
MEMOIRE TECHNIQUE



MARTIN Erwan

APPRENTI RECHERCHE A CESI LINEACT
ETUDIANT A CESI ECOLE D'INGENIEURS

Maître d'apprentissage (CESI LINEACT) :

Fabrice DUVAL

Pilote de promotion (CESI Ecole d'Ingénieurs) :

Sara ZACHARIE

Plan/Sommaire

Table des matières

Plan/Sommaire.....	1
Introduction	2
Présentation de l'entreprise	2
Présentation du sujet et des objectifs.....	8
Résumé.....	9
Présentation de la mission:.....	10
Logigramme.....	10
Présentation détaillée	11
Cas d'usage défini.....	12
Prise en main des fonctionnalités d'un robot	14
Mise en réseau.....	15
Contrôle et coordination des robots.....	16
Présentation des autres packages ROS	21
Installation et guide.....	23
Ordonnancement en utilisant Node-Red.....	24
Bilan et conclusion	28
Bibliographie :	29

Introduction

Présentation de l'entreprise

CESI est un groupe d'enseignement supérieur créé en 1958 par des grandes entreprises industrielles françaises SNECMA, Renault, Télémécanique, Chausson et CEM pour pallier la pénurie d'ingénieurs en France et permettre à leurs techniciens supérieurs et à leurs agents de maîtrise d'accéder à une fonction d'ingénieur.



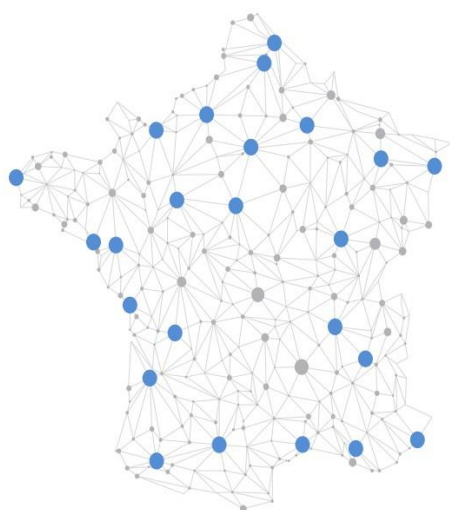
Entreprises industrielles liées à la création de CESI

Réseau de campus d'enseignement supérieur et de formation professionnelle, CESI poursuit sa mission sociétale en permettant à des étudiants, alternants et salariés de devenir acteurs des transformations des entreprises et de la société, grâce à :

- Ses Écoles d'Ingénieurs
- Son École de Formation des Managers
- Son École Supérieure de l'Alternance
- Son activité de Certification.

Pionnier en France dans les méthodes de pédagogie active et tourné vers l'innovation et les technologies, CESI opère dans tous les secteurs d'activités et forme aux métiers et compétences de demain dans l'industrie et les services, le bâtiment et la ville du futur. Il mène des activités de recherche dans son Laboratoire d'Innovation Numérique (LINEACT).

CESI regroupe 25 centres de formation en France et 3 sites à l'international en Espagne, Algérie et Cameroun




60 000
diplômés


22 000
étudiants, alternants
et salariés formés
chaque année


1 000
salariés


2 600
intervenants experts


115 M€
Budget 2019


9 000
entreprises partenaires


94
universités partenaires
partout dans le monde


56
diplômes et titres

Carte centres CESI et chiffres clés

Fruit du rapprochement de l'ei.CESI et de l'exia.CESI, CESI École d'Ingénieurs permet à chaque élève ingénieur de construire un parcours personnalisé dont il est acteur, en cinq ans, sous statut étudiant, en apprentissage ou en formation continue ; 33 options et 25 campus en France.

- 2 400 ingénieurs diplômés chaque année
- 4 spécialités : généraliste, informatique & numérique, BTP et systèmes embarqués - 33 options
- 12 Mastères Spécialisés labellisés par la CGE
- 80 universités internationales partenaires
- 1/4 des ingénieurs diplômés en apprentissage en France sont issus de CESI École d'Ingénieurs.

Première école 100% alternance, CESI École Supérieure de l'Alternance offre aux talents de niveau bac à bac + 5 une formation en alternance sans cesse renouvelée, dont les parcours s'adaptent en permanence aux évolutions des métiers grâce à un réseau de 9 000 entreprises partenaires.

Ses formations reconnues par l'État sont garantes d'une forte employabilité.

- 10 600 étudiants formés
- 94% de réussite
- 88% des diplômés en poste dans les 6 mois suivant la formation
- 3100 diplômés en 2019
- 6 filières d'excellence
- 26 cursus professionnalisants

CESI École de Formation des Managers forme aux principales fonctions de l'entreprise au travers de parcours diplômants et personnalisés. Elle propose 15 cursus diplômants, enregistrés au Répertoire National des Certifications Professionnelles (RNCP), éligibles au CPF, et individualisés selon les compétences de chacun. CESI École de Formation des Managers propose des cursus reconnus par l'État et garants d'une évolution professionnelle.

- 7 000 managers formés chaque année
- 15 cursus diplômants éligibles au CPF
- 97% de réussite - 500 formations
- 6 secteurs d'activités

Fondée au Mans en 1987 par la Chambre de Commerce et de l'Industrie du Mans et de la Sarthe, l'ISMANS a rejoint CESI en 2016 pour former l'ISMANS CESI, lui permettant de mutualiser ses ressources et d'augmenter le rayonnement de l'école.

L'ISMANS CESI forme des ingénieurs en mécanique & calcul de structure et des ingénieurs en génie mécanique & productique. Elle propose une pédagogie active, dans le cadre d'une école en 5 ans, sous statut étudiant ou en alternance en dernière année.

- 1 140 diplômés
- 100% des diplômés en poste 6 mois après la formation
- 17 universités partenaires dans le monde
- 40 intervenants
- 100 entreprises partenaires

Laboratoire d'Innovation Numérique, LINEACT CESI anticipe et accompagne les mutations technologiques des secteurs et des services liés à l'industrie et au BTP. LINEACT CESI est organisé autour de deux thèmes scientifiques interdisciplinaires "Apprendre et Innover" et "Ingénierie et Outils Numériques" et de deux domaines applicatifs que sont l'Industrie et la ville du futur.

En mars 2015, la Chaire d'enseignement et de recherche CESI-Cisco-Vinci Energies intitulée « Industries et services de demain » a été créée. Les problématiques de recherche actuellement traitées portent sur les architectures et le traitement de données distribuées, ou encore les usages de la réalité virtuelle appliqués à nos deux domaines d'application.

Depuis le 1er janvier 2019, LINEACT CESI – Laboratoire d'Innovation Numérique pour les Entreprises et les Apprentissages au service de la Compétitivité des Territoires – formant l'unité de recherche de CESI, a été labellisé Équipe d'Accueil – EA 7527 – pour la qualité des travaux de recherche de ses équipes par le ministère de l'Enseignement Supérieur, de la Recherche et de l'Innovation.

- 60 enseignants-chercheurs, ingénieurs de recherche, techniciens & doctorants
- 3 plateformes de recherche et de transfert

Reconnaisances de CESI

CESI est reconnu par différents organismes de certification pour répondre aux exigences des apprenants et des entreprises



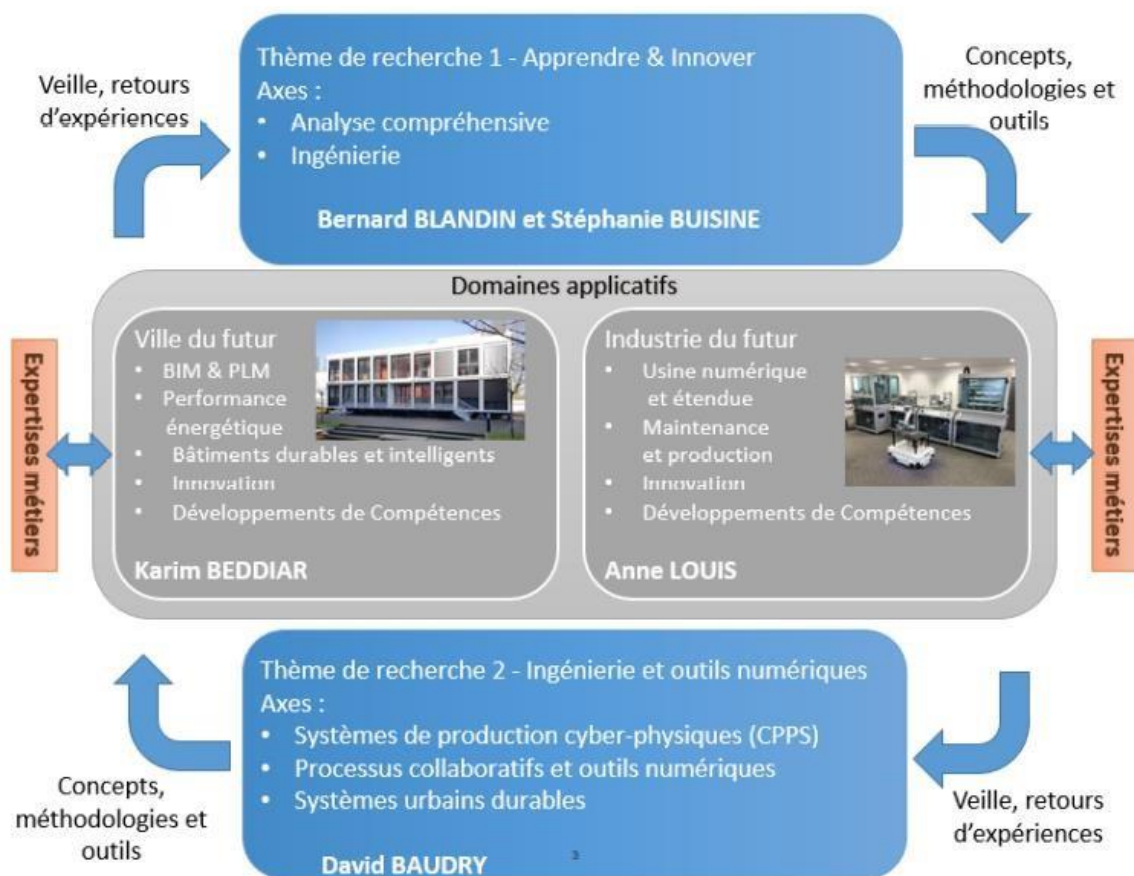
Organismes de reconnaissance CESI



Laboratoire d'Innovation Numérique, LINEACT CESI anticipe et accompagne les mutations technologiques des secteurs et des services liés à l'industrie et au BTP.

LINEACT CESI est organisé autour de deux thèmes scientifiques interdisciplinaires "Apprendre et Innover" et "Ingénierie et Outils Numériques" et de deux domaines applicatifs que sont l'Industrie du Futur et la Ville du Futur. Ces derniers correspondent aux besoins des territoires sur lesquels opèrent les entreprises partenaires de CESI et s'inscrivent également dans les axes de recherche des pays de l'OCDE.

Trois plateformes de recherche et de transfert viennent soutenir l'ensemble de ces travaux : deux sont consacrées à l'usine du futur sur les campus de Rouen et Toulouse, et une au bâtiment du futur sur le campus de Nanterre.



Domaines applicatifs Recherche CESI

Le pôle de recherche a été créé pour permettre à CESI de développer et de maintenir les compétences et l'expertise de son corps enseignant, d'accompagner nos apprenants dans le cadre de leurs formations et notamment de la pédagogie par projets, d'apporter un appui aux formations proposant un parcours dédié à l'innovation et d'être un démonstrateur pour ses journées portes ouvertes et autres manifestations par le biais du démonstrateur Usine du Futur.

Une direction nationale de la Recherche et de l'Innovation, dirigée par Bélahcène MAZARI (Directeur national de la Recherche et de l'Innovation), a été mise en place pour décliner la recherche dans les régions en cohérence avec la stratégie nationale. Les correspondants Recherche, dans chacune des 6 régions du CESI, assurent la coordination et la diffusion des différentes actions vers les équipes régionales. Une cellule de développement et gestion des programmes complète le dispositif en lien avec la direction financière du CESI.

Des FabLab ou ateliers de prototypage complètent le dispositif afin de déployer des situations d'apprentissage instrumenté dans le cadre de pédagogies actives. Ils sont également le lieu d'une activité de recherche et de transfert avec l'appui d'un démonstrateur Usine du futur avec ses outils numériques. Ces FabLabs permettent :

- - La réalisation de prototypes d'objets 3D en plastique,
- - La création de maquettes numériques à partir de numérisation de pièces,
- - La programmation de systèmes embarqués open source pouvant comporter des capteurs et actionneurs.

Recherche et Innovation
 RD : Anne LOUIS (HDR)

David BAUDRY : Directeur de Recherche

Mohamed Amin BENATIA : Enseignant Chercheur

Belgacem BETTAYEB : Enseignant Chercheur (Lille)

Fabrice DUVAL : Enseignant Chercheur (HDR)

Vincent HAVARD : Enseignant Chercheur

Nicolas BRIANT : Ingénieur de Recherche et Innovation

Mourad MESSAADIA : Enseignant Chercheur

M'Hammed SAHNOUN : Enseignant Chercheur

Elodie PILLON : Enseignant Chercheur

Nicolas RAGOT : Enseignant Chercheur (CAEN)

Alexandre COURALLET : Ingénieur de Recherche et Innovation

Roosbeh SADEGHIAN BROUJENY : Enseignant Chercheur (HDF)

Safa BEN AYED : Enseignant Chercheur (Arras)

Léo CHEVALLIER : Ingénieur de Recherche et Innovation(CDD)

Ayman DAMOUN : Ingénieur de Recherche et Innovation(CDD)

Hibat KHYARI : Ingénieur de Recherche et Innovation(CDD)

Louis CHOCHOY : Ingénieur de Recherche (CAP)

Imad HARAOUA : Ingénieur de Recherche et Innovation (CDD)

Rahma LAOUITI : Ingénieur de Recherche et Innovation (CDD)

Erwan MARTIN : Ingénieur de Recherche et Innovation(CAP)

Thibaud BOUNHARD : Ingénieur de Recherche (CDD) LILLE

Antoine ERISAY : Technicien de Recherche (CAP)

Gaëtan SAVARIT /Amal AYADI/Bouchra SAHBANI/Nourddine BOUAZIZ : Doctorants

Direction Régionale

Sandrine SOURISSEAU : Assistante de Direction – Correspondante SMQ – DD / RSE

Ressources Humaines :

Orlane MARTI : Responsable RH

Ingrid BAUDRY : Chargée RH

Communication :

Marion CAVAILLE : Chargée de marketing opérationnel

Catherine GAILLARD : Chargée Relations Media Institutionnelles

Cécile PALLU : Chargée de marketing opérationnel

Léo LECLERC : Assistant communication (CAP)

Marie NICOLINI : Chargée Projets Diffusion Scientifique

Emmanuelle ROSSIGNOL : Assistante Services Supports

Ruth LIED : Correspondante Internationale Régional (LILLE)

Vie de Campus – Rouen

Directeur d'établissement : Raynald LEVILLAIN (20%)

Valérie CADET : Responsable Exploitation Logistique

- Sophie HALLEY : Hôtesse d'accueil
- Gwénaëlle VAN DE PERRE : Hôtesse d'accueil
- Franck ALEKSIEJEW : Employé Logistique
- Igor DECONIHOUT : Employé Logistique
- Christelle BOULLAN : Employée d'entretien
- Ethan RANSON : Apprenti logistique
- Alexandre LEBAS : Apprenti sécurité

Maximilien DENIS : Responsable Exploitation SI

- Alexandre GROUCHY : Technicien d'Exploitation SI
- Nathan HAREL : Apprenti Informatique

Ecole d'Ingénieurs : Rouen/Caen

RD : Sébastien BLONDEL

Farid BAGUI : Enseignant Responsable Pédagogique (HDR)

Medjber BOUZIDI : Enseignant Responsable Pédagogique INFO (CAEN)

Alain DJEYARAMANE : Enseignant Responsable Pédagogique

Stéphanie LESEUR : Enseignante Responsable Pédagogique

Sabine FREFIELD : Correspondante Internationale Régionale

Pierre FONTAINE : Enseignant (CDD)

Camille BERNICOT : Chargée de Relations Candidats Entreprises (CAEN)

Aurélien DUMARCHE : Responsable d'unité

Marina BLANQUET : Chargée de Relations Candidats Entreprises

Anais BOUBTANE : Chargée de Relations Candidats Entreprises

Fatiha BRAHIM : Chargée d'affaires

Elodie CHRISTIAENS : Chargée de Relations Candidats Entreprises

Mathilde CRUCIANI : Apprentie Pédagogique

Laura DE SISTO : Apprentie Pédagogique (CAP)

Sophie GUESREE : Chargée de Relations Candidats Entreprises

Aurélien LAYNE : Chargée de Relations Candidats Entreprises

Camille LECRAS : Chargée de Relations Candidats Entreprises

Cécile MALLEVILLE : Chargée de Relations Candidats Entreprises

Adil TICHANI : Chargé de Relations Candidats Entreprises

Sophie HOULIER : Responsable d'Unité

François DUGARD : Enseignant Responsable Pédagogique

Benoit JEANNE : Enseignant

Antoine BAILLY : Enseignant

Lazher ZAIDI : Enseignant Chercheur

Issam TAKLA : Enseignant

Bruno MACHEFERT : Enseignant Responsable Pédagogique

Walid DEBOUCHA : Enseignant

Manja JAENICKE : Enseignante

Emilie HACHET : Enseignante

Romain MAHIEU : Enseignant

Philippe QUEDRAGO : Enseignant

Sébastien YON : Enseignant Responsable Pédagogique

Eric RINGUET : Enseignant

Christelle ETIENNE : Assistante Pédagogique

Mathilde GATINE : Apprentie commerciale

Stéphanie LABBE : Assistante Pédagogique

Julie PEGARD : Assistante Pédagogique

Christine TSAFACK : Assistante Pédagogique

Roland COMA : Responsable d'Unité

Aurélien MARTIN : Enseignant Responsable Pédagogique

Mickaël DELAMARE : Enseignant Responsable Pédagogique

Mohamad EL FALOU : Enseignant Responsable Pédagogique

Sara ZACHARIE : Enseignant Responsable Pédagogique

Figure 1: Organigramme

Présentation du sujet et des objectifs

Sujet : Mise en place d'une chaine de fabrication didactique

Dans le cadre du projet Defi&co, Nous cherchons à mettre en place dans la pédagogie une chaine de production de robots, en l'occurrence des niryo NEDs .

Objectif :

Prise en main des robots | Mise en réseau des robots | Parallélisme des tâches des robots et synchronisation | Réalisation de supports techniques pour la mise en place de la chaine | Détermination des équipements optimaux sur le robot pour qu'il puisse réaliser ses missions

DOMAINES THÉORIQUES :

Robotique, Gestion de production, suivie de production, Gestion de stock, synchronisation/ordonnancement de robots

Remarques :

La chaine devra donc être fonctionnelle et simple d'utilisation car les robots pourront être utilisés par des promotions d'informaticiens mais également par des promotions d'étudiant sans aucune expérience avec ce genre de technologie.

Résumé

Ce mémoire technique est une description du développement de la MLF(micro-learning-factory), qui est une chaîne de production didactique. J'ai détaillé les événements non pas par ordres chronologiques, mais par sujets techniques et ces parties sont détaillées chronologiquement :

Cas d'usage :

Le cas d'usage est-ce à quoi la chaîne doit ressembler une fois terminée, je m'en suis servi de cahier des charges pour ce projet lors de mon développement car j'y trouvais notamment une grande partie des contraintes

Prise en main du robot :

Fonctionnalités de base du robot, permettant de voir ce qu'il est possible de faire ou non lors du projet.

Mise en réseau :

Au fil du temps avec les contraintes qui évoluaient, il a fallu changer plusieurs fois la configuration réseau des robots. Pour un ordonnancement efficace, il leur faut pouvoir communiquer entre eux.

Contrôle et coordination des robots :

Dans cette partie sont détaillés tous les outils qui m'ont permis de contrôler les robots, de les coordonner entre eux et comment je m'en suis servi afin de réaliser la chaîne.

Présentation des packages ROS :

Description du fonctionnement de ROS (Robot operating system), et ce que j'y ai ajouté au sein des robots.

Installations et guides :

J'ai dû réaliser des guides techniques et des guides d'installation de mes solutions. Pour simplifier l'installation, j'ai réalisé un script afin de l'automatiser.

Ordonnancement avec node-red :

Description plus détaillée de l'outil node-red qui m'a permis de faire tout l'ordonnancement de la chaîne.

Le projet étant très complet et très vaste, j'ai pensé que séparer les événements en parties techniques rendrait ce mémoire plus compréhensible.

Présentation de la mission:

Logigramme

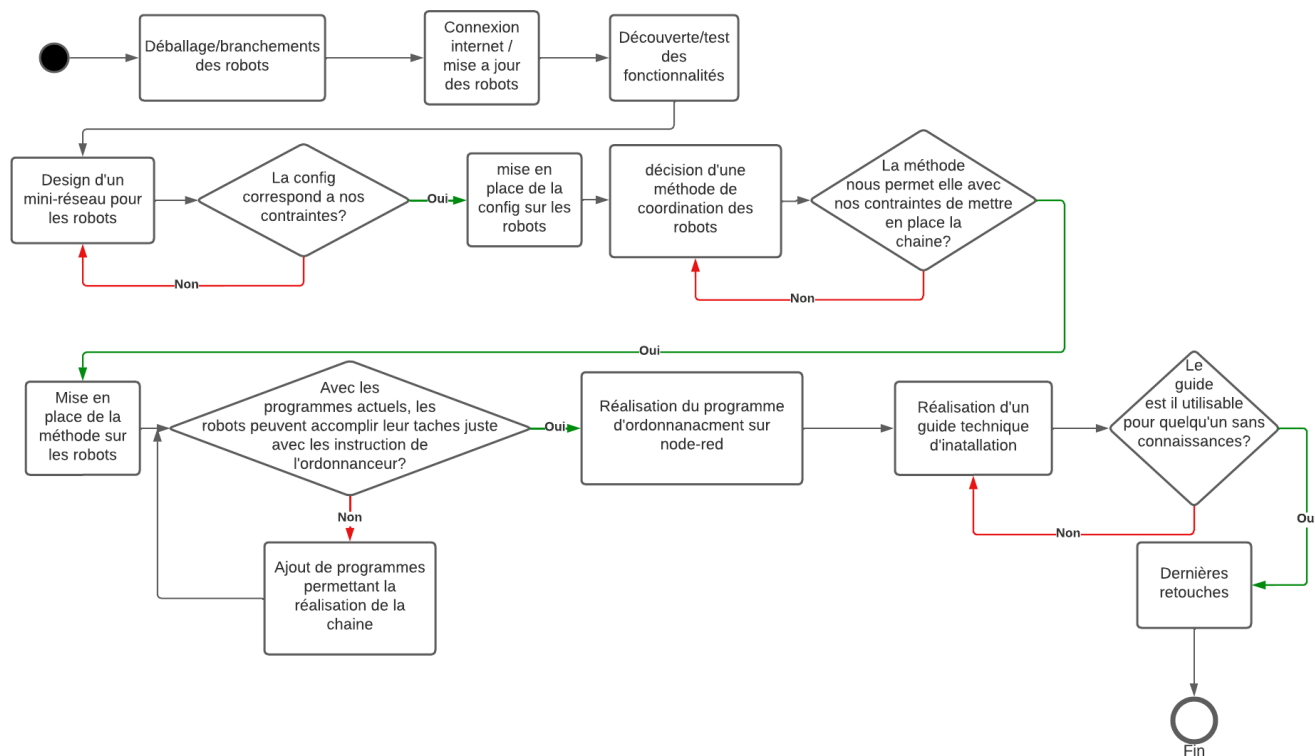


Figure 2: Logigramme d'organisation globale

Les étapes de vérifications se font avec Fabrice DUVAL, mon maître d'apprentissage ainsi qu'avec l'équipe du national, notamment Sébastien MONIER et David GARCIA lors d'une réunion.

Chaque étape du logigramme est détaillé dans la partie suivante.

Présentation détaillée

Organisation :

Avec l'outil Notion, j'ai pu mettre ne place un tableau kanban pour organiser et répartir mes tâches en entreprise et à l'école. Ces tâches avaient des indices de priorités en fonction de leur importance.

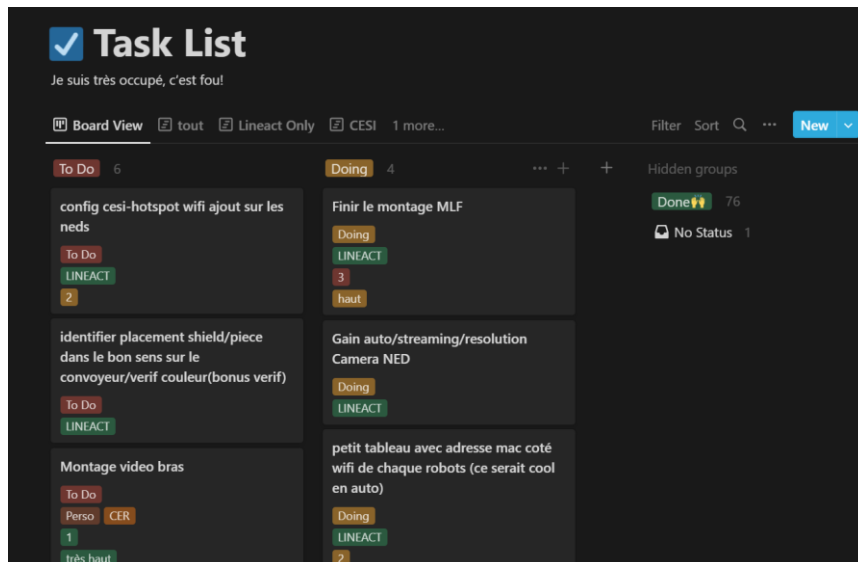
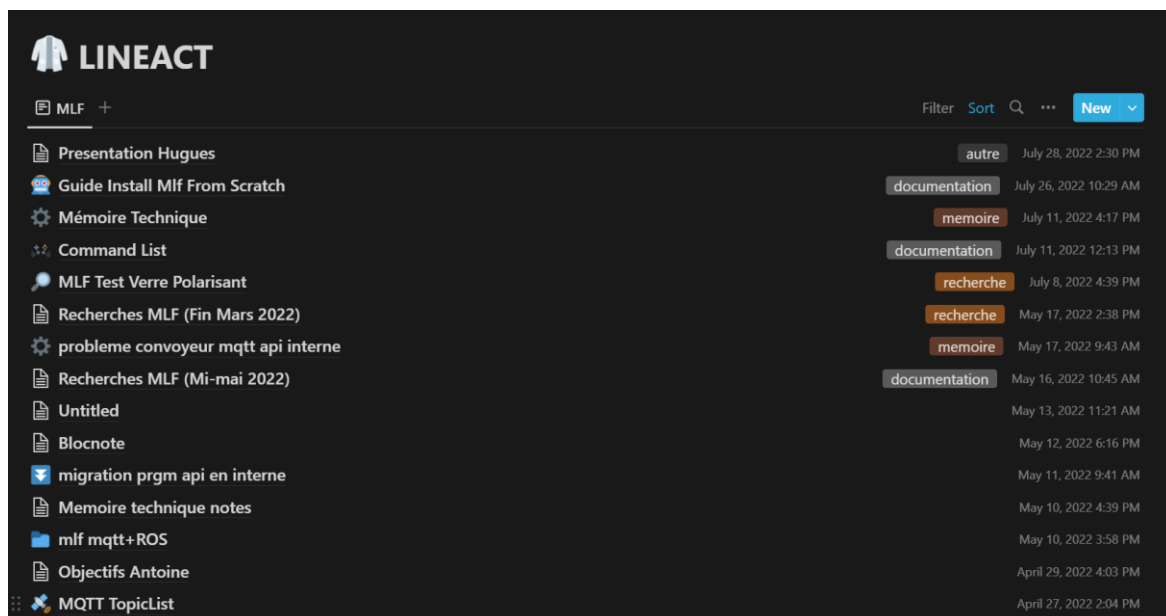


Figure 3: Tableau dynamique des taches que je me suis donné

Notion m'a aussi permis d'écrire, centraliser, stocker mes différentes documentations, c'est donc un outil que j'ai beaucoup utilisé lors de cette année d'apprentissage.



Cas d'usage défini

Une disposition de chaîne a été définie :

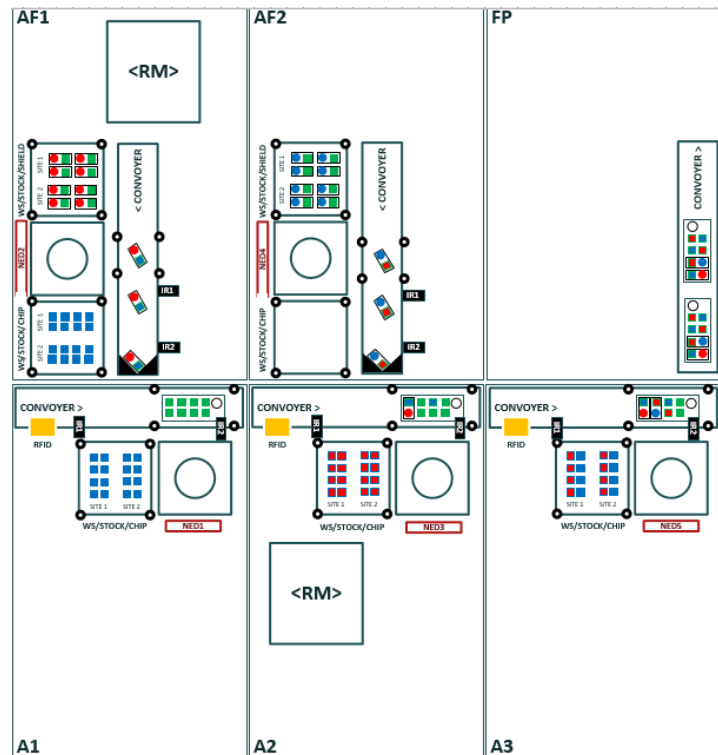
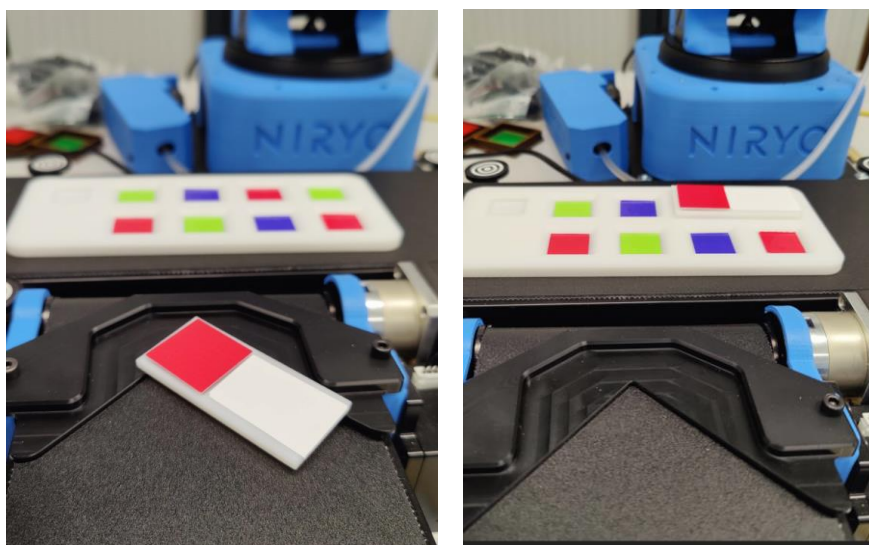


Figure 4: Cas d'usage de la MLF

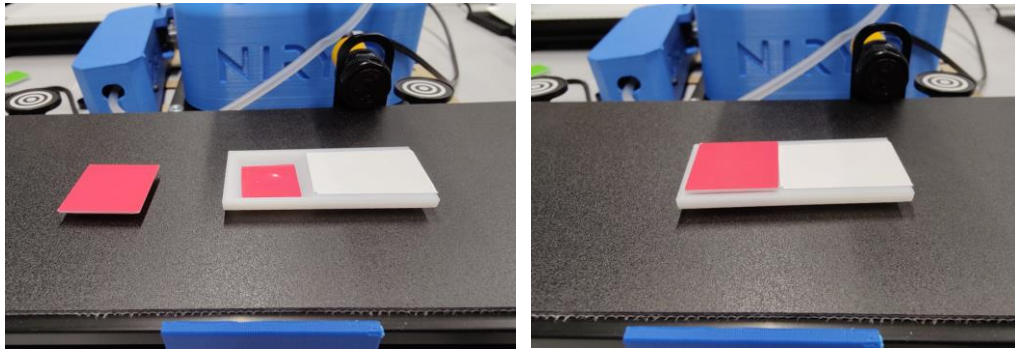
Les robots ici sont nommés « ned1/2/3/4/5 », et on leur a attribué une place, il y en a 2 types :

- Les Robots en « A », dont les convoyeurs constituent une grande ligne sur laquelle la pièce principale va transiter ils vont poser des petites pièces dites **chips** directement sur le circuit et placer les pièces moyennes confectionnés par les robots en « AF »



(Les couleurs sur le circuit étaient encore expérimentales, à l'heure actuelle toutes les cases vides sont vertes)

- Les Robots en « AF » dont les convoyeurs vont être en perpendiculaire par rapports a ceux positionnés en A, ils vont envoyer des pièces moyennes dit **shields** que le robot au bout du convoyeur sera chargé de récupérer et de placer sur le circuit



Nous voyons sur ces images le montage d'un chip sur un shield

Les circuits sont équipés d'une puce RFID afin de pouvoir les identifier de manière unique dans la chaine.

L'objectif de la chaine est donc de concevoir des simulations de circuits constituées par les pièces que l'on a pu voir ci-dessus.

Le vocabulaire de ce cas d'usage sera réutilisé lors de la présentation de la mission, notamment au niveau de la partie donnancement en **utilisant Node-Red**

Prise en main des fonctionnalités d'un robot

Matériel :

Robot :



Le Robot Niryo NED est un robot fonctionnant sur ROS disposant d'une API de control. Il a 6 axes de rotation imaginé pour l'éducation et la recherche.

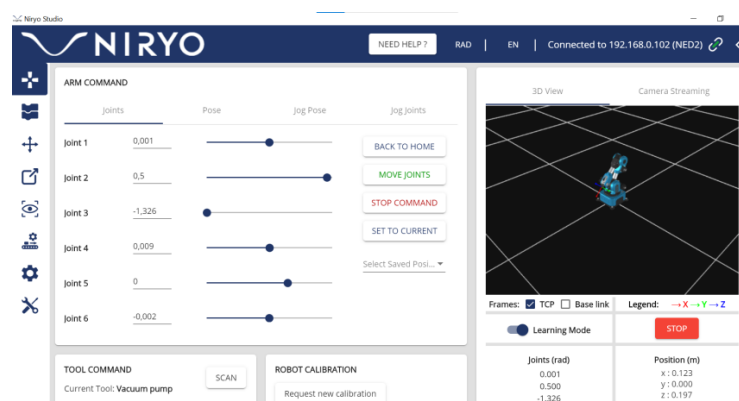
Le problème majeur de ce robot est son manque de répétabilité, il a tendance à être imprécis ou parfois à s'interrompre.

Accessoires :



Avec chaque robot, nous avons un convoyeur permettant de faire transiter nos pièces, une caméra nous permettant de faire de la reconnaissance visuelle, et différents systèmes de pinces et de ventouses. La caméra est utile pour pallier au manque de répétabilité du robot.

NiryoStudio :



NiryoStudio est un logiciel graphique utilisant l'API du robot. Le tout est intuitif, mais cela ne permet pas de coordination entre les robots. J'ai souvent utilisé ce logiciel pour avoir un rendu caméra du robot.

Mise en réseau

A l'origine, nous n'avons que deux moyens de se connecter au robot :

Méthode	Ethernet direct	Hotspot du robot
IP	10.10.10.10	169.254.200.200

Nous n'avons pas d'accès à aucun écran étant donné que ce sont des robots, le raspberry PI est à l'intérieur. Nous ne nous connectons donc pas à l'HDMI pour un accès écran. Il va donc falloir utiliser SSH¹. La première chose que j'ai faite a été de mettre les robots sur le wifi intitulé « cesi_recherche », ainsi nous pouvions leur bloquer une IP fixe facilement à l'aide de l'adresse MAC² de chacun des robots. Le souci avec cette méthode sont les latences dues aux interférences. Nous avons donc choisi de faire un mini-réseau en Ethernet pour pouvoir supprimer ces latences. Un routeur a été connecté au LAN du bâtiment pour l'accès internet, mais également pour faire notre adressage IP fixe :

NOM	IP
ned 1	192.168.0.101
ned 2	192.168.0.102
ned 3	192.168.0.103
ned 4	192.168.0.104
ned 5	192.168.0.105
ned 6	192.168.0.106

Au niveau du routeur, qui gère le DHCP³ j'ai bloqué les IP afin de permettre un adressage fixe pour ces machines, toujours avec l'adresse MAC. Lors d'une réunion, nous nous sommes rendus compte que tous les centres ne pouvaient pas prendre de routeur pour leurs MLF, ainsi, nous avons dû nous adapter, en supprimant le routeur et en connectant directement le switch au LAN⁴ du CESI en question. Les IP ont également dû changer, le tout est répertorié sur l'excel ci-dessous. Le service informatique de chaque centre s'occupera de l'autorisations de l'autorisation pour chaque robot à accéder au réseau.

Nom	Adresse IP	Masque réseau	Passerelle par défaut	DNS 1	DNS 2
Serv-00	192.168.24.100	255.255.255.0	192.168.24.100	192.168.24.100	-
Robar-01	192.168.24.1	255.255.255.0	192.168.24.100	192.168.24.100	-
Robar-02	192.168.24.2	255.255.255.0	192.168.24.100	192.168.24.100	-
Robar-03	192.168.24.3	255.255.255.0	192.168.24.100	192.168.24.100	-
Robar-04	192.168.24.4	255.255.255.0	192.168.24.100	192.168.24.100	-
Robar-05	192.168.24.5	255.255.255.0	192.168.24.100	192.168.24.100	-
Robar-06	192.168.24.6	255.255.255.0	192.168.24.100	192.168.24.100	-
...					
Robar-10	192.168.24.10	255.255.255.0	192.168.24.100	192.168.24.100	-
Robmo-01	192.168.24.11	255.255.255.0	192.168.24.100	192.168.24.100	-
Robmo-02	192.168.24.12	255.255.255.0	192.168.24.100	192.168.24.100	-

Après pas mal de changements, nous avons pu avoir une architecture qui fonctionne sur tous les centres avec les contraintes qui vont avec.

¹ Secure Shell : protocole de contrôle d'un terminal à distance

² Adresse physique unique propre à chaque machine

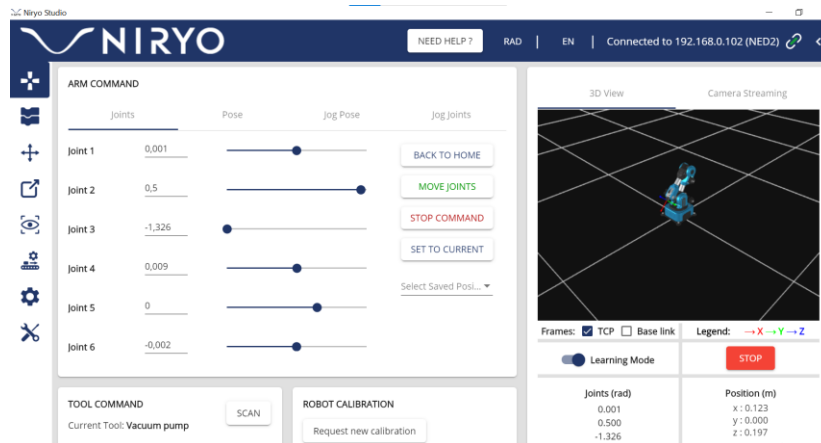
³ Adressage automatique des machines connectés

⁴ Réseau Local

Contrôle et coordination des robots

Niryo Studio :

NiryoStudio est un logiciel fourni par Niryo (les concepteurs du robot NED). Il permet notamment d'avoir une visualisation en temps réel de la caméra et d'avoir un jumeau numérique 3D du robot. Nous avons une interface graphique pour pouvoir donner au robot toute sortes d'instructions.



Le logiciel est très pratique pour avoir un rendu caméra et une vision de la position fictive du robot. Etant donné que le robot est peu fiable et qu'il se « perd » la position détectée dans son programme interne et donc qu'on peut voir sur le jumeau numérique ne correspond pas toujours à sa position réelle (la position qu'on perçoit physiquement sur le robot). Avec le jumeau numérique, je peux donc très vite repérer une erreur. Les mises à jour des programmes du robot sont également proposées par Niryo via l'application. Nous ne contrôlons pas les robots avec ce logiciel car il faut pouvoir gérer plusieurs robots à la fois.

PyNiryo:

PyNiryo est un package python qui regroupe l'ensemble des commandes que l'on peut effectuer sur NiryoStudio. Le tout est cependant séquentiel, donc si nous voulons contrôler deux robots en même temps, par exemple lever le bras, le robot n°1 lève le bras, puis le robot n°2 lève le bras.

Threads python utilisant l'API :

Un Thread est un sous-processus qui effectue les instructions de manière indépendante du processus principal, donc du programme principal.

En utilisant des threads et donc en en faisant un par robot, Nous pouvons tout contrôler en même temps. Il reste un souci, toutes les instructions sont codées en dur dans un programme python, or on voulait une solution facile d'utilisation et modulable pour des personnes sans expérience. Il a donc fallu changer la manière de donner les instructions au robot.

MQTT :

MQTT est un protocole de messagerie publish-subscribe basé sur TCP/IP.

Un topic est une section où transitent des messages, on peut avoir plusieurs topics si on veut séparer nos messages, les mettre dans des espaces différents.

-Le broker est l'endroit où va transiter tous les messages, il va gérer la rediffusion en fonction de quelle machine est abonné à quel topic.

-Le client peut effectuer deux actions :

*Publish : publier dans un/plusieurs topics

*Subscribe s'abonner à un/des topics

Le schéma ci-dessous illustre avec des thermomètres les rôles cités précédemment.

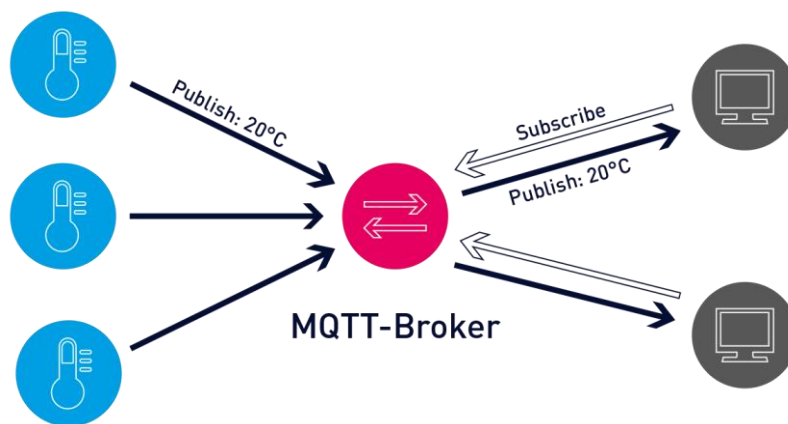


Figure 5: Schéma explicatif d'MQTT

En créant des topics sur les informations qui nous intéressent, on peut avoir les infos en temps réel et donc faire réagir un système. Nous pouvons également envoyer nos messages pour faire réagir le robot. Avec ces deux choses, on peut construire un **ordonnancement**. (L'ordonnancement sera détaillé dans la partie Node-Red).

Afin de garder une chaîne modulable chaque robot sera son propre broker mqtt, on aura donc sur notre cas d'usage 5 robots qui seront brokers et clients entre eux, le client supplémentaire sera le PC utilisateur.

- move (m1, m2, m3, m4, m5, m6)
- StartLine
- StartPerpendiculaire
- assembling workspaceName -rotationZero(bool) -Zdown(z)
- pick workspaceName -translation(x, y, z)
- change Color into color*
- change Shape into shape*
- conveyor speed direction
- translation x y z
- wait seconds
- release
- finished
- grasp

Afin de garantir cette modularité, j'ai réalisé un système de commande. Ainsi le PC utilisateur aura pour seul rôle d'écouter les différents topics et d'envoyer des commandes. Je les ai répertoriées dans le document dont vous pouvez voir une capture d'écran ci-contre.

Par exemple, pour ordonner au robot de mettre tous ses moteurs à angle 0, Il faut se connecter en mqtt, puis publier sur le topic « /command », la commande :

`move (0,0,0,0,0,0)`

Les paramètres avec des tirets sont des paramètres facultatifs. Une documentation plus détaillée sera effectuée lorsque la chaine sera fonctionnelle. Mais j'ai réalisé un script permettant de déchiffrer et traiter ces commandes

Node-Red :

Node-Red est un outil de développement basé sur les flux pour la programmation visuelle.

Le fonctionnement est simple :

Lorsque le serveur est lancé, les flows(flux) vont démarrer :

```
rl@EMartin:~$ node-red
14 Aug 18:22:07 - [info]

Welcome to Node-RED
=====

14 Aug 18:22:07 - [info] Node-RED version: v2.2.2
14 Aug 18:22:07 - [info] Node.js version: v16.15.0
14 Aug 18:22:07 - [info] Linux 4.4.0-19041-Microsoft x64 LE
14 Aug 18:22:09 - [info] Loading palette nodes
14 Aug 18:22:12 - [info] Dashboard version 3.1.6 started at /ui
14 Aug 18:22:12 - [info] Settings file : /home/rl/.node-red/settings.js
14 Aug 18:22:12 - [info] Context store : 'default' [module=memory]
14 Aug 18:22:12 - [info] User directory : /home/rl/.node-red
14 Aug 18:22:12 - [warn] Projects disabled : editorTheme.projects.enabled=false
14 Aug 18:22:12 - [info] Flows file : /home/rl/.node-red/flows.json
14 Aug 18:22:12 - [info] Server now running at http://127.0.0.1:1880/
14 Aug 18:22:12 - [warn]

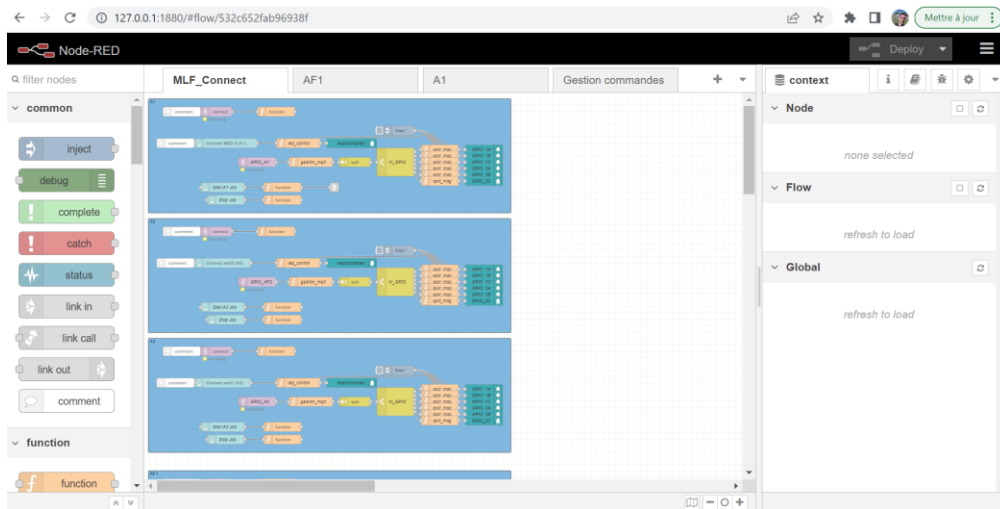
-----
Your flow credentials file is encrypted using a system-generated key.

If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.

You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.
-----

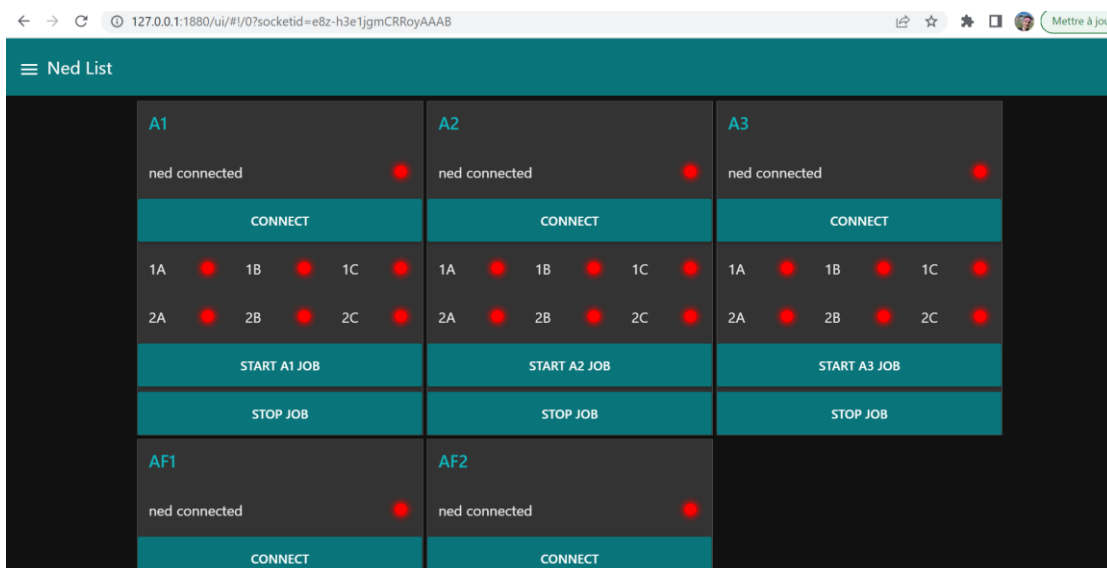
14 Aug 18:22:12 - [info] Starting flows
14 Aug 18:22:12 - [info] Started flows
```

Nous avons ensuite via un navigateur accès à une interface de développement en bloc, il y a la possibilité de faire ses propres fonctions en JavaScript, donc de créer ses propres blocs personnalisés.



L'avantage c'est qu'il y a une très bonne compatibilité avec MQTT et que nous pouvons vraiment traiter les messages que l'on reçoit et que nous voulons envoyer au système.

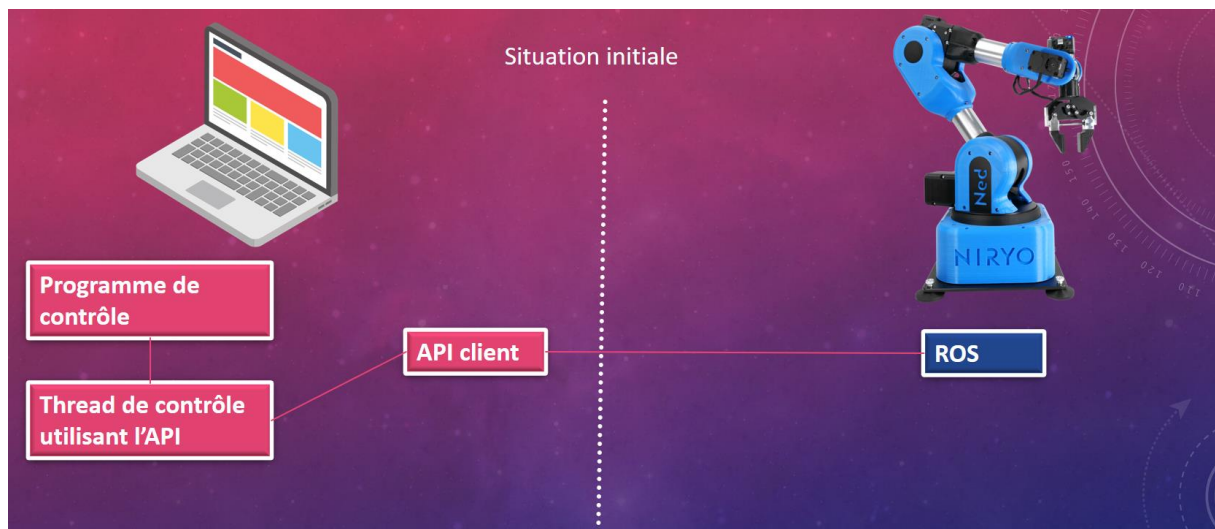
Il y a en installant un package via node-red, une interface graphique qu'on peut créer très facilement pour un développement rapide :



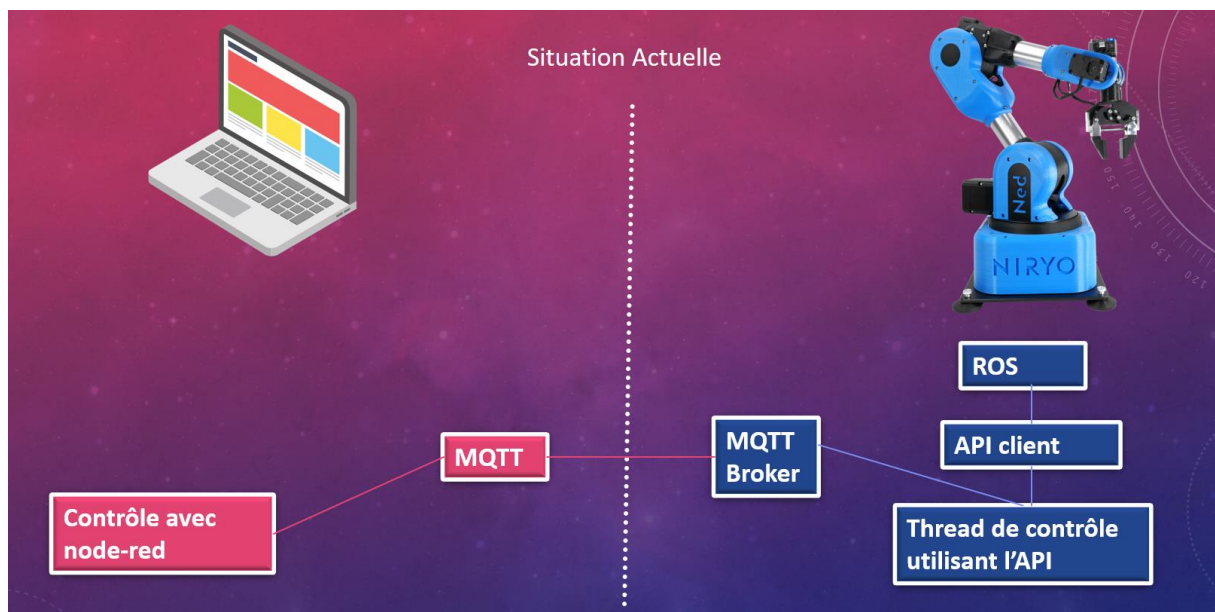
Les numéros types 1A/1B/... représentent les noms des pins des robots et la LED indique leurs états électriques en temps réel, vert ou rouge respectivement pour « true » ou « false ».

C'est ce genre d'interface qui rends la chaine facile d'utilisation. Le développement en bloc de node-red devrait permettre une compréhension plus simple pour quelqu'un qui n'est pas à l'aise avec le code, sachant que les fonctions sont toutes écrites en JavaScript dedans.

Bilan de l'évolution de la partie contrôle au sein d'un robot :



Dans un premier lieu, nous avons cette disposition avec un programme et ses threads qui ordonnait et gérait les déplacements de tous les robots



A l'heure actuelle, Nous avons sur le PC de l'utilisateur que node-red qui va gérer l'ordonnancement et va envoyer des commandes MQTT à chaque robot. Un thread que j'ai ajouté et qui démarrera au démarrage à l'aide d'un script en bash ⁵permettra de communiquer avec l'API en local.

L'API va ensuite communiquer avec le ROS (robot operating system), dont les programmes qui tournent vont directement impacter le robot et déclencher les mouvements.

⁵ Langage des commandes de consoles le plus répandu dans le monde, dans notre cas, les commandes Linux.

Présentation des autres packages ROS

Qu'est-ce que ROS ?

ROS (robot operating system) est un ensemble d'outils informatiques sous formes de logiciels libres open source, permettant de développer des logiciels pour la robotique. Il fonctionne sur Ubuntu exclusivement (sauf pour les dernières versions mais c'est plutôt expérimental).

De nombreuses fonctionnalités permettent de faciliter le développement :

- Architecture de communication inter-processus et inter-machine :
Via les nodes et les topics, un node lancé sur une machine peut écouter un topic où un node lancé une autre machine peut publier. Mais deux nodes lancés sur une même machine peuvent également communiquer entre eux. Cela sert notamment pour faire des diagnostics ou mettre des états à notre système.
- Serveur de paramètre
Les nodes lancés utilisent le serveur de paramètre pour stocker et récupérer des paramètres lors de leur exécution.
- Système d'enregistrement et de rejeu
On peut enregistrer les mouvements que va faire notre robot afin de le « rejouer » ou le répéter en boucle. Cela peut être utile par exemple pour un robot industriel intervenant dans une chaîne de production.
- Système de test
Permet de tester l'intégration d'un package(groupe de programmes réalisés afin de donner des fonctionnalités à un robot) dans ROS si on veut le rendre publique.
- Simulateur
Permet de Simuler un robot si on ne l'a pas physiquement, on aura une visualisation 3D du robot sans même l'avoir physiquement

Nous avons des outils utilisés pour créer, lancer et distribuer des logiciels basés sur ROS. Nous avons également des commandes permettant la création simplifiée de paquets et des outils simplifiant la navigation dans le système de fichier, il existe aussi des commandes permettant de lancer des programmes dans des paquets spécifiques :

Par exemple : « rosrn paquet1 node1 » lancera le node node1 situé dans le paquet paquet1 et on peut lancer la commande ou que l'on se situe dans le système de fichier linux.

Dans ces paquets, Il y a les différents programmes qui permettent d'opérer tout en communiquant, Nous utilisons en général les langages : C++(utilisant la bibliothèque roscpp) et python (utilisant la bibliothèque rospy).

Au niveau de la communication, l'architecture est plutôt simple. On définit tout d'abord une machine « maître » :le rosmaster. Cette machine gèrera toutes les nouvelles communications. Par la suite, les programmes que nous allons lancer vont s'appeler des **nodes** (nœuds), on les distingue en deux catégories :

- Les nodes Publisher qui vont publier dans un ou des topics.
- Les nodes Subscriber qui vont écouter un ou des topics.

Un topic sera donc une place où on envoie des messages. Un node peut être à la fois Publisher et Subscriber, il peut donc écouter et recevoir de plusieurs topics en même temps

L'intérêt consiste à faire de la communication interne dans la machine afin de coordonner nos programmes mais également externe afin de coordonner différentes machines connectées à un même rosmaster.

Le fonctionnement global de la communication est similaire a MQTT qui est aussi un publish-subscribe

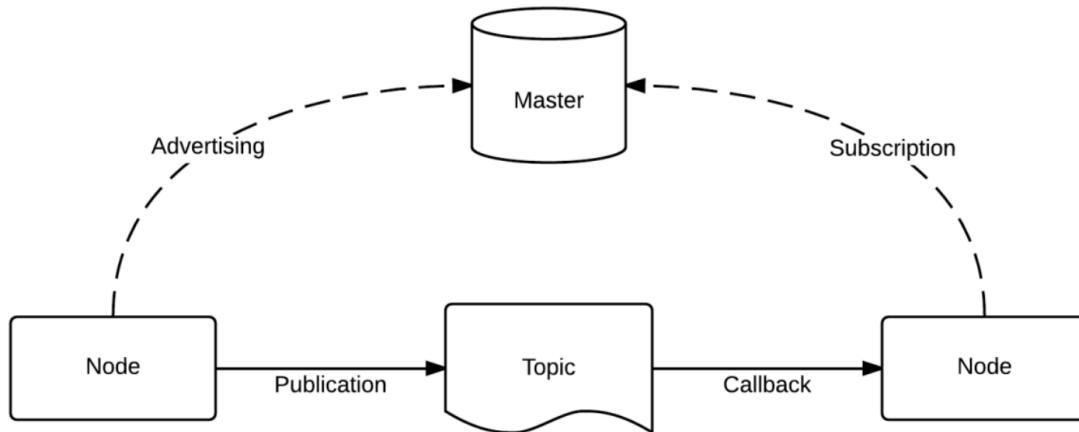
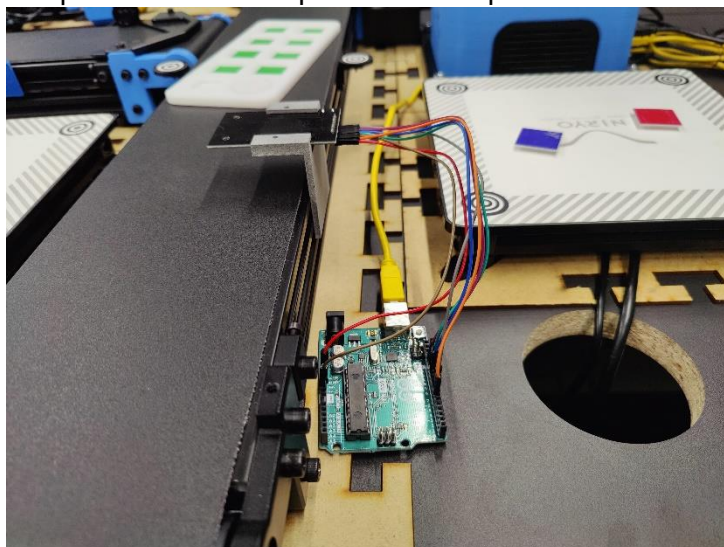


Figure 6: Schéma explicatif de la communication de ROS

Packages ajoutés au ROS :

- **mlf_extra_port :**
L'API traite toutes les commandes en séquentiel. Cela implique que nous ne pouvons pas contrôler les convoyeurs en même temps que le robot effectue un mouvement par exemple. Ce package a pour but en fonctionnant sur des sockets de se mettre sur un autre port et de contrôler le convoyeur séparément de l'API ou on va maintenant gérer exclusivement le robot.
- **RFID :**
Nous avons ajouté un Arduino avec un capteur RFID pour pouvoir identifier les circuits qui transitent sur le convoyeur. En ajoutant ce capteur supplémentaire, on a donc créé un package qui va publier l'état du capteur sur un topic.



- `mlf_mqtt`
Ce package sert à publier les informations que nous souhaitons (états des capteurs, etc...) dans des topics mqtt respectifs
- `launch_bringup`
Ce package contient juste un fichier launch (permettant de lancer plusieurs programmes sur plusieurs packages). Il nous sert à lancer nos programmes que l'on a ajouté au démarrage du robot.

Pour le moment, seuls ces packages ont été développés et été ajoutés aux robots au niveau du ROS.






En les installant on pourra ensuite contrôler les robots uniquement avec des commandes MQTT sur chacun leur topic /command

Installation et guide

Afin que tous les centres aient la même base de travail, j'ai réalisé un guide technique de l'installation afin que tous les robots dans chaque centre aient strictement le même fonctionnement interne.

J'ai dans un premier temps fait un descriptif de toutes les installations/commandes qu'il fallait effectuer dans le Shell linux, puis pour faire au plus simple pour quelqu'un qui n'aurait pas d'expérience en la matière, j'ai effectué un script d'installation automatisé qui installe tout en une seule commande. Car en effet il y avait des installations de bibliothèques, d'environnement virtuels python pour pouvoir utiliser python 2 ET python 3, des build de l'environnement ROS à effectuer, des fichiers sources à modifier pour un lancement des programmes automatique au démarrage du robot... Une erreur de frappe suffirait à faire basculer tout le fonctionnement des packages, sachant qu'il faut le faire sur les 6 robots, nous avons eu la conclusion que plus l'installation était automatisée, moins nous avions de chances de faire des erreurs.

Il y a juste à importer les fichiers en SSH, il a fallu expliquer ces étapes dans les moindres détails, puis se connecter en SSH pour enfin lancer la commande.

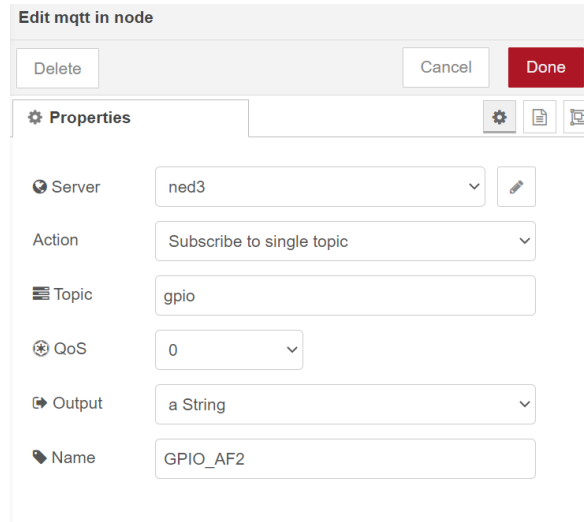
<div> <div>> install_mlf ></div> <div>Rechercher dans : install</div> </div>		
Nom	Modifié le	Type
 <code>pkg_python_self_api</code>	20/05/2022 16:23	Dossier de fichiers
 <code>ros</code>	25/07/2022 10:51	Dossier de fichiers
 <code>bootStart.sh</code>	22/05/2022 20:42	Fichier source SH
 <code>install_mlf.sh</code>	08/07/2022 13:49	Fichier source SH
 <code>network.sh</code>	26/07/2022 10:22	Fichier source SH

Une fois le script « `install_mlf.sh` » exécuté, il n'y a plus qu'à redémarrer le robot pour pouvoir profiter des nouvelles fonctionnalités.

Ordonnancement en utilisant Node-Red

Pour cette partie, je recommande une relecture du Cas d'usage défini [Cas d'usage défini](#).

Node-Red va servir d'ordonnanceur, l'objectif va être d'envoyer les bonnes commandes MQTT au bon moment, aux robots pour qu'ils effectuent leurs tâches de manière synchronisée.



Le node paramétré ci-dessus nous permet d'écouter les messages qui transitent sur le topic « gpio » du ned3 (nous avons parlé du package ros mlf_mqtt, c'est lui qui va publier les informations qui transitent sur le ROS vers le MQTT, ce qui nous permet de pouvoir les exploiter ici).

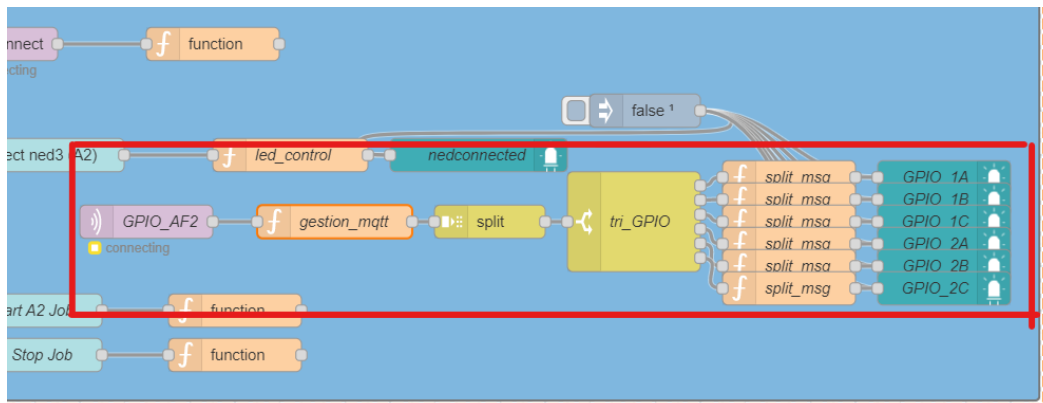
A chaque fois qu'on aura un message sur le topic en question, un déclenchement d'évènement peut être organisé en réaction à celui-ci

Ensuite, en javascript, nous pouvons éditer nos propres nodes, ainsi j'ai traité les données afin de les mettre dans un format particulier que je peux afficher :

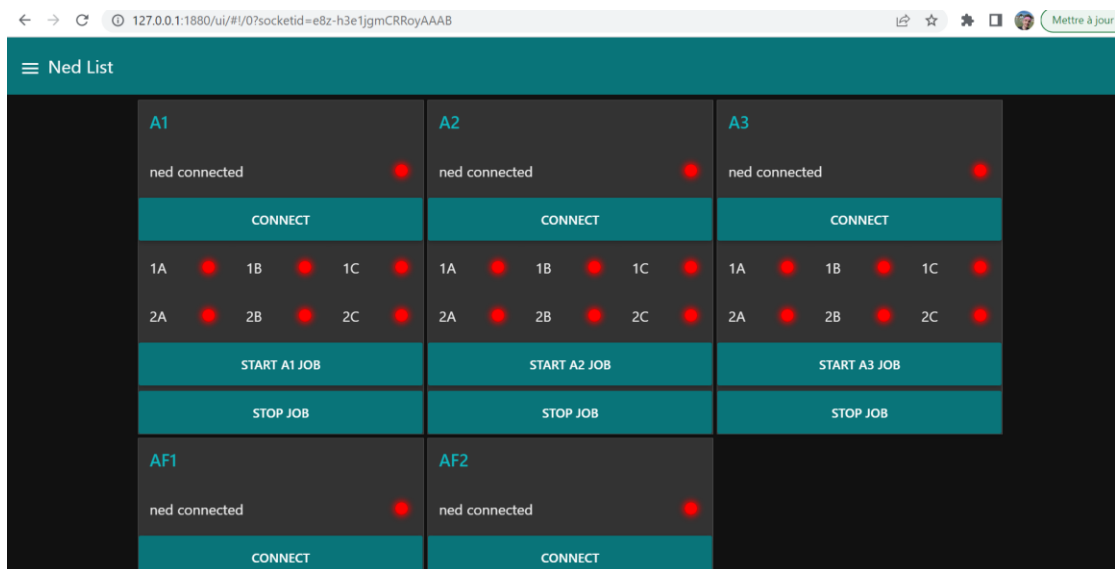


```
1 if(global.get("A2") == true){
2   pins = []
3   traductedPins = [];
4   table = msg.payload.split("{")[1];
5   table = table.split("}")[0];
6   table = table.split(",");
7   table.forEach(pin =>{
8     pins.push([pin.split(":")[0], pin.split(":")[1]]);
9   });
10  pins.forEach(pin=>{
11    if(pin[1] == 'True'){
12      traductedPins.push(pin[0], true)
13    }
14    if(pin[1] == 'False'){
15      traductedPins.push(pin[0], false)
16    }
17  });
```

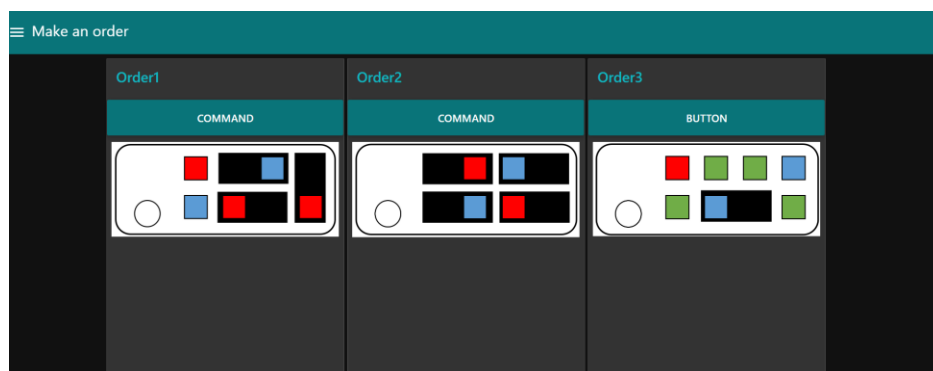
Ainsi, (pour un robot, AF2 en l'occurrence) nous avons avec ces nodes encadrés en rouge :



le résultat suivant sur l'interface graphique (uniquement les 6 LED représentant les GPIO⁶):



L'objectif est de produire des pièces, donc une interface de commande de pièces a été effectuée :

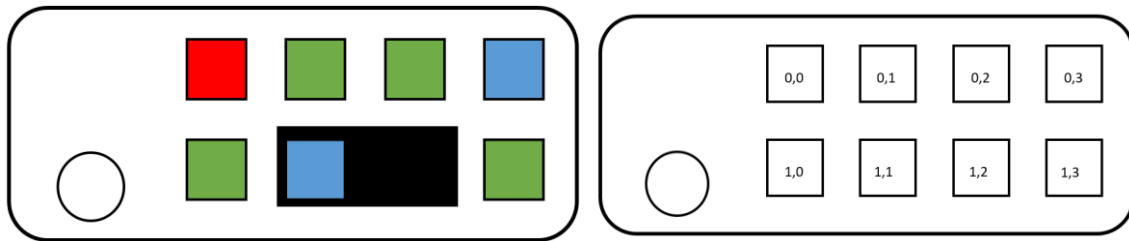


Ainsi, l'utilisateur n'aura qu'à connecter les robots, puis appuyer sur un bouton commande pour lancer la production.

⁶ Pins similaires à me ceux présents sur un Arduino, on peut lire leur état électrique ou leur imposer un état.

Avant toute chose, il a fallu modéliser nos données, c'est-à-dire créer une structure bien définie pour chaque pièce qui vont transiter sur la chaîne.

Circuit en lui-même :



Afin de modéliser les emplacements d'un circuit, le plus logique m'a semblé de faire une matrice, de la forme : $[[a1,a2,a3,a4],$

$[b1,b2,b3,b4]]$

Ensuite, pour pouvoir placer nos pièces dessus, il a fallu également leur donner une structure. (Afin de remplir notre matrice).

```
a1 = ["c1", "chip", "red"]
//[id,type,color]
```

J'ai défini notre chip par un identifiant, son type et sa couleur.

```
a2 = ["s1", "shield", "blue", false]
a3 = ["s1", "shield", "blue", true]
//[id, type, color, is it the color side?]
```

Le shield a une structure plus complexe, en effet, prenant 2 emplacements sur le circuit, il a fallu séparer la pièce en deux parties au niveau des données. J'ai donc défini chaque partie comme pour le chip et le dernier

élément du tableau est le booléen « est ce qu'on est du côté où il y a le chip ? » Qu'on appellera ColorSide. L'avantage de cette démarche est de pouvoir contrôler notamment l'orientation de notre pièce sur le circuit en analysant la matrice.

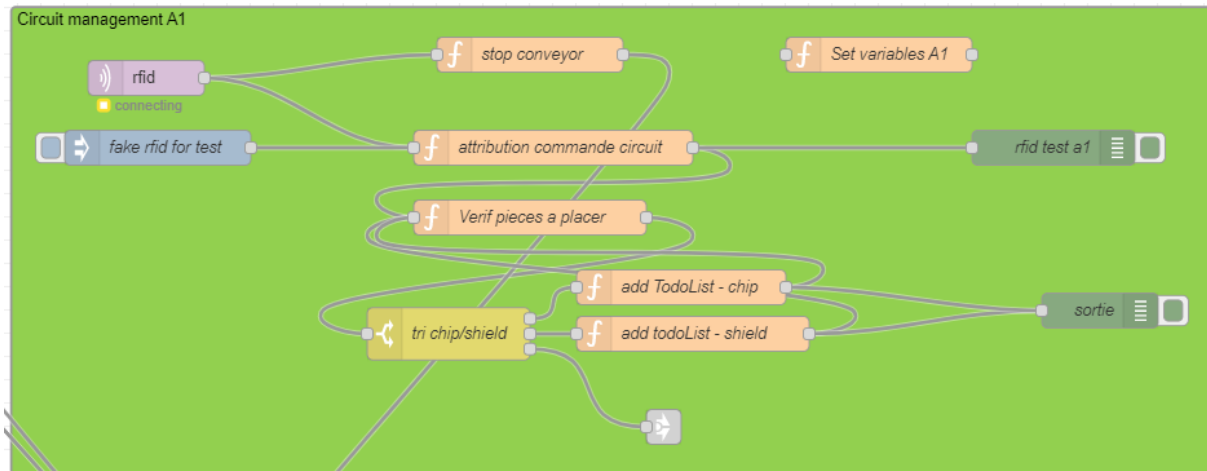
En remplissant la matrice avec ces deux types de pièces, nous avons un modèle de pièce finie qu'on peut commander.

Pour modéliser un circuit en cours de transition sur la chaîne, j'ai réalisé la structure suivante :

```
class Circuit {
  constructor(id, position, itemsToPlace = [], itemsAlreadyPlaced = []) {
    this.id = id; //rfid
    this.position = position; //a quel robot est-il situé?
    this.itemsToPlace = itemsToPlace; //commande initiale
    this.itemsAlreadyPlaced = itemsAlreadyPlaced; //objets present reellement sur le circuit
  }
}
```

L'objet est défini par son RFID, sa position sur la chaîne, la matrice de produit final attribué, et les objets qui ont déjà été placés sur le circuit. Avec ces éléments, nous pouvons maintenant contrôler notre production.

Comment ordonnancer les robots pour qu'ils effectuent uniquement les tâches nécessaires à la réalisation des pièces commandées ?



Au niveau du robot A1, il y aura la première détection RFID de circuit. S'il y a une commande de faite, nous aurons une attribution de la commande vers la pièce en question, puis on commencera à donner les pièces à produire aux robots en fonction des couleurs des pièces qu'ils gèrent.

En effet, par exemple, les robots A1/AF1 gèrent les pièces rouges tandis que les pièces A2/AF2 gèrent les pièces bleues.

Nous nous assurons avec la structure du circuit de produire le bon nombre de pièces et de les assembler au bon endroit avant d'envoyer la pièce vers la suite de la production. Pour cela il faut se baser sur la structure que nous avons définie plus haut.

Avec ce fonctionnement, nous arrivons au bon nombre de pièces placés au bon endroit en fonction du circuit que nous avons commandé.

Pour mettre les robots en action, une chaîne de commandes MQTT se lance en fonctions des pièces attribuées aux robots.

Bilan et conclusion

Ce projet était très vaste. J'ai pu voir un grand nombre de notions très différentes les unes des autres, autant en robotique, qu'en réseau, qu'en gestion de projet/de temps... J'ai dû aussi m'adapter à différentes contraintes et problèmes techniques.

Par exemple les stickers sur les pièces étaient trop brillants au début du projet, nous avions des reflets blancs et la caméra ne détectait plus les couleurs. Nous avons tenté différentes techniques comme mettre du verre polarisant sur les caméras, mais au final un type de stickers particulier a été pris, celui-ci ne reflétait pas la lumière.

Les problèmes que j'ai rencontrés, les solutions que j'ai trouvées en réponses et les compétences que j'ai acquises en trouvant ces réponses, m'ont permis de réaliser la chaîne de production didactique qu'est la MLF.

Avec l'expérience que j'ai acquise lors de ce projet, j'ai développé plusieurs compétences de ma cible PFI et qui me seront utiles au long de mon parcours professionnel.

Pistes d'améliorations :

- Plutôt que de passer par l'API à la réception des commandes MQTT dans le robot, on pourrait directement envoyer les commandes sur ROS, cela supprimerait directement une étape.

- Il manque encore des fonctions de caméras à développer, comme par exemple le choix de l'emplacement ou poser, aujourd'hui l'algorithme de détection des couleurs ne nous permet pas de choisir quel carré vert on sélectionne, on va poser sur le premier que l'on voit. C'est un problème pour le bon positionnement des pièces sur le circuit. Cependant je travaille avec Antoine, un autre apprenti, et je le dirige avec Nicolas Ragot son maître d'apprentissage sur le développement d'un algorithme qui me permettra de choisir mes emplacements. Il travaille dessus en parallèle de la fin de mon développement sur la chaîne.

- Aujourd'hui, pour le développement on a le choix entre 3 circuits différents, mais faire une interface de commande de circuit modulable apporterait bien plus de possibilités.

Au final, bien que tout ne soit pas encore opérationnel à 100% au moment où j'écris ce mémoire j'estime avoir rempli mon objectif dans le sens où la chaîne sera opérationnelle dans les prochaines semaines.

Remerciements :

En premier lieu, je tiens particulièrement à remercier Fabrice DUVAL, mon maître d'apprentissage. La confiance qu'il m'a accordée tout en me guidant dans mes objectifs, l'autonomie et la liberté qu'il m'a laissée m'ont permis de développer mes compétences.

Je tiens également à remercier Ayman DAMOUN qui m'a donné de précieux conseils et outils techniques que je pourrais réutiliser lors de mon parcours professionnel.

Merci à Nicolas RAGOT, à David GARCIA et à Sébastien MONNIER pour avoir suivi de près mon travail et pour m'avoir encouragé comme ils l'ont fait, cela a boosté ma motivation lors de ce projet.

Pour finir, je tiens à remercier Nicolas BRIANT, Léo CHEVALLIER, Alexandre COURALLET, Hugues BIBAUT et Louis CHOCHOY pour leur aide dans la réalisation de ce mémoire, ainsi que pour la réalisation de pièces 3D dont j'avais besoin dans le cadre du projet.

Bibliographie :

ROS : <https://www.ros.org/>

MQTT : <https://mqtt.org/>

Node-Red : <https://nodered.org/>

Niryo Ned documentation: <https://docs.niryo.com/product/ned/v4.0.0/en/index.html>