Rapport de Projet

EI4 AGI

Contrôle d'un convoi de véhicule

Projet réalisé par :

DAUMAS Maxime

LORON Bastien

Projet encadré par :

AUTRIQUE LAURENT

Remerciements

Avant de débuter le développement de cette expérience, il paraît naturel d'entamer ce rapport par des remerciements aux diverses personnes qui nous ont entourés pendant la durée de ce projet.

Nous tenons à remercier :

- Monsieur AUTRIQUE L., initiateur du projet, pour la confiance qu'il nous a accordée.
- Monsieur COTTENCEAU B., enseignant chercheur pour nous avoir fourni une partie du matériel.
- L'ensemble des enseignants, sur lesquels nous avons pu compter pour répondre à nos questions, nous conseiller et nous transmettre leurs connaissances.
- L'ISTIA, l'école d'ingénieur de l'université d'Angers pour nous avoir alloué le budget nécessaire à l'acquisition du matériel.
- Atlantique Composant à Beaucouzé et JMR (Jouet Modèle Réduit) à Angers pour leur disponibilité et leurs conseils.

Sommaire

Daumas Maxime
Bastien Loron

Introduction

Dans le cadre de notre quatrième année à l'Institut des sciences et techniques de l'ingénieur d'Angers nous avons eu l'opportunité de travailler en groupe afin de réaliser un projet. Cette expérience nous a permis de mettre en pratique l'ensemble des connaissances acquises depuis le début de notre cursus.

La problématique proposée pour ce projet repose sur un besoin bien précis. Aujourd'hui le nombre de véhicules sur les routes est en constante augmentation et la plupart du temps la fluidité du trafic n'est pas satisfaisante.

Dans ce contexte nous utiliserons un système de locomotives pour simuler des véhicules en circulation. L'enjeu de ce projet est de maîtriser la distance qui sépare un véhicule de celui qui le précède tout en faisant évoluer de manière automatisée un convoi de véhicules.

Ce rapport présente dans un premier temps la gestion du projet. Dans la seconde partie nous détaillerons les parties techniques du projet. La dernière partie présente la conclusion.

Gestion de projet

I. Finalité du projet

1. Contexte du Projet

Dans de nombreux pays industrialisés, le nombre de véhicules sur les routes est en constante augmentation. Les infrastructures routières existantes sont vieillissantes et s'avèrent assez vite saturées, ne pouvant assurer la fluidité du trafic. Ainsi, le samedi 17 aout 2013, un record de 877 km de bouchons cumulés a été atteint en France. Par définition un bouchon est l'accumulation, sur une ou plusieurs fils continues et sur une distance d'au moins 500 mètres, de véhicules progressant à une allure très lente et par bonds.

Afin d'éviter la congestion du trafic, une solution existe. Celle-ci consiste à faire rouler des « trains de véhicule » de manière durable et assurant la sécurité. Il s'agit de faire évoluer de manière automatisée un convoi de véhicules. Pour chacun des véhicules formant un convoi, l'enjeu est de maîtriser la distance qui le sépare du véhicule qui le précède.

2. Objectif

L'objectif de ce projet est de modéliser un tel « train de véhicule ».Pour cela nous ferons circuler à vitesse accélérée un convoi de locomotives non attachées de manière à minimiser autant que possible la longueur du convoi. Des perturbations telles que: ralentissement d'une locomotive, accélération du leader, adjonction d'une locomotive, retrait d'une locomotive, seront étudiées afin de juger la pertinence des méthodes de contrôle proposées.

3. Performance

La bonne performance du système sera évaluée par différents indicateurs tels que la stabilité de la vitesse des locomotives, la distance entre les locomotives, la longueur maximum du convoi. La vitesse devra être la plus stable possibles, la distance entre chacune des locomotives devra être constante et la longueur du convoi devra être la plus petite possible.

II. Les parties prenantes

1. Maîtrise d'ouvrage stratégique

Le tuteur de projet, monsieur Laurent AUTRIQUE est la seul personne à être amené à décider des grandes orientations durant toutes les périodes du projet.

2. Maîtrise d'ouvrage opérationnelle

Les étudiants d'EI4 AGI, Maxime DAUMAS et Bastien LORON, seront les seuls intervenants sur le projet.

III. Contraintes

1. Contraintes liées à l'environnement

Tous les longs du projet nous utiliserons la salle 211 ainsi que les salles informatique 213 et 217.

2. Contraintes techniques

Tout le long du projet nous devrons utiliser les locomotives de marques ROCO ainsi que les rails de train ROCO. D'autres matériels pourront être utilisés si besoin.

3. Contraintes temporelles

Le projet devra être terminé avant la fin du mois d'Avril 2014. Une soutenance de stage sera organisée pour présenter notre projet.

4. Contraintes de budget

Il n'y a pas de contrainte de budget pour le moment. Cependant, afin de nous placer dans les conditions les plus proches du milieu professionnel, nous souhaitons limiter nos dépenses au maximum, sans pour autant, dégrader l'efficacité du système conçu.

IV. Matériel

Pour ce projet nous disposons :

i. De plusieurs locomotives Roco :

Image 1 : Locomotive jaune

Image 2 : Locomotive SNCF

ii. Une centrale Roco

Image 3 : Centrale Roco

iii. Un arduino Uno

Image 4 : Arduino Uno

iv. Un pont DC motor driver

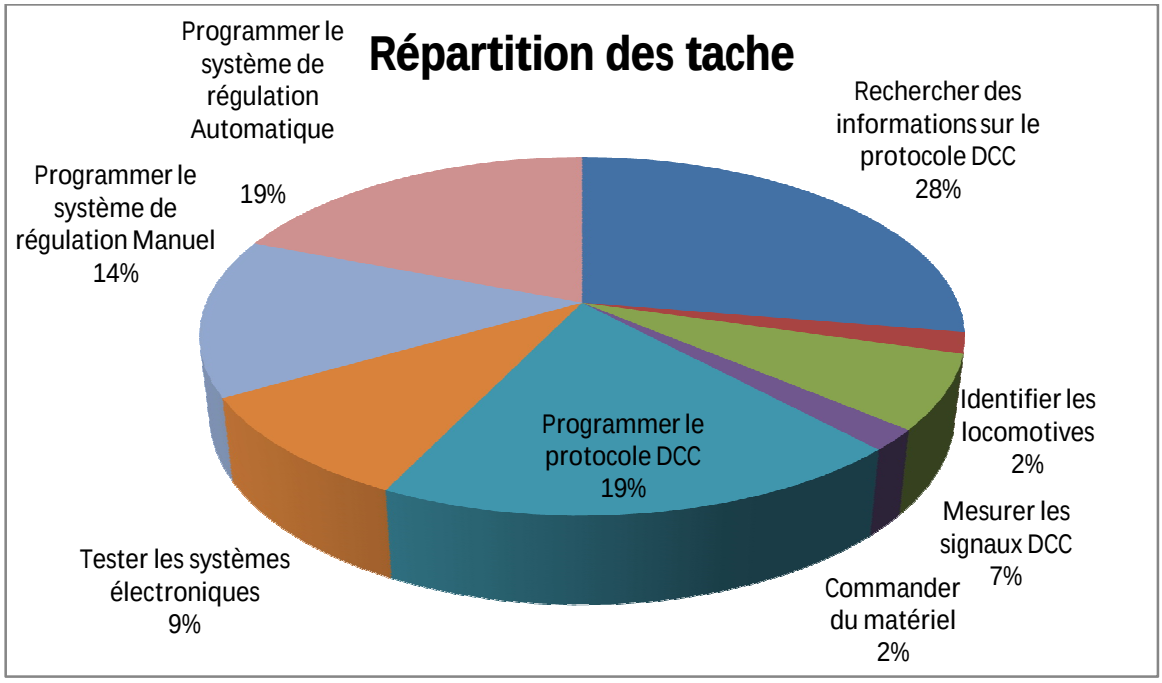
Image 5 : Un pont DC

Daumas Maxime
Bastien Loron

V. Description des tâches principales

Description	Nom de la tâche
Nous avons effectué différente recherche sur le protocole DCC. C'est un protocole qui n'est utilisé que par quelques passionnés.	Rechercher des informations sur le protocole DCC
L'identification des locomotives a été une partie importante de notre projet. Il a fallu effectuer une modélisation mécanique de chacune des locomotives	Identifier les locomotives
Le protocole DCC étant basé sur une variation de fréquence de signaux il a été très important de mesurer ces signaux avant de les recréer.	Mesurer les signaux DCC
Pour créer les signaux DCC nous avons du commander de nouveaux matériaux comme un pont en H	Commander du matériel
Une fois le protocole DCC compris nous avons du le créer à l'aide d'un Arduino et d'un pont en H	Programmer le protocole DCC
Une fois le protocole DCC programmé dans l'Arduino nous avons du tester son efficacité sur la commande des locomotives	Tester les systèmes électroniques
Une fois que le protocole DCC fonctionne correctement nous avons développé un programme en python pouvant utiliser le port série de l'Arduino	Programmer le système de régulation Manuel
Le développement final du système de régulation sera effectué en C #. Ce langage nous permettant d'utiliser plus précisément les objets et le port Série.	Programmer le système de régulation Automatique

VI. Répartition des tache



VII. Digramme de Gantt détaillé

Le diagramme de Gantt détaillé ce trouve en annexe.

Daumas Maxime
Bastien Loron

Projet

I. Fonctionnement général du système

Lors du démarrage du projet nous disposions d'une centrale DCC Roco et de plusieurs locomotives et de plusieurs rails Roco.

La centrale DCC n'étant pas adapté pour notre projet nous avons du la remplacer par notre propre central conçue grâce à un Arduino, une alimentation continu et un pont en H.

Une fois la centrale terminée nous allons développer un ou plusieurs logiciels qui peuvent communiquer avec la centrale pour pouvoir envoyer des informations aux locomotives.

Ces logiciels permettront plus tard d'effectuer des mesures de position réelle des locomotives par le biais d'algorithme ce qui permettra de créer une régulation manuelle de vitesse des locomotives et ensuite une régulation de distance entre chaque locomotives.

Cette régulation permettra de créer des convois de train circulant à une vitesse la plus rapide possible subissant des perturbations telles que le ralentissement d'une locomotive ou encore le retrait d'une locomotive.

II. La commande des trains

1. Digital Command Control

a. Le principe de fonctionnement du système :

L'objectif d'un système de commande digitale est de transmettre des ordres à des récepteurs qui les interprètent et agissent en conséquence.

Pour communiquer, le système de commande utilise un bus de données avec au minimum 2 fils. Pour transmettre les ordres aux locomotives on utilise la voie de chemin de fer.

Les systèmes de commande digitale utilisent la tension servant à l'alimentation des trains pour faire transiter les ordres pour ces mêmes trains. Les signaux d'ordres doivent donc être amplifiés pour pouvoir fournir plus de courant aux locomotives.

Voici les grandes règles qui doivent être absolument respectées :

- L'alimentation du système doit se décomposer en deux parties :
  - Une tension continue, qui sert à fournir la puissance pour permettre aux trains de se déplacer
  - Une tension alternative, qui permet de transmettre les ordres aux trains.
- Chaque locomotive va posséder une adresse, qui doit être unique.
- Il est possible d'avoir plusieurs locomotives sur une même voie non isolée.
- Les accessoires possèdent une adresse qui leur est propre. Les numéros bien que commençant comme les locomotives sont distincts. La commande d'un aiguillage à l'adresse 1 ne commandera pas une locomotive avec l'adresse 1 et réciproquement.

b. Les informations transmises:

Les informations envoyées aux récepteurs sont composées de plusieurs éléments :

Une adresse: pour dire quel récepteur doit tenir compte de l'ordre.

L'ordre: la vitesse, le sens, l'activation des feux, les arrêts, des fonctions personnaliser.

Un dispositif de contrôle: pour éviter la prise en compte d'ordres erronés, en cas de mauvaise transmission un dispositif de contrôle est prévu dans chaque message.

Entre chaque message d'ordre il est prévu l'envoi d'une séquence permettant aux récepteurs ou décodeurs de savoir qu'un nouveau message va commencer, dans cette suite ininterrompue de bits. Il existe aussi un code spécial indiquant la fin du message.

c. Les types de digital

Le digital simple :

Il s'agit de l'approche la plus attractive du digital et aussi la plus simple. Dans cette configuration, la centrale digitale est connectée au réseau avec deux fils. Chaque machine est équipé d'un décodeur et peut circuler en simultané sur votre réseau avec une gestion indépendante de la vitesse.

Le digital complexe :

Cette solution est adaptée à partir d'un réseau de taille moyenne qui peut envisager de faire circuler plus de 5 trains en simultané. Pour nous cette une solution n'est que partiellement satisfaisante. En effet elle nous oblige à utiliser des logiciels propriétaire payant, seulement compatible avec certain décodeur.

III. Trame

1. Alimentation/ Commande

La tension dans les rails n'est pas continue. Cette tension varie donc de 18V à -18V.

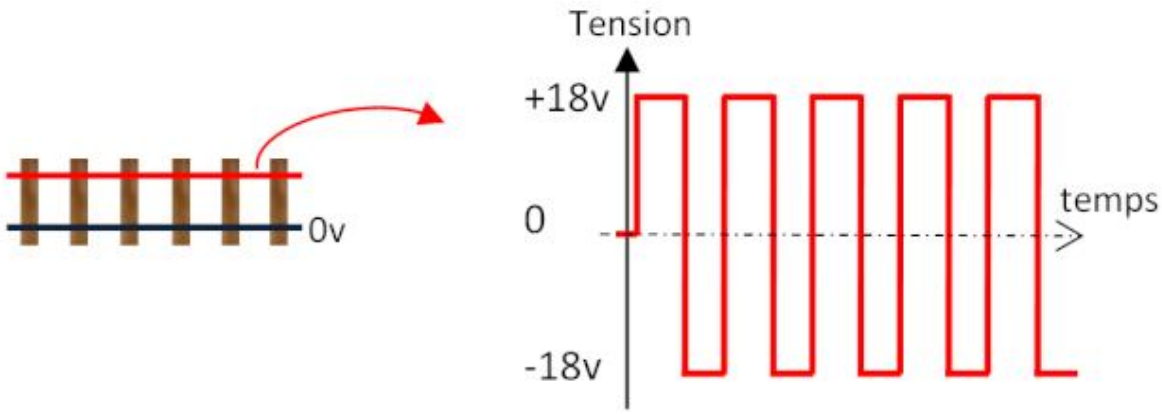


Image 6:Alimentation et Commande

La figure ci-dessus montre qu'un rail reste toujours à la masse tandis que la centrale fait varier la tension sur l'autre rail en envoyant des tensions symétriques (généralement +/- 18V).

Un créneau positif (+18V) est donc immédiatement suivi d'un créneau négatif (-18V) de même durée. Cette durée identique est importante dans au cas où le décodeur serait branché à l'envers. Ce branchement n'a ainsi aucune conséquence sur le fonctionnement du décodeur.

2. Récupération de l'information

Chacun des systèmes compatibles DCC doit posséder un décodeur DCC.

Le décodeur, grâce à des transistors, adapte les créneaux reçus en annulant la tension négative et en diminuant la tension positive.

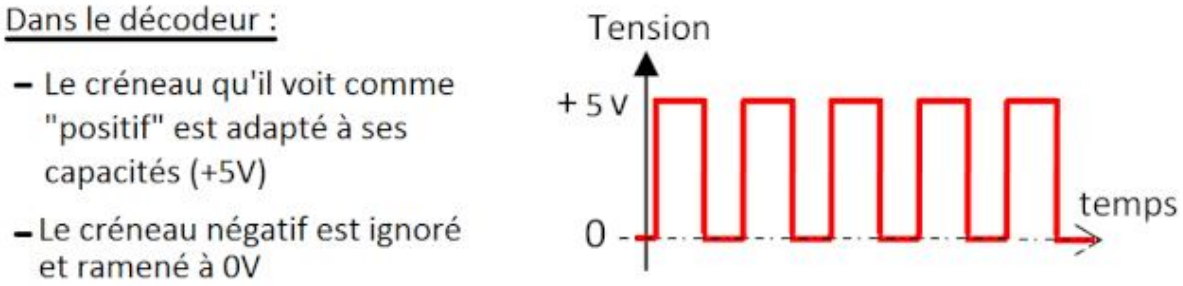


Image 7:Tension d'entré du décodeur

Il mesure ensuite la durée du signal positif et ignorer le créneau nul.

La durée du signal est configurée comme ceci :

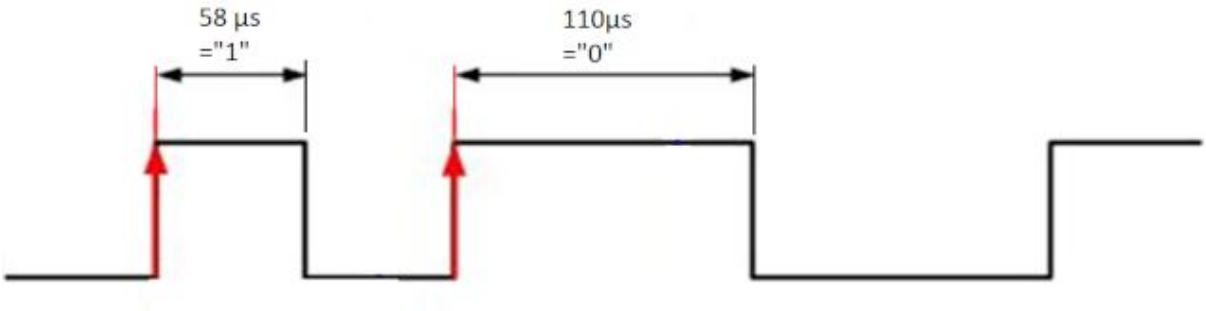


Image 8:Un exemple de signal

- Un créneau court (environ 58 microsecondes) sera interprété comme une information « 1 » par le décodeur.
- Un créneau long (environ 110 microsecondes) sera interprété comme une information « 0 » par le décodeur.

Ces informations « 1 » ou « 0 » seront par la suite utilisées comme des bits. Evidemment, un seul bit (« 1 » ou « 0 ») n'est pas suffisant pour obtenir tous les ordres que nous souhaitons. C'est pourquoi la centrale envoie toute une série de bits consécutifs. C'est ce qu'on peut appeler une trame.

3. La trame

Toutes les trames obéissent à la même structure pour que la centrale et le décodeur puissent se comprendre. Les informations arrivent donc dans un ordre précis sous forme d'*octets* de 8 bits. La trame pourra être constituée de 3 à 6 octets au maximum.

Constitution de la trame :

1: Le préambule : il permet de synchroniser la centrale et le décodeur. Ainsi, un décodeur perdu au milieu d'une trame va se contenter d'attendre le début de trame suivant. Il attend donc le préambule. Il est constitué de 10 bits à 1. En pratique, ce nombre de bits varie entre 11 et 17 suivant les centrales.

2: Bit de START : 0 Il est constitué de un bits à 0, on l'appelle aussi le bit de séparation, il est situé entre deux octets consécutifs.

3: Octet d'adresse : Il permet de savoir à qui s'adresse la centrale. Evidemment, tout décodeur ne correspondant pas à cette adresse ignorera tout simplement le reste de la trame.

4: Bit de START : 0 Il est constitué de un bits à 0, on l'appelle aussi le bit de séparation, il est situé entre deux octets consécutifs.

5: Octet de donnée : Il contient l'ordre à exécuter, cela peut correspondre à une instruction de vitesse ou une fonction (allumage des lumières …). Un décodeur peut posséder des instructions spécifiques.

3: Bit de START : 0 Il est constitué de un bits à 0, on l'appelle aussi le bit de séparation, il est situé entre deux octets consécutifs.

7: Octet de vérification : Il sert à vérifier que toute la trame a correctement été transmise. Il est **calculé** séparément par la centrale d'une part et par le décodeur d'autre part. Si leurs résultats concordent, alors la trame est validée et on peut exécuter l'ordre. Sinon, la trame est rejetée.

8: Bit de STOP : 1 La centrale envoi un bit à 1 pour signaler que le précédent octet était le dernier (celui de vérification).

4. Codage des données

a. L'octet d'adresse :

L'adresse d'une locomotive peut dépasser 128, pour cela il suffit de rajouter un octet comme nous l'avons dit plus haut, il peut y avoir jusqu'à 6 octets en tout. Cette adresse est codée en binaire non réfléchi en partant du principe que l'adresse 0 est un broadcast qui s'adresse à toutes les locomotives en même temps.

b. L'octet de données

Il est composé de 8 bits: Il peut exister deux types d'Octet de donnée, les fonctions globales et les fonctions vitesse.

V4	V3	V2	V1	V0	S	F2	F1
0	0	0	0	1	0	0	1

Pour coder une fonction globale il suffit de mettre le premier bit à 1 et le second à 0. Pour ensuite utiliser par exemple la Fonction 1 de la locomotive qui correspond à « allumer les lumières » il suffit de mettre le bit V0 à 1.Par la suite pour les autres fonctions elles sont codées en binaire sur les bits V0 à V4.

Dans le mode de fonction globale le bit S n'est pas utilisé.

Pour coder une fonction vitesse il suffit de mettre le bit F1 à 0 et le bit F2 à 1. Pour faire varier le sens de circulation de la locomotive il suffit de changer la valeur du bit S.

Pour les vitesses si l'on souhaite utiliser 14 paliers de vitesse pour les locomotives il faut mettre le bit V0 à 0 sinon par défaut le nombre de palier d'une locomotive est de 28. Il est aussi possible d'avoir 128 paliers de vitesse mais toutes les locomotives ne sont pas compatibles. Dans note cas nous utiliserons le mode avec 14 paliers de vitesse.

Le codage des bits de vitesse V0 à V4.

V4	V3	V2	V1	V0	Vitesse
0	0	0	0	0	Arrêt
1	0	0	0	0	Urgence
0	1	0	0	0	1
1	1	0	0	0	2
0	0	1	0	0	3
1	0	1	0	0	4
0	1	1	0	0	5
1	1	1	0	0	6
0	0	0	1	0	7
1	0	0	1	0	8
0	1	0	1	0	9
1	1	0	1	0	10
0	0	1	1	0	11
1	0	1	1	0	12
0	1	1	1	0	13
1	1	1	1	0	14

c. L'octet de vérification

D'âpres la documentation DCC c'est une opération OU EXCLUSIF « bit-à-bit » entre deux ou plusieurs octets. Ainsi, pour 4 octets consécutifs, on a le schéma suivant :

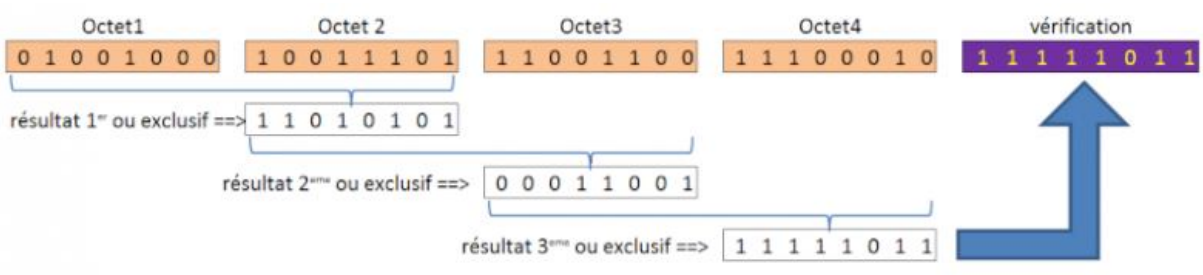


Image 9 : Calcule d'octet

d. Une trame complète

Une trame complète est donc composée d'un préambule et de trois octets, si nous donnons un exemple:

Une locomotive d'adresse 1, feu allumé, doit circuler dans le sens 1 avec une vitesse 12.

Préambule :

1	1	1	1	1	1	1	1	1	1	1

Bit de Start

0

Octet d'adresse :

0	0	0	0	0	0	0	1

Bit de Start

0

Octet de donnée:

0	0	0	0	0	1	0	1

Octet de donnée:

1	0	1	1	0	1	1	0

Octet de vérification:

0	1	0	1	1	0	0	1

Bit de stop

1

Daumas Maxime
Bastien Loron

IV. Systèmes électroniques

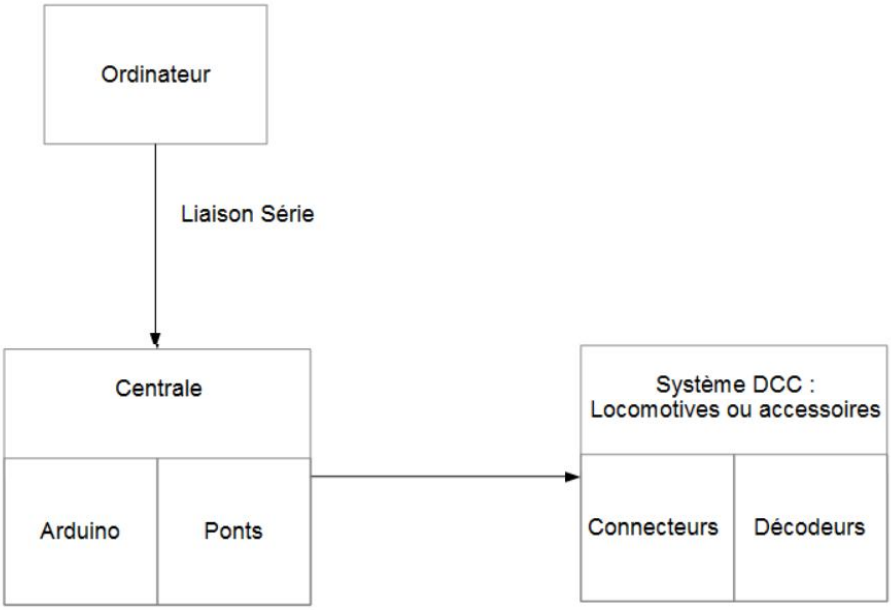


Image 10 : Systèmes électroniques

Pour créer les trames constitué de bit 0 et 1 nous avons besoin de générer un signal alternatif de 18V. Pour créer ce signal nous avons eu besoin d'un microcontrôleur, d'un pont en H et d'un ordinateur.

Le rôle du microcontrôleur est de communiquer avec l'ordinateur pour ainsi recevoir des informations, commander le pont en H pour ainsi transmettre les informations sous forme de signal DCC.

Image 11 : Arduino Uno

Le microcontrôleur que nous avons choisi pour réaliser cette tâche est un Arduino Uno. Sa simplicité d'utilisation et nos précédentes expériences ont été décisives lors de ce choix.

Bien que l'Istia ait beaucoup de pont en réserve, nous avons choisi d'acheter un nouveau pont en H DFRobot qui permettra d'utiliser au plus dix locomotives simultanément avec une tension de 18V 15 A maximum tout en étant protégé électriquement.

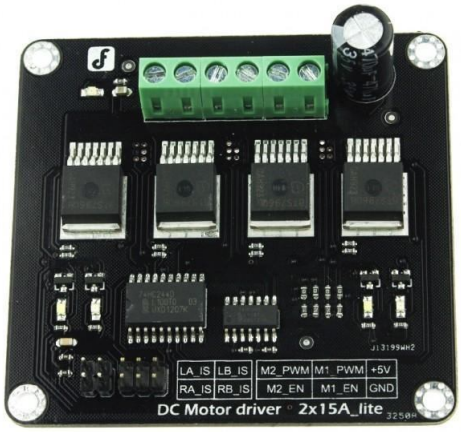


Image 12 : Pont DCC

1. Schéma de câblage de la centrale

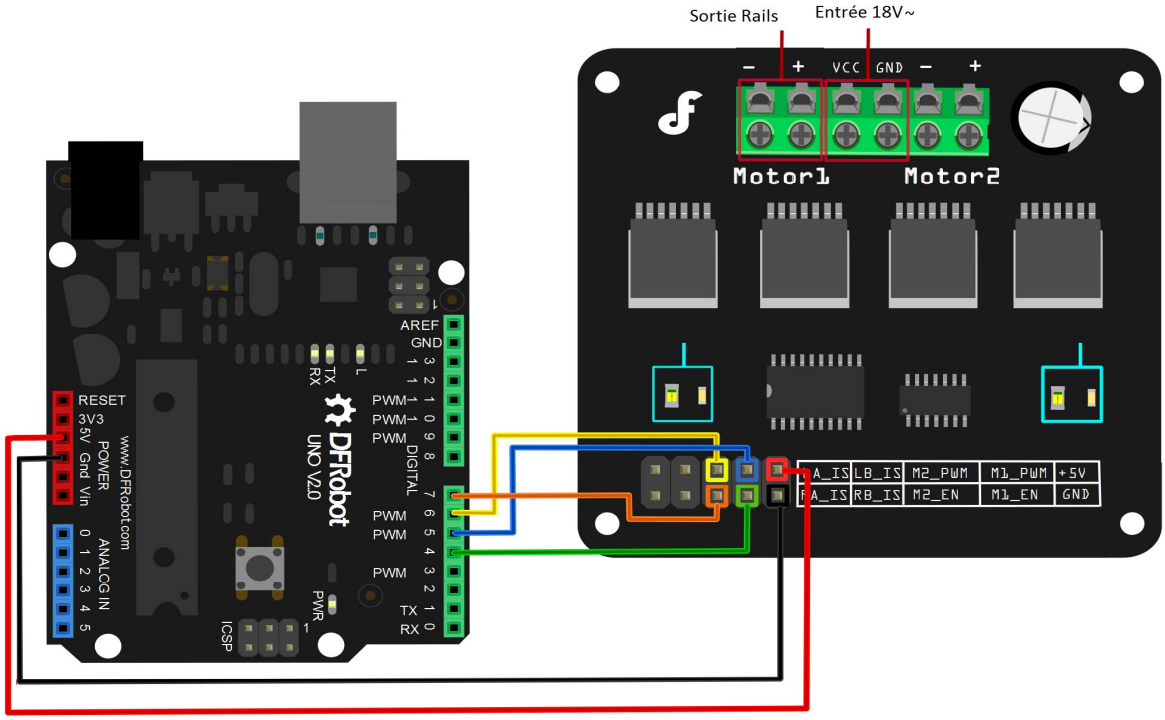


Image 13 : Câblage de la centrale

Le principe de fonctionnement du pont en H est le suivant. On utilise le pont en activant les 4 transistors de différente façon pour obtenir le branchement voulu.

Le tableau suivant résume les combinaisons :

Etat	Etat des commutateurs				Résultats
	S1	S2	S3	S4	
1	F	F	F	F	Aucune tension
2	O	F	F	O	Tension positive sur la charge
3	F	O	O	F	Tension négative sur la charge
3	O	O	F	F	Charge court-circuitée.
4	F	F	O	O	Charge court-circuitée.

L'activation des transistors a déjà été protégée par le constructeur. Pour commander le pont on utilise le bit de contrôle appelé X_EN on peut alors atteindre les états 2 et 3 du tableau.

En effectuant ainsi un changement d'état rapide du pont on obtient une tension alternative aux bornes de la charge.

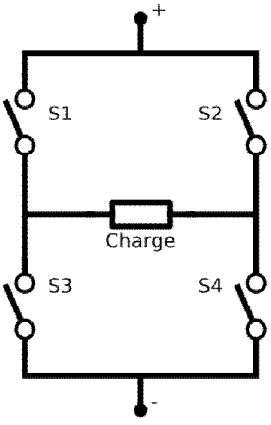


Image 14: Pont en H

Daumas Maxime
Bastien Loron

2. Code Arduino

Comme tous les programmes destinés à l'Arduino, les fonctions *setup* et *loop* ont été définies. Nous utilisons la fonction *setup* configurer la carte (port série, entrées/sorties). La fonction *loop* contient l'algorithme que le microcontrôleur va exécuter en boucle. Dans un premier temps, l'Arduino récupère les informations reçues via le port série.

Ensuite, on construit la trame à partir des dernières informations reçues puis la trame est envoyé en respectant la norme DCC.

Plusieurs macros sont définies afin de pouvoir modifier rapidement certains paramètres du programme (pins, longueurs des en-têtes et fin de trame).

Enfin, le code est divisé en fonctions simples, ceci évite les redondances, facilite la maintenance, augmente la lisibilité et la compréhension du code.

La code a été commenté et une documentation Doxygen a également été généré afin de faciliter la reprise du code.

3. Communication PC - Arduino

Afin de faire d'envoyer les instructions à l'Arduino, une interface a du être mise en place. Plusieurs options s'offraient à nous:

- le port-série, disponible de base sur l'arduino
- l'Ethernet, disponible en ajoutant une carte d'extension

Image 15 : Carte Arduino

Afin de minimiser les coûts et de simplifier la programmation, nous avons retenu la solution du port série. Nous reviendrons plus tard sur les inconvénients de cette solution.

Cette liaison série permet de communiquer à l'Arduino, l'adresse de la locomotive à laquelle il doit s'adresser, le sens et la vitesse à laquelle il doit la configurer. Pour séparer les différentes composantes du message, nous utilisons un caractère autre que numérique (ponctuation, lettre ...).

Voici la fonction qui décode les informations envoyé sur le port série

```
void SerialRead()
{
        if (Serial.available() > 0)
        {
                digitalWrite(pwmA, LOW);

                loco.adresse = Serial.parseInt();
                if (Serial.parseInt() == 0)
                {
                        loco.sens = true;
                }
                else
                {
                        loco.sens = false;
                }
                loco.vitesse = Serial.parseInt();
                Serial.parseInt();
                digitalWrite(pwmA, HIGH);
        }
}
```

Quand la fonction est appelée nous vérifions si le buffer du port série contient quelque-chose.

Si c'est le cas, nous récupérons les trois informations contenues dans le message.

Dans le cas contraire, on renvoie une nouvelle fois la même commande que précédemment.

Image 16 : Fonction lecture

Pendant la lecture de ces données, nous stoppons l'alimentation du pont. L'Arduino n'étant pas multitâche, il n'est pas possible de commander le pont et de lire le port série en même temps. Cette opération permet d'éviter le passage des locomotives en mode analogique provoquant ainsi des accélérations et des changements de sens non souhaités.

V. Le système de mesure

Le protocole DCC utilisé par les locomotives est un système sans retour d'état. En effet il ne retourne aucune information.

Le rôle du système de mesure sera donc de calcules la vitesse des locomotives, la position des locomotives sur les rails à tous instant et calculer ainsi la distance entrer chaque locomotives.

1. Identification

Dans un premier temps nous avons du effectuer « l'identification du système ». Cette identification nous a permis d'obtenir des modèles mathématiques et les hypothèses qui caractériseront le fonctionnement des locomotives et le système en général.

a. Le système global

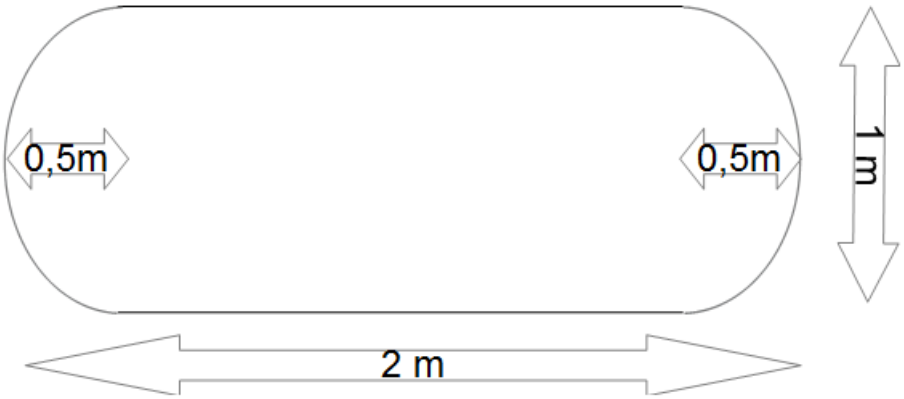


Image 17:Le système

Pour la suite pour modélisation nous prendrons comme hypothèse :

- Il n'y a pas de perturbation extérieure (élément sur les voies, vents, …).
- La vitesse des locomotives reste linéaire dans les virages.
- La distance totale périphérique des rails (le rail extérieur) est de 8.28 mètres.

b. La locomotive

Pour la suite nous prendrons comme hypothèse pour l'identification des locomotives:

- Pas de perturbation extérieure (élément sur les voies, vents, …).
- Pas de ralentissement dans les virages causé par l'effet centrifuge.
- Une locomotive est capable de garder une vitesse stable si elle ne reçoit pas de nouvelle commande.

c. Calcul de vitesse réelle

Dans un premier temps nous utiliserons une première méthode pour récupérer la vitesse réelle de la locomotive en fonction de la commande. Pour cela nous avons utilisé un chronomètre et le système de commandes manuel des locomotives. Pour effectuer une mesure de vitesse nous effectuons ce protocole :

- dans un premier temps nous envoyons une commande de vitesse à la locomotive (1 à 14)
- nous laissons faire au moins un tour à la locomotive pour qu'elle prenne de la vitesse
- au bout de ce tour nous lançons le chronomètre à un endroit marqué
- nous attendons que la locomotive fasse au moins 5 tours.
- puis au dernier tour nous récupérons le temps écoulé depuis le départ

Pour obtenir la vitesse réelle de la locomotive il faut ensuite effectuer le produit de la distance total périphérique des rails et du nombre de tour divisé par le temps écoulé depuis le départ.

Nous avons effectué cette méthodes plusieurs fois pour chaque commande et locomotives et avons obtenus des caractéristiques.

d. Calcul de la distance d'arrêt du train

Nous allons utiliser une nouvelle méthode pour récupérer la distance d'arrêt de la locomotive en fonction de la commande.

Pour cela nous avons utilisé une grande règle et le système de commandes manuel des locomotives.

Pour effectuer une mesure de la distance d'arrêt nous effectuons ce protocole :

- dans un premier temps nous envoyons une commande de vitesse à la locomotive (1-14)
- nous laissons faire au moins un tour à la locomotive histoire qu'elle prenne de la vitesse
- au bout de ce tour nous choisissons un endroit marqué ou arrêter la locomotive
- nous attendons que la locomotive arrive au point d'arrêt
- puis nous envoyons une commande d'arrêt
- nous mesurons ensuite la distance parcourue par la locomotive

Nous avons effectué cette méthodes plusieurs fois pour chaque commande et locomotives et avons obtenus des caractéristiques.

e. Calcule de position de la locomotive

Nous allons utiliser une nouvelle méthode pour récupérer la position du train à chaque instant en fonction de la vitesse et de la distance d'arrêt.

Pour cela nous donnerons un exemple:

Nous prendrons une locomotive Jaune à l'arrêt qui démarre à une commande de dix soit une vitesse de 0.84m/s à une position initial de 0. Au bout de 2s on effectuera un changement de vitesse.

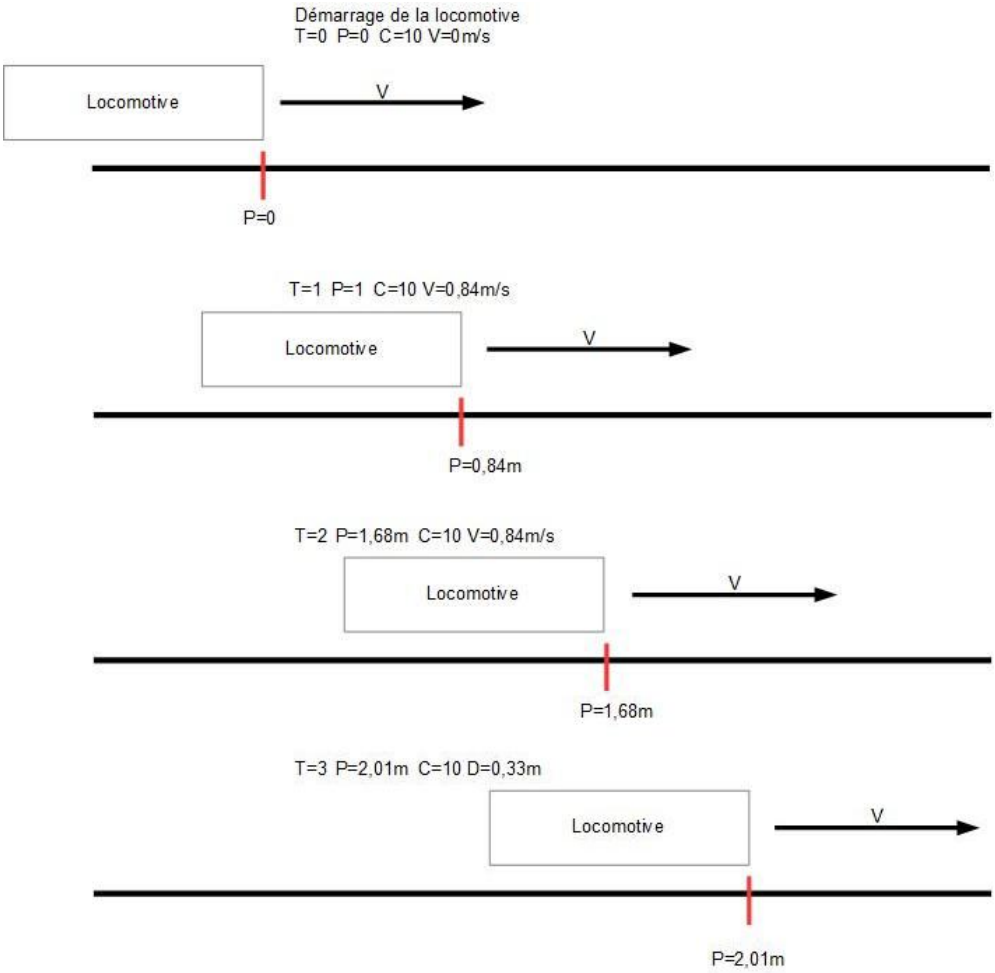


Image 18 : Calcule de position

La position sera donc calculée de la façon suivante:

$$PositionLoco = (\text{Vitesse} \times \text{temps de fonctionnement}) + \text{positionInitialLoco} + \text{distance de freinageLoco}$$

2. Résultats

i. Pour la locomotive jaune :

- Pour la caractéristique de la vitesse réelle nous obtenons une fonction puissance :

$$\mathrm{VitesseReel(Commande)} = 0.04852 \times \mathrm{Commande}^{1.230}$$

Avec 3.7% d'erreur sur la mesure et un écart type de 28.81 mm

- Pour la caractéristique de la décélération nous obtenons une fonction puissance :

$$\mathrm{DistanceArret(Commande)} = 0.3551 \times \mathrm{Commande}^{3.098}$$

ii. Pour la locomotive SNCF :

- Pour la caractéristique de la vitesse réelle nous obtenons une fonction puissance :

$$\mathrm{VitesseReel(Commande)} = 0.06302 \times \mathrm{Commande}^{1.214}$$

Avec 5% d'erreur sur la mesure et un écart type de 60 mm

- Pour la caractéristique de la décélération nous obtenons une fonction puissance :

$$\mathrm{DistanceArret((Commande)} = 0.0225 \times \mathrm{Commande}^{1.1}$$

Après l'identification nous pouvons dire que la locomotive jaune a une plus grande vitesse maximum que la locomotive SNCF mais celle-ci dispose d'une accélération et une décélération plus faible.

La vitesse moyenne d'une locomotive est de 0.86m/s avec une distance de freinage de 32cm

VI. La régulation

Dans ce dispositif, le procédé de régulation aura pour rôle de garder une distance égale entre chaque locomotive et ainsi pouvoir créer un système de convoi.

Nous définissons les trois grandeurs physiques du système :

- La grandeur réglée: la distance entre chaque locomotive
- La grandeur réglante: la vitesse des locomotives, la position de chaque locomotive
- La grandeur perturbatrice: variation de vitesse, temps de réponse du système, variation de tension d'alimentation, ralentissement d'une locomotive

1. Régulation manuelle

Le but de la régulation manuelle est de permettre à un opérateur de pouvoir commander séparément chacune des locomotives. Pour cela nous avons développé un logiciel en python qui est capable de vérifier la présence d'un Arduino, de choisir le bon port COM sur lequel écrire et ainsi d'envoyer des données à l'Arduino.

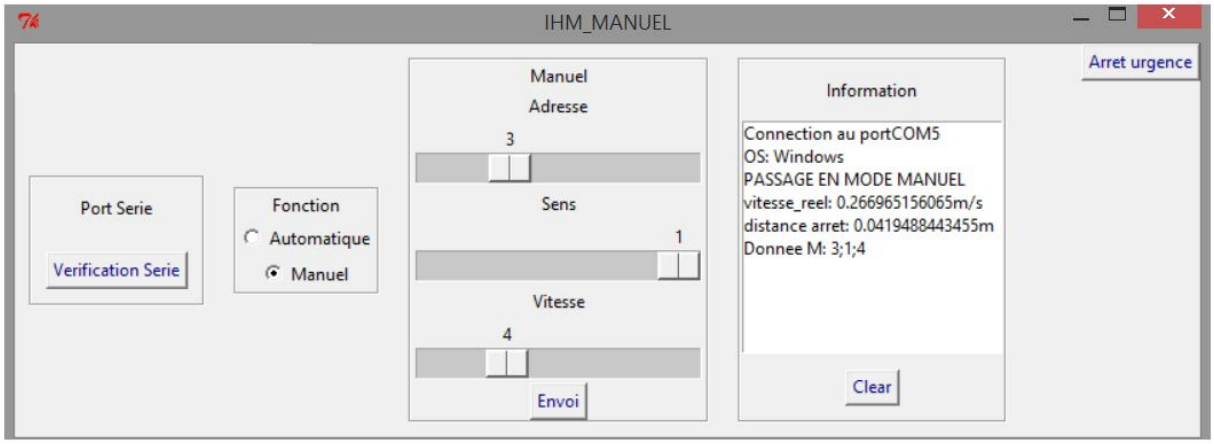


Image 19 : Mode Manuel

Dans ce programme nous utilisons les librairies Tkinter pour l'interface graphique et PySerial pour la gestion du port Séries

Nous avons décidé d'effectuer le mode manuel en python pour sa simplicité de codage et le choix de ses bibliothèques qui sont adapté à l'Arduino.

2. Régulation Automatique

Dans ce cas, la mesure de la grandeur réglée et la modification de la grandeur réglante s'effectuent automatiquement au moyen d'une loi de commande. Il n'y a donc plus d'intervention d'un opérateur.

a. Chaîne de régulation globale

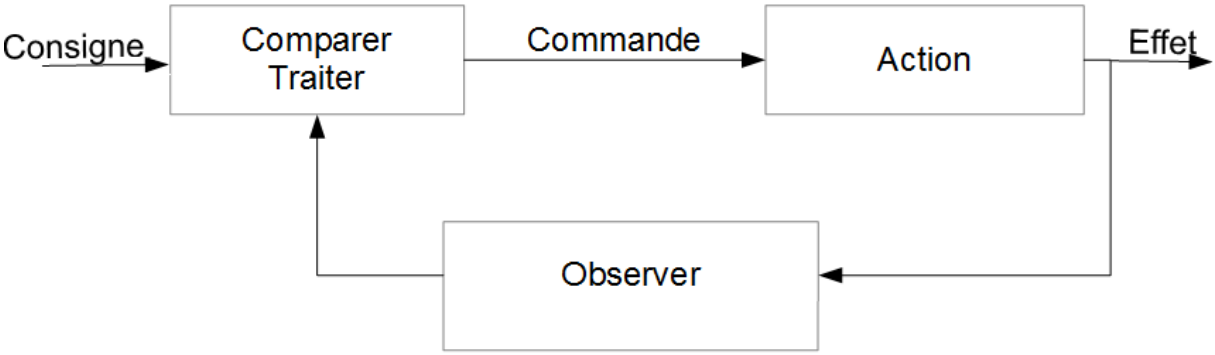


Image 20 : La chaine de mesure

b. Consigne de régulation

Dans cette régulation, on mesure d'une manière continue la grandeur réglée et nous devons la comparer nos consignes de régulation. Quand on détectera un écart entre elles, on doit produire une action appropriée qui doit ramener la grandeur réglée en accord avec la consigne de régulation.

Cette consigne de régulation sera calculée en fonction de la distance de freinage de chaque locomotive. Il y aura donc une consigne de distance différente pour chaque locomotive.

$$\text{Consigne minimum} = \text{distance d'arrêt de la locomotive en aval} + (\text{temps de réponse} * \text{vitesse reel})$$

$$\text{Consigne maximum} = 70\text{cm}$$

$$\text{Consigne Conseillé} = 4\text{cm}$$

La consigne maximum n'a pas été choisie au hasard en effet la taille moyenne d'une locomotive est de 25cm, si jamais nous faisons fonctionner huit locomotives en même temps cela nous donnerai 8*(25cm+60cm)=6,80 mètres de convoi total.

La consigne conseillée sera utilisée lorsque la mesure dépassera la consigne maximum.

Ces consignes permettent lors d'une nouvelle commande ou d'un arrêt d'éviter les collisions entre les locomotives.

c. Stratégie de régulation

Le système dispose d'une commande par palier, il est donc impossible d'obtenir une valeur de vitesse fixe.

Nous avons alors choisi d'utiliser une régulation par hystérésis à 2 seuils.

Le principe de cette régulation est qu'au lieu de définir une seul valeur de consigne le système va calculer un intervalle (consigne maximum et minimum) dans lequel la mesure doit rester. Si jamais la mesure sort de cet intervalle la régulation ramène la mesure jusqu'à la valeur de consigne conseillé.

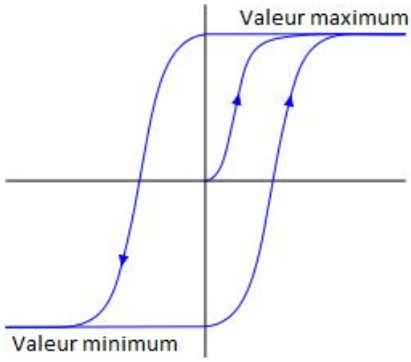


Image 21 : Hystérésis

3. Comportement de la régulation

a. Régulation de distance

La vitesse lors d'une régulation automatique la régulation de distance possède ces comportements:

- Lorsque la mesure m augmente, la distance h entre les deux locomotives augmente. Si jamais h dépasse la consigne maximum, la régulation diminue la vitesse de la locomotive en aval jusqu'a ce que h soit inferieur à la consigne conseillée.

- Lorsque la mesure m diminue, la distance h entre les deux locomotives diminue. Si jamais h dépasse la consigne minimum, la régulation augmente la vitesse de la locomotive en aval jusqu'a ce que h soit supérieur à la consigne conseillée.

- Lorsque la mesure m reste dans les intervalles de consigne la régulation ne commande plus les locomotives qui gardent une vitesse stable.

b. Gestion de convoi

En partant du principe que la régulation de distance est fonctionnelle et que la consigne de vitesse de la locomotive de tête est donnée par l'opérateur.

Si nous donnons un exemple avec un convoi composé de trois locomotives. La distance entre les trois locomotives sont les mêmes. La locomotive A étant à l'arrêt, la distance DA-B ne varie pas et donc les locomotives B et C restent à l'arrêt.

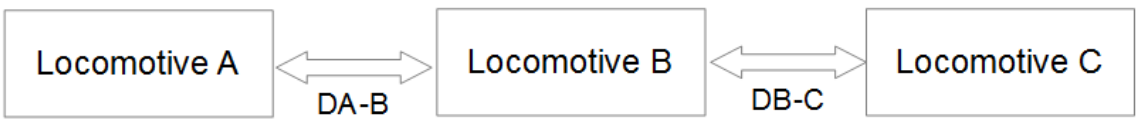


Image 22 : Gestion de convoi

Lors du démarrage de la locomotive A ou d'une augmentation de la vitesse de la locomotive de tête la distance DA-B augmente jusqu'à atteindre la consigne maximum. Lorsque la consigne maximum est atteinte la Locomotive B va accélérer jusqu'à atteindre la consigne conseillée. La distance DB-C va donc augmenter jusqu'à dépasser la consigne maximum. Une fois atteinte la Locomotive C va accélérer jusqu'à atteindre la consigne conseillée.

Lors d'une grosse diminution de la vitesse de la locomotive de tête la distance DA-B va diminuer fortement jusqu'à atteindre la consigne minimale. Si jamais DA-B descend en dessous de la consigne minimale, la Locomotive B va diminuer sa vitesse jusqu'à obtenir la consigne conseillée. Si la vitesse de la Locomotive B diminue la distance DB-C va diminuer fortement jusqu'à atteindre la consigne minimale. La Locomotive C va alors réduire sa vitesse jusqu'à obtenir la consigne conseillée.

4. Chaine de régulation détaillée

Nous partons du principe que nous avons que deux locomotives, dans cette exemple la locomotives de tête est la L1.

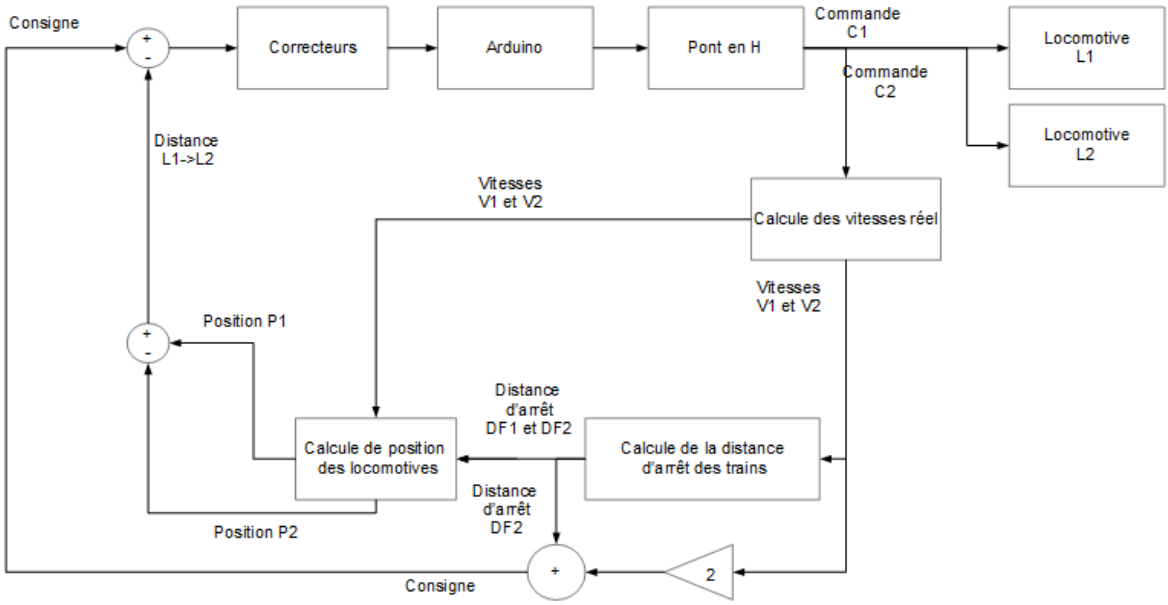


Image 23 : Régulation

VII. Programmation

1. Choix du langage de programmation

Afin de communiquer les instructions à l'Arduino, nous avons choisi de développer un second logiciel destiné à être exécuté sur PC. Celui-ci reprend les fonctionnalités du précédent en gérant en intégrant la notion de convois et en mettant en place la régulation.

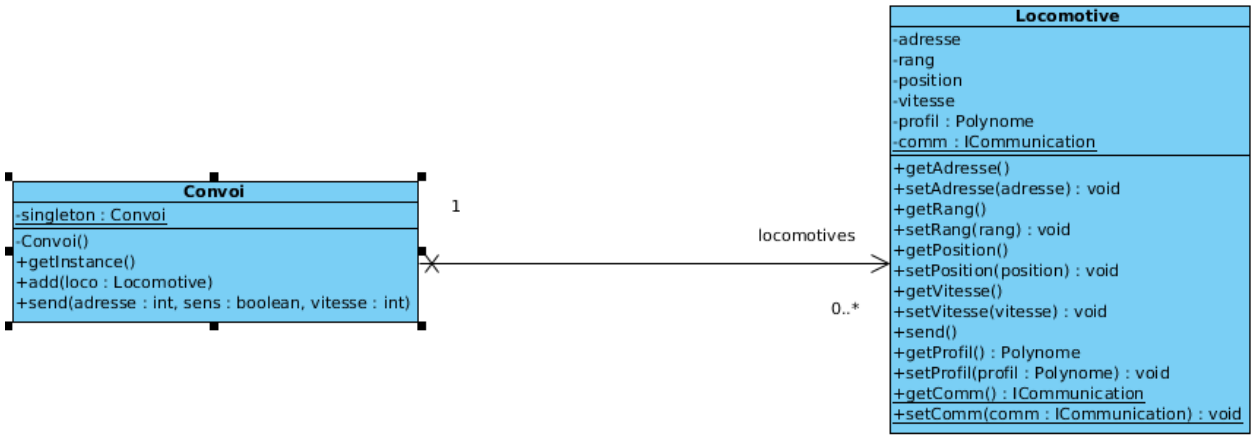
Plusieurs langages de programmations s'offraient à nous pour effectuer cette tâche.

- Java :
  Disponible sur tous les systèmes d'exploitation, ce langage est le plus adapté pour développer un logiciel multi-plateforme. Cependant, Java est un langage interprété nécessitant l'environnement d'exécution java. Cette interprétation nécessite du temps, le rendant moins efficace qu'un langage compilé. Dans notre cas, nous écart cette option car la gestion des ports série n'est pas prise en charge nativement.

- C++:
  Comme le Java, le C++ est un langage qui est disponible sur toutes les plateformes, cependant, il s'agit d'un langage compilé. Il est donc impossible de créer un exécutable unique prenant en charge toutes les plateformes et architectures (environnement 32 et 64 bits). Enfin, la gestion du port série n'existe pas dans la STL (Standard Template Library) mais celle-ci est implémenté par certains frameworks (Qt par exemple).

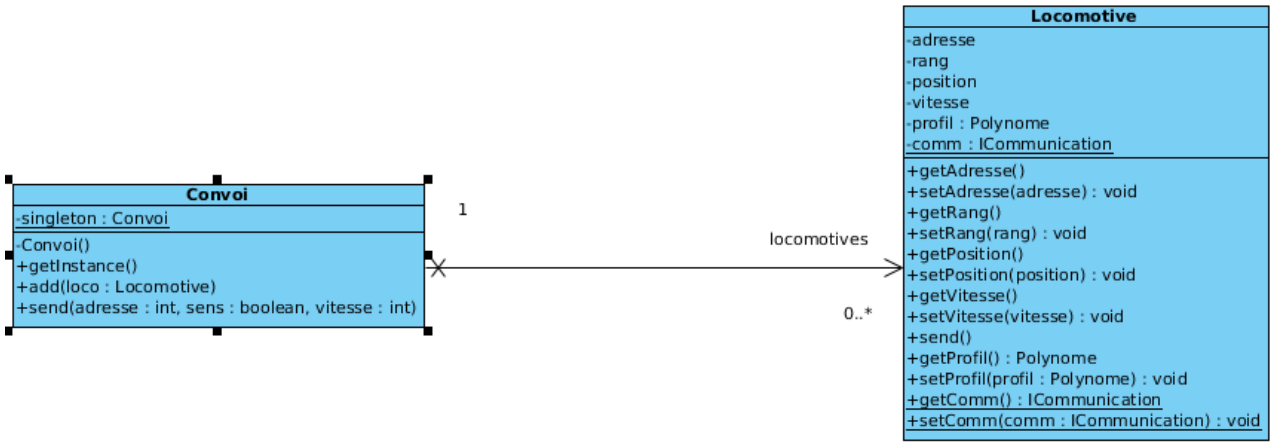
- Les langages .NET (Visual Basic et C#):
  Disponible uniquement sur Windows, les langages .NET nécessite un environnement de développement spécifique développé par Microsoft, Visual Studio. Tout comme le java, les langages .NET ne sont pas compilés, mais interprétés par le framework Microsoft .NET. Ceux-ci présentent l'avantage de prendre en charge la communication série, cette raison nous a convaincu de choisir cette technologie.

2. Structure système UML



Notre système ne comporte qu'un seul convoi. La classe Convoi a donc été défini selon le design pattern singleton (instanciation unique). Les locomotives sont simplement stockées dans un conteneur.

En plus de ces attributs propres, la classe Locomotive contient également un élément statique (comm) qui lui permet de communiquer les instructions à l'Arduino au travers du port série.



Une interface ICommunication a été définie afin de faciliter les évolutions futures du logiciel. Actuellement, nous utilisons le port série mais nous pourrions envisager un changement du protocole de communication pour une interface de type IP (Ethernet ou Wifi) moyennant quelques aménagements sur l'Arduino (ajout d'une carte réseau et mise à jour des sources pour prendre en charge le nouveau bus de communication).

Actuellement, le mode de fonctionnement automatique via la mise en place d'une régulation ne fonctionne pas bien.

VIII. Amélioration possible

1. Webcam

Ce système est basé sur l'utilisation d'une webcam ainsi que de pastilles de couleurs.

Nous utilisons une webcam de résolution 1080p. Pour la programmation du système nous utilisons la librairie Open CV. Le principe de la mesure est de reconnaître les couleurs dans l'espace de la maquette. Chaque locomotive dispose d'une pastille de couleur différente. Le programme devra être en mesure d'identifier la couleur de chacune des pastilles et ainsi identifier chacune des locomotives. La position des locomotives sera par la suite calculée en mètre selon la position de la pastille de couleurs sur la maquette. Une fois la position calculée, il sera très simple de calculer la vitesse des locomotives ainsi que leurs accélérations.

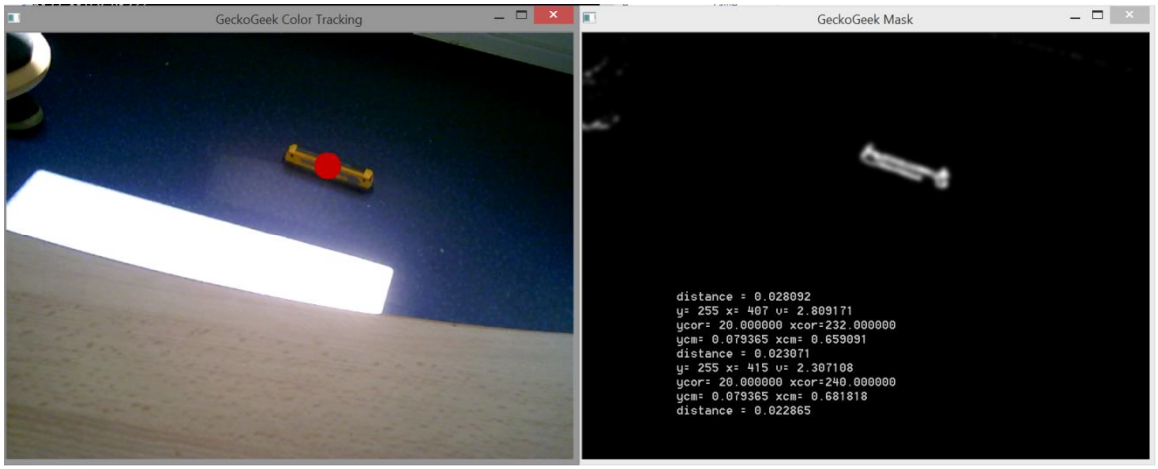


Image 24 : Démonstration de webcam

Ce système a été développé en python et en C++ mais suite à diffèrent essais la stabilité de la mesure n'était pas satisfaisante et comportait beaucoup d'erreur suite à des problèmes d'exposition de lumière.

Ce système aurai pu être utilisé dans une détection à distance des locomotives sur les rails pour associer directement au programme les locomotives déjà présentes.

2. Raspberry

Le Raspberry Pi est un nano-ordinateur monocarte conçu par le créateur de jeux vidéo David Braben.

Cet ordinateur est destiné à encourager l'apprentissage de la programmation informatique. Il est cependant suffisamment ouvert (ports USB voir réseau) et puissant (ARM 700 MHz, 256 Mo de mémoire vive pour le modèle d'origine, 512 Mo sur les dernières versions) pour permettre une grande palette d'utilisations.
Il embarque son propre système d'exploitation (souvent basé sur des distributions Linux aménagées pour s'adapter à la configuration de la carte) et son chipset graphique lui permet d'être branché directement sur un écran, comme pourrait l'être un ordinateur standard.

Image 25 : Raspberry pi

Nous avons envisagé son utilisation au début du projet en lieu et place de l'Arduino Uno.
La commande du pont en H nécessitant une précision à quelques microsecondes près, les systèmes d'exploitation standards ne répondaient pas à notre cahier des charges.

Une solution existe pour s'affranchir de ce problème, l'utilisation de noyaux temps réel. Xenomai est l'un d'en eux. N'ayant aucune expérience sur de tels systèmes (qui demandent du temps pour les maîtriser), nous avons choisi d'utiliser un Arduino Uno.

Aujourd'hui, la question se pose à nouveau pour faire évoluer le projet. Celui-ci nous permettrait de supprimer les perturbations que l'on peut observer actuellement lors de la lecture du contenu de port série.

IX. Conclusion

Les objectifs que nous nous étions fixés au début du projet étaient très ambitieux, et nous n'avons pas réussi à les atteindre en totalité. Toutefois même si nous sommes déçus pour nous cette expérience est positive.

En effet, ce projet nous a, tout d'abord, permis de réutiliser nos connaissances acquises au cours de notre cursus. L'axe que nous avons suivi au cours de ce projet nous a permis de mettre en place notre savoir faire dans un projet concret et sur une durée de plusieurs mois. Ces 5 mois de projet à l'ISTIA ont également été l'occasion de découvrir de nouvelles technologies.

Ensuite, il nous a permis d'expérimenter et de comprendre l'intérêt du travail en groupe.

Venant chacun d'horizons divers, cette différence nous a permis d'aborder le travail plus efficacement, chacun complétant l'autre et apportant une autre approche des difficultés.

Enfin, ce projet nous a permis d'acquérir et d'exploiter des méthodes de travail usitées dans le monde professionnel (outils de versionning, tests unitaires, optimisation du type des variables et division du code en fonctions simples facilement maintenables).

Bibliographie

Opencv [En ligne] Itseez, 2014 [consulté le 12 decembre 2013]. Disponible sur :

http://opencv.org

Tkinter[En ligne] Python Software Foundation, 2014 [consulté le 1 janvier 2014]. Disponible sur :

https://docs.python.org/2/library/tkinter.html

Tkinter[En ligne] Python Software Foundation, 2014 [consulté le 1 janvier 2014]. Disponible sur :

https://docs.python.org/2/library/tkinter.html

Tutorialspoint[En ligne] Tutorialspoint, 2014 [consulté le 15 decembre 2013]

Disponible sur :

http://www.tutorialspoint.com/python/python_gui_programming.htm

GoTronic[En ligne] GoTronic, 2014 [consulté le 15 janvier 2014]. Disponible sur :

http://www.gotronic.fr/

Alain.mionnet[En ligne] Alain mionnet, 2014 [consulté le 23 decembre 2013]. Disponible sur :

http://alain.mionnet.pagesperso-orange.fr/lrdigtech.htm

Opendcc[En ligne] Wolfgang Kufer, 2014 [consulté le 10 janvier 2014] .

Disponible sur :

http://www.opendcc.de/index_f.shtml

Train35[En ligne] Cercle Ferroviphile, 2014 [consulté le 08 janvier 2014]. Disponible sur :

http://www.train35.com/

Benoit.bouchez[En ligne] Benoit bouchez, 2014 [consulté le 08 janvier 2014] . Disponible sur :

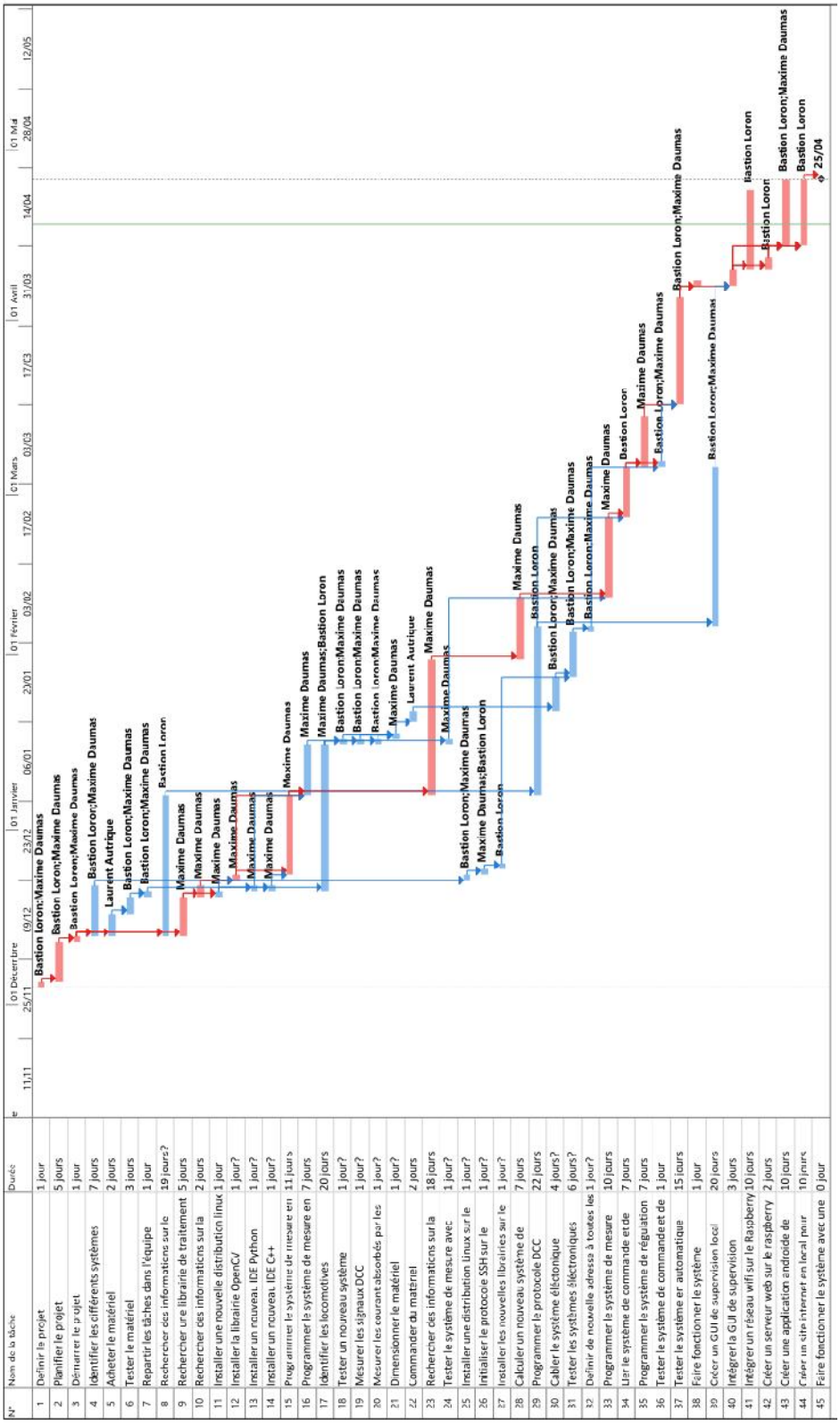
http://benoit.bouchez.free.fr/digitrain4.htm

E**space Train Miniature**[En ligne] Espace Train Miniature, 2013 [consulté le 1 janvier 2014] . Disponible sur :

http://www.espacetrainminiature.fr/

Annexes

1. Annexe 1: Diagramme de Gantt



Projet réalisé par: Maxime Daumas, Bastien Lauron

Projet encadré par: Laurent Autrique

Résumé

Dans de nombreux pays industrialisés, le nombre de véhicules sur les routes est en constante augmentation.
Les infrastructures routières existantes sont vieillissantes et s'avèrent assez vite saturées, ne pouvant assurer la fluidité du trafic.
Par définition un bouchon est l'accumulation, sur une ou plusieurs fils continues et sur une distance d'au moins 500 mètres, de véhicules progressant à une allure très lente ou par bonds.
Dans ce contexte, notre mission était de décongestionner le trafic sur les routes. Nous utiliserons des « trains de véhicule » de manière durable. Il s'agit de faire évoluer de manière automatisée un convoi de véhicules.
Pour chacun des véhicules formant un convoi, l'enjeu est de maîtriser la distance qui le sépare du véhicule qui le précède.
Ce procédé de régulation pourrai alors tripler les capacités des autoroutes.

Mots clés: véhicules, embouteillage, automatiser, convoi, régulation

Abstract:

In many industrialised countries, the number of vehicles on the road constantly grows. The road infrastructure is old, its capacity do not correspond to the needs and traffic jams are a real problem.

By definition a traffic jam is the accumulation of one or more vehicles over a distance of at least five hundred meters, vehicles progressing at a very slow speed or jumps.

To solve this problem and make the road safer, the solution is to regulate the speed of each vehicle and make them go forward by convoy, the stake being to control the distance between the vehicle ahead.

The automobile manufacturers have already begun to develop equipments that will allow them to do this in a few years.

At our level, we have studied the proccessus and developped a small-scale model in our research laboratory, using elements from model railroading. The goal of our project was to apply this technique on locomitives which are moving on the same rail lines.

Using such a solution, the capacity of the road will be able to increase by three.

Keywords: vehicles, traffic jam, automate, convoy, control