatiques de programmation. Les logiciels utilisés pour programmer les API sont : *Isagraf* pour le *LT160*, *PL7 Pro* pour les *Premium* et *Micro* et *Unity Pro* pour le *M340*.

fig 1 : E/S déportées et réseau ferroviaire

Les API *Premium*, *LT 160*, *M340* et les E/S dépostées

Alto sont connectés sur un réseau Ethernet TCP-IP afin de dialoguer ensemble et avec les postes informatiques de programmation (voir *figure du réseau* en fin de cet article). Les API *Micro* sont connectés à un sous réseau adressé Fipway, géré par un des *Premium*. Ils peuvent donc également dialoguer avec les postes de programmation via une passerelle informatique gérée par le *Premium*.
De plus, afin de montrer les possibilités d'un réseau, un serveur FTP et DNS, une caméra sur IP et une interface homme machine *Magelis* sont également connectés.

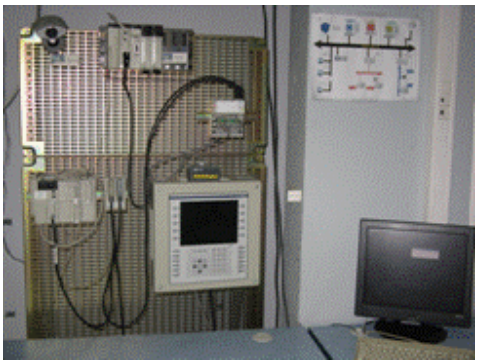

fig 2 : Poste de développement, caméra, interface, et automates premium

3 DESCRIPTION DES OBJECTIFS

3.1 Objectifs des TP

L'un des TP, utilisant une maquette représentative d'un pont transbordeur, est conçu comme une prise en main des API avec une première approche de leurs spécificités de fonctionnement. Il est proposé aux étudiants de première année du cycle de formation. Il permet également d'asseoir certaines notions d'informatique industrielle comme l'exécution séquencée d'un programme ou encore la nécessité de ne pas multi-affecter des variables [1][2].
Deux autres TP, proposés en deuxième année du cycle de formation, portent plus spécifiquement sur la modélisation et la programmation en GRAFCET, en liaison directe avec les autres formes d'enseignement (cours et TD), mais aussi les normes et les contraintes de l'operating system de cet environnement informatique [3][4][5]. Ils permettent d'une part d'implanter quelques structures classiques du GRAFCET répondant à un cahier des charges précis comme le partage de ressources, et d'autre part de tester le fonctionnement des automates par rapport aux règles du formalisme GRAFCET (franchissement, stabilité, synchronisme…) [6][7][8].
Les TP sur les API permettent également d'aborder les notions de réseau avec l'adressage des matériels ou encore les temps de communication entre matériels.

3.2 Logique programmée sur calculateur : Pont transbordeur

Ce TP est proposé aux élèves de première année du cycle d'ingénieur. Il permet, via un API programmé en Ladder (langage à contacts), de commander une partie opérative représentant un pont transbordeur pour le traitement de surfaces. Une photographie de la partie opérative est donnée *fig3*. L'API utilisé est le *M340* avec l'environnement logiciel *Unity Pro*.
Deux modes de fonctionnement sont proposés aux élèves. En guise d'introduction, le premier permet de commander manuellement le pont transbordeur en se servant des boutons poussoirs présents sur la maquette (logique combinatoire programmée). L'intérêt est porté sur l'adressage des variables d'entrée et de sortie sur un automate ainsi que sur la scrutation et l'affectation cyclique des variables.

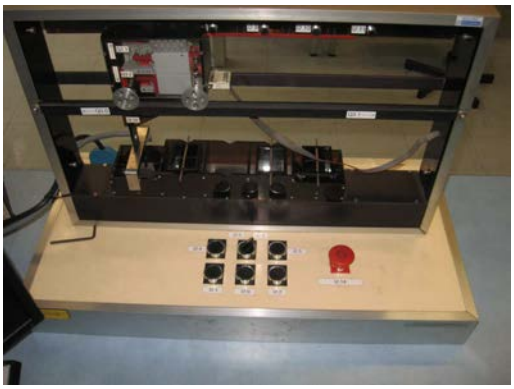

fig 3 : Partie opérative du pont transbordeur

La seconde partie du TP permet d'aborder un mode en cycle 'automatique', dans lequel, par une action fugitive sur un bouton poussoir, le pont transbordeur décrit un cycle de traitement. Dans ce cycle, le pont transbordeur passe plusieurs fois par la même position, entraînant alors un « conflit » dans le séquencement des actions. Ainsi, l'accent est mis sur l'ordonnancement des actions. Pour cela, le concept de transitions et de mémoires (auto-alimentation et logique séquentielle programmée) est mis en avant au travers du schéma de fonctionnement générique présenté en fig4.

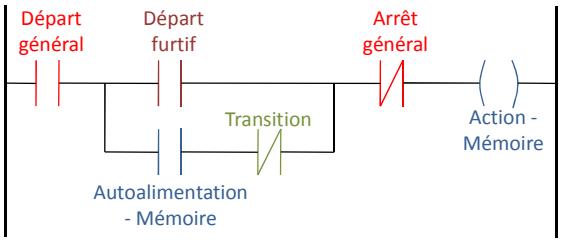

fig 4 : Schéma générique des actions en Ladder pour le séquencement des actions

De plus, le mode manuel et le mode automatique cohabitant dans le même programme, les élèves sont souvent confrontés au problème de la multi-affectation des variables. Dans ce cas, l'utilisation de mémoires internes avec une affectation unique des sorties de l'automate en fin de programme, est introduit. Il est à noté que ce concept est valable pour tous les types de variables (sorties, internes) et rares sont les élèves qui ne commettent pas alors la même erreur, mais sur les variables internes. Les élèves proposent des solutions qui ne font que déplacer le problème et seule l'assimilation des notions de séquencement des lignes d'instruction et de structuration de la programmation permet de résoudre le problème en respectant les phases dans l'ordre d'exécution du processeur :
-initialisation – mémorisation des entrées – calcul des conditions d'activation (dénommées 'transition' dans le GRAFCET) – calcul des réceptivités pour ensuite désactiver ou activer (franchissement) des mémoires internes (Etat/étape) – Puis calculer les actions associées (activation/désactivation des sorties) – mise à jour pour le cycle suivant - rebouclage.
Par ailleurs, les élèves peuvent utiliser une interface homme machine tactile, connectée au réseau principal et permettant de lire ou d'affecter des variables de l'API *M340*. Ils peuvent alors aborder les concepts et les limitations des communications sur un réseau informatique (même si les délais de communication ne sont pas pénalisants ici du fait du micro-réseau réalisé et localisé dans la salle).

3.3 Partage de ressource et contraintes temps réel : Micro-réseau ferroviaire

Ce TP est proposé aux élèves de deuxième année du cycle d'ingénieur. Il permet, via un API programmé en GRAFCET et d'E/S déportées Alto, de commander une partie opérative constituée d'un micro-réseau ferroviaire. Une photographie de la partie opérative est donnée fig5. L'API utilisé est le *LT160* avec l'environnement logiciel *Isagraf*. Le microréseau ferroviaire est constitué de deux voies principales de circulation et de différentes voies de garages, ces voies étant subdivisées en sections, alimentées de manière indépendante et d'aiguillages permettant d'assurer la liaison entre elles. Des capteurs de détection de passage des trains sont disposés tout au long des différents parcours possibles.
Ce TP permet aux élèves de mettre en application les structures fondamentales du GRAFCET déjà étudiées en cours et en TD, telles que les notions de convergence et de divergence ou encore de parallélisme (partage de ressource, sémaphore et exclusion mutuelle). En effet, ils doivent répondre à un cahier des charges dans lequel deux trains tournent en sens inverse sur le réseau ferroviaire, tout en partageant une section commune. Cette section particulière doit être traitée comme une ressource unique qui doit être partagée entre deux demandeurs que sont les trains.

fig 5 : Micro-réseau ferroviaire

Les programmes implantés par les élèves s'inscrivent dans un environnement plus large incluant la gestion des communications entre l'API *LT160* et les E/S déportées *Alto*. Les élèves peuvent alors appréhender de manière plus pratique les spécificités et les limitations des communications sur réseau concernant principalement les voies de communication ainsi que les vitesses de transmission. Cet aspect met en avant la dynamique du procédé et des capteurs au regard du cycle de l'automate. La tension d'alimentation des voies ou le type de locomotive font que le train passe plus ou moins vite d'un secteur à l'autre ou sur un capteur de détection de sa position. Du fait des délais introduit pas le système et les communications, les élèves constatent donc qu'il faut continuer à alimenter un secteur ou faire l'alimentation de deux secteurs consécutifs pour maintenir la continuité du déplacement. Et si le train est 'trop' rapide, l'impossibilité de détecter à coup sur sa position. Ce qui impact la modélisation faite.

3.4 Implantation informatique : Etude du comportement des API

Ce TP est proposé aux élèves de deuxième année du cycle d'ingénieur. Il permet, via un API programmé en GRAFCET d'interpréter et de comprendre les éléments de base du GRAFCET, mais également de tester et de valider le comportement d'un API vis à vis des règles d'évolution du GRAFCET. L'API utilisé est un *Premium* avec l'environnement logiciel *PL7 Pro*, mais aussi un automate de marque *Festo* nécessitant la programmation

'SL' (Statement List) ne respectant pas les normes en vigueur.
Dans une première partie, il est proposé aux élèves d'interpréter et d'implanter un GRAFCET fig 6, ainsi que son évolution au moyen d'équations logiques. Dans ce GRAFCET, il n'y a qu'un seul jeton et la logique séquentielle doit traduire ce comportement.

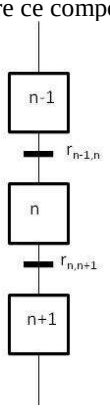

fig 6 : GRAFCET pour le codage d'équations logiques

Nous proposons un codage des activation/désactivation de deux manières différentes (1)(2), avec *Xn* l'état de l'étape *n* et $r_{n-1,n}$ la réceptivité associée à la transition entre les étapes *n-1* et *n*. Ces équations ne sont valables que si les réceptivités ne s'expriment qu'avec des entrées externes et non des états d'étapes.

$$X_n = \left[X_{n-1} \cdot r_{n-1,n} + X_n \right] \cdot \overline{X_{n+1}} \qquad (1)$$

$$X_n = \left[X_{n-1} \cdot r_{n-1,n} + X_n \right] \cdot \overline{r_{n,n}} \qquad (2)$$

Cette interprétation logique ainsi que le fonctionnement de l'API vis à vis du GRAFCET doivent respecter les trois premières règles du GRAFCET concernant : l'initialisation, le franchissement des transitions et l'évolution de la situation du GRAFCET. Ces règles sont bien sur toutes bien implantées dans le cas général, mais l'accent est mis sur des aléas de fonctionnement ou des interprétations possibles du fonctionnement en fonction de la manière dont l'ingénieur fait son programme ou dont l'O.S. de l'automate exécute le programme.

Le TP permet aux étudiants de se focaliser sur les règles 4 et 5 du GRAFCET concernant respectivement les transitions simultanément franchissables et les activations et désactivation simultanées d'étapes (*fig 7*).
En effet, au travers de ces implantations spécifiques, il leur est montré que, la simultanéité parfaite n'existe pas en informatique. Enfin, une étude des algorithmes avec ou sans recherche de stabilité est proposée. L'API est alors testé pour identifier son principe de fonctionnement.

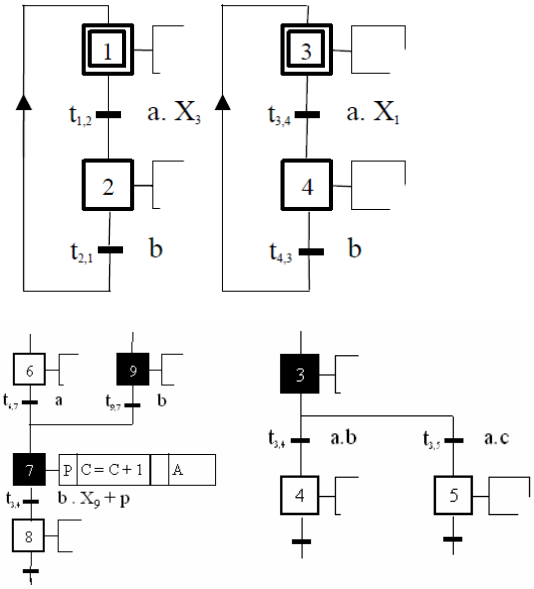

fig 7 : Exemple de GRAFCET simple permettant de tester le respect des règles : en haut - franchissement simultané sur condition d'entrée et état d'étape, en bas à gauche - maintien à 1, en bas à droite - parallélisme interprété

3.5 Gestion du temps et synchronisme : API Festo

Pour illustrer les problèmes liés au concept de la gestion du temps réel dans un cycle informatique (synchronisme), nous proposons aux élèves de programmer en langage 'statement list' pour un automate *Festo* le graphe *fig8*.
A noter ici encore, la nécessité de programmer de manière structurée en programmant des variables internes additionnelles liées aux transitions, aux tests des réceptivités, à l'évolution des étapes, à l'affectation des sorties. Cette méthodologie de programme est rendue obligatoire ici par la multiplicité des branches ET et OU.
De plus, à la fin de cet exemple *fig8*, une transition liée à une temporisation pose le problème de la prise en compte synchronisée du déclanchement et de la fin de cette temporisation.
Les élèves constatent que si leur programmation du démarrage de la temporisation est conditionnée à l'état vrai de l'étape *Xn* (front haut durant plusieurs cycles), la temporisation peut se réinitialiser et ne pas permettre le franchissement voulu.
Les élèves constatent ainsi que la temporisation se déclenche 'un certain nombre de fois' et de temps en temps, la fin de la temporisation se termine après le test de son redémarrage et avant le test de franchissement, c'est le seul cas ou sa fin est alors détectée comme effective. Ceci produisant ainsi un aléa de fonctionnement et un non respect de la temporisation et donc du fonctionnement du cahier des charges normalement modélisé par ce GRAFCET.

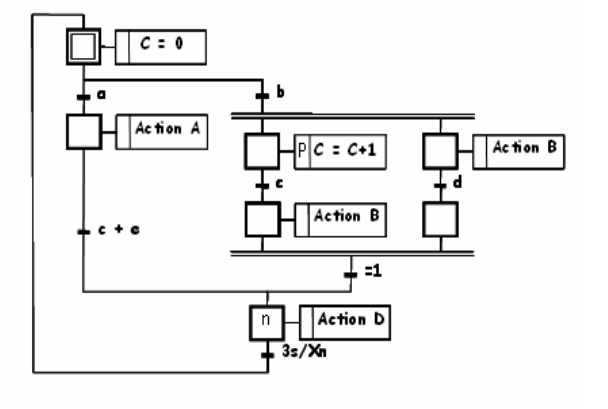

fig 8 : Exemple de GRAFCET *avec gestion du temps*

Si ce test est effectué de manière structuré et si la condition de déclenchement de la temporisation est effectuée une seul fois en utilisant non pas l'étape *Xn*, mais la transition précédente mettant *Xn*=1 la 1ere fois, alors la tempo ne démarre qu'une fois et même si la fin de la temporisation intervient à n'importe quel instant du programme, elle sera prise en compte correctement, au pire, juste au cycle suivant.

4 DIFFICULTES CONCEPTS-REALISATION ET OBJECTIFS ELEVES

Dans ces TP, il est intéressant de souligner les fortes similitudes existantes entre le fonctionnement des API et celui de calculateurs embarqués que les élèves abordent par ailleurs au travers de TP, portant notamment sur la commande temps-réel de machines électriques ou encore sur la programmation de robots mobiles.
Il est intéressant dans ces TP, de ne pas présenter aux étudiants seulement des applications pratiques 'mobiles' voire 'ludique', mais aussi des problèmes de fonctionnement réels, ainsi que des interfaces hommes machines. En ne se limitant pas à de la programmation des API en langage à contacts, statement list ou GRAFCET, nous pensons ainsi éviter le message dans lequel les étudiants peuvent parfois perdre de vue la partie conceptuelle et les contraintes de la gestion de l'automatisation d'un procédé temps réel à l'aide d'un environnement informatique (avec un 'compilateur' et un Operating System dédié).

Ces notions sont alors connectées à l'exécution ligne à ligne d'un programme informatique, avec initialisation des variables au démarrage et la mise à jour pour l'instant suivant (notions non déconnectées des besoins pour l'automatique échantillonnée).
Dans l'utilisation des API, les difficultés des élèves ingénieurs résident fréquemment dans la prise en main des logiciels de programmation ainsi que dans la prise en compte des contraintes liées au fonctionnement cyclique des API, ou encore des contraintes de calcul d'un processeur.

Les élèves réussissent ainsi plutôt bien à faire le lien entre les chronogrammes, la description d'un cahier des charges et de sa modélisation sous forme d'un graphe séquencé, ainsi que la différence entre l'exécution sur papier et celle de la réalité informatique.
Nous laissons les élèves créer les problèmes d'aléas pour voir l'impact sur la partie opérative et ensuite argumenter les concepts théoriques de modélisation, programmation et de structuration temps réel.
Le lien est fait également en fin de 2^{e} année, avec la modélisation par Réseaux de Pétri, généralisant la notion de modélisation en machine d'état et en traitant de manière plus générique la notion d'activation et désactivation de tâches conditionnées à des événements discrets.
Ces cours complémentaires sont vue par ailleurs dans la formation ingénieurs INPT-ENSEEIHT/GEA. Nous détaillons ailleurs les langages synchrones, les concepts liés aux fonctions concurrentes devant respecter un séquencement, mais aussi une approche permettant l'exécution sécurisée et en parallèle ou en temps partagé (OS Temps Réel, multi tâche, tests et blocages, priorité entre taches, sécurité informatique…).

5 CONCLUSION

Cet article présente plusieurs maquettes de travaux pratiques illustrant les notions d'informatique Industrielle dans le département Génie Electrique et Automatique (GEA) de l'INPT-ENSEEIHT.
Ces parties opératives permettent de faire le lien de manière presque ludique avec les concepts inhérents à la logique programmée et le séquencement de tâche dans un environnement informatique temps réel.
L'aspect technique n'est pas occulté, mais nous cherchons à illustrer par ces TP sur API et ces maquettes :
- le respect de la logique combinatoire et séquentielle dans un environnement programmé (auto alimentation, mémoires internes, principe d'unicité)
- le respect d'un graphe par programmation et codage automatique (respect des règles GRAFCET, programmation structurée automatisable)
- la gestion du temps et du synchronisme pour éviter les aléas de fonctionnement (front, action pendant 1cycle, synchronisme).

L'accent est donc mis pour le cycle ingénieur, sur les concepts nécessaires à la bonne compréhension de l'automatisation de systèmes à événement discrets dans un environnement informatique.

Bibliographie

[1] M.BLANCHARD – Comprendre, maîtriser et appliquer le GRAFCET – Cepadues Editions

[2] D.GENDREAU – L'enseignement du GRAFCET : pas si facile ! – Revue Technologies – n° 94, p 11/18 - 1998

[3] NFC-82. *Norme NF C 03-190 : Schémas, diagrammes, tableaux : Diagramme fonctionnel GRAFCET pour la description des systèmes logiques de commande*, 1982.

[4] IEC International Electronical Commission. IEC 848-Preparation of function charts for control systems, 90.

[5] IEC International Electronical Commission. IEC 1131-Standard for programmables controllers part : programming languages, 93

[6] GREPA - LE GRAFCET: de nouveaux concepts - Cepadues Editions

[7] R. DAVID, H. ALLA - Du GRAFCET aux réseaux de Petri - Editions HERMES

[8] P. LE PARC - Apports de la méthodologie synchrone pour la définition et l'utilisation du GRAFCET-Thèse de l'Université de Rennes 1, 94

Remerciements

La mise en œuvre de ces TP a été réalisée notamment grâce à Olivier DAVIAU, Technicien du plateau d'Automatique de l'ENSEEIHT.
Qu'il en soit ici remercié.

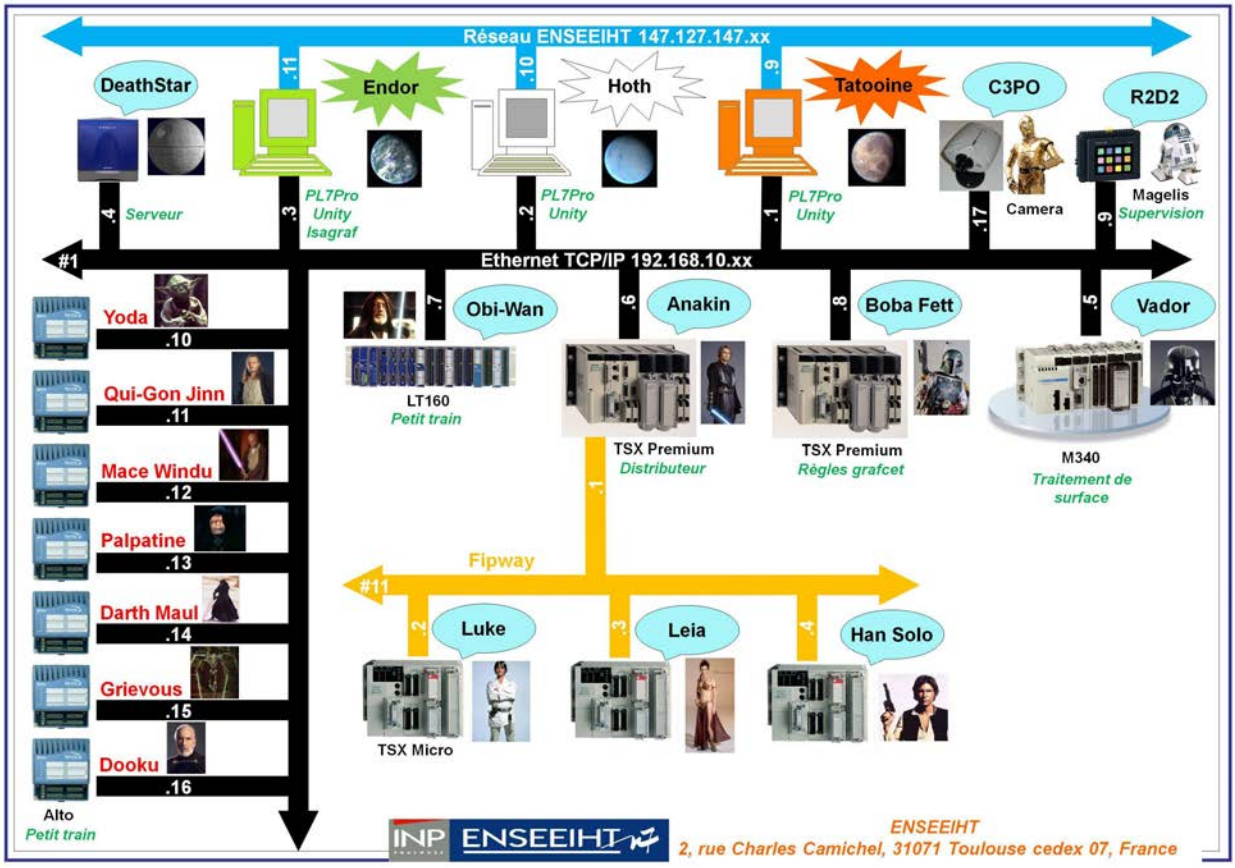

Figure du réseau : Schéma complet du réseau d'automates du département INPT-ENSEEIHT/GEA