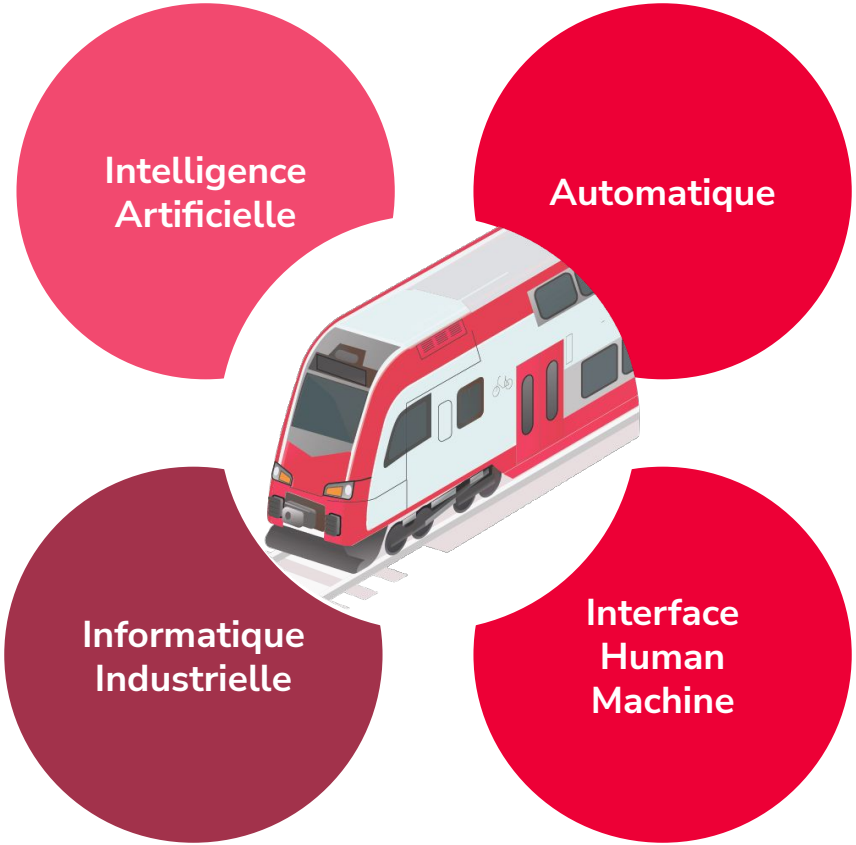
Système de Transport Autonome

Train Autonome

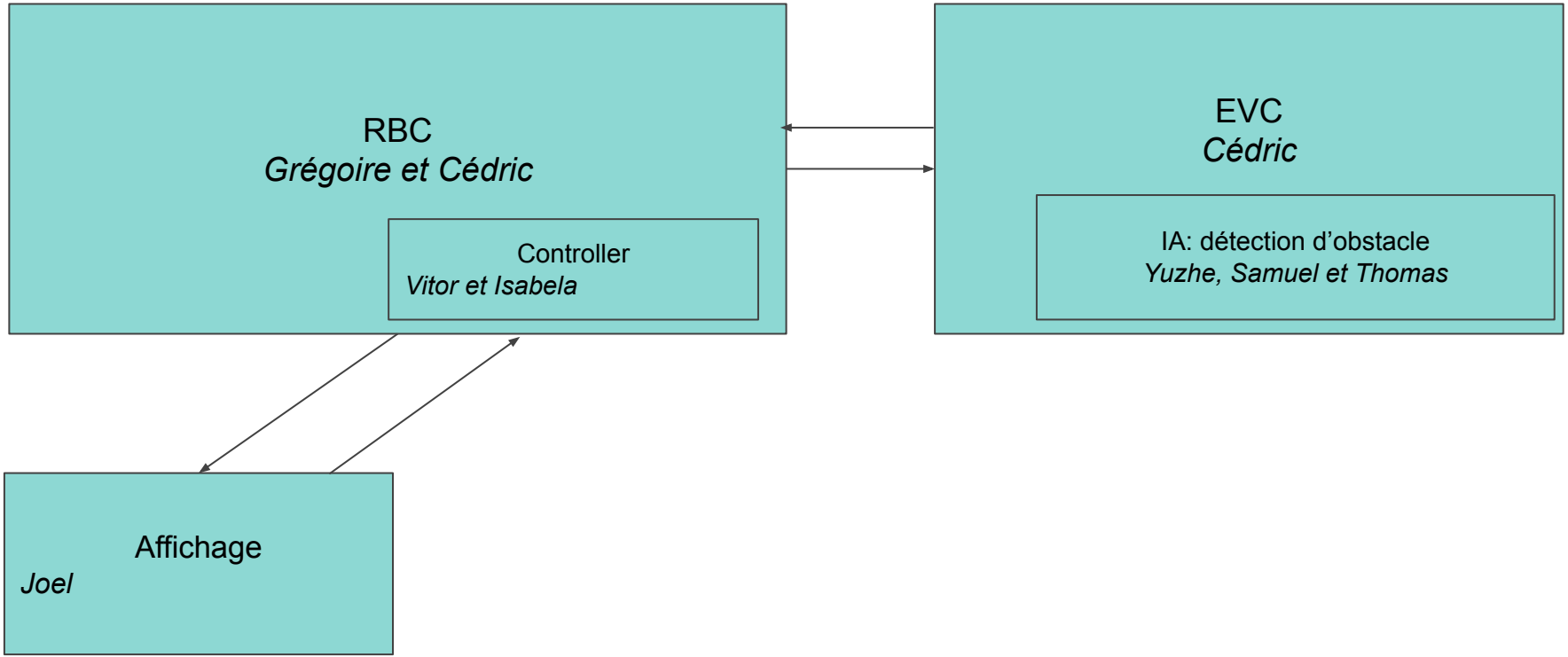
Cédric JUNG
Gregoire GAZEAU
Joel Kalil PONTES
Yuzhe YAO
Isabela FARIA
Samuel MARCIANO
João Vitor PISNO
Thomas DEFFONTAINES


Intelligence Artificielle
Automatique
Informatique Industrielle
Interface Human Machine

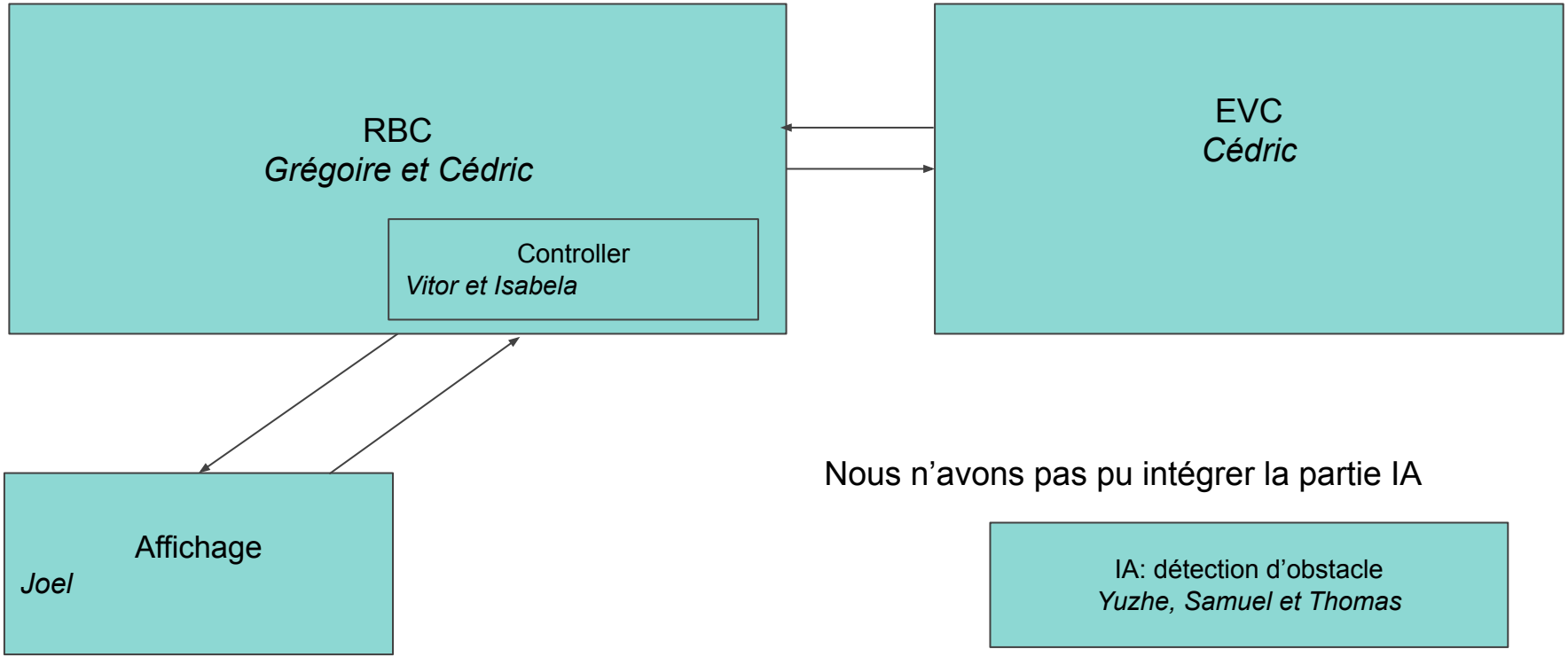
Sommaire

I) Structure, Réseaux et Communication

Structure voulue



Structure réalisée



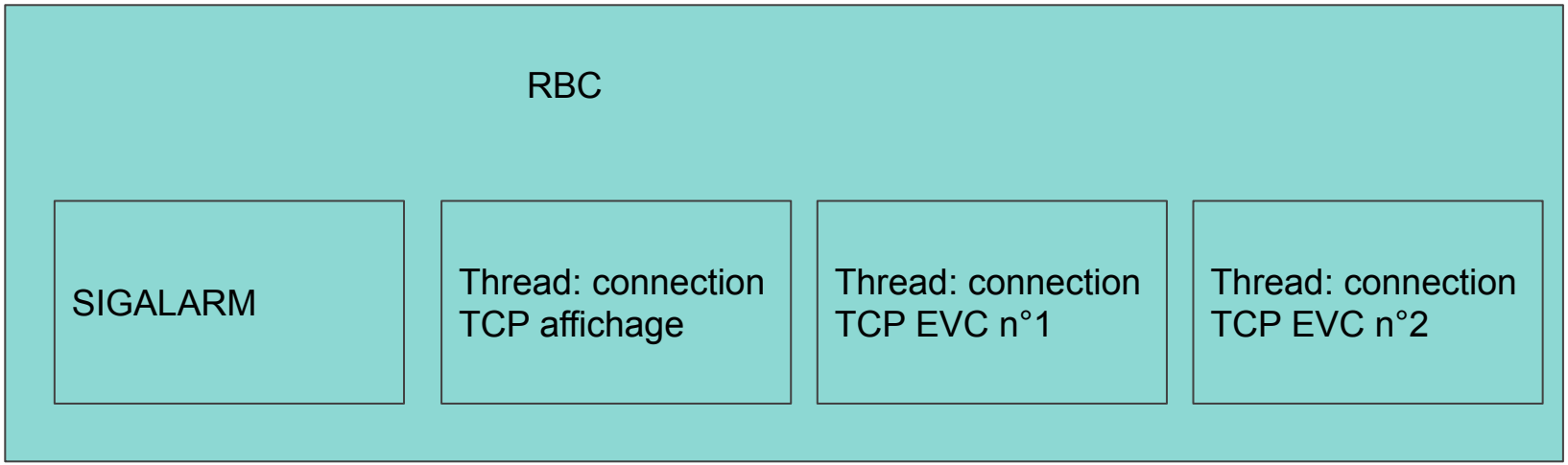
Nous n'avons pas pu intégrer la partie IA

IA: détection d'obstacle
Yuzhe, Samuel et Thomas

Nous l'aurions fait avec des socket AF_UNIX

Structure des processus RBC

- Un thread par connection TCP: 1 pour l'affichage et 1 par EVC connecté
- Une clock fonctionnant par SIGALARM qui fonctionne toutes les 34ms pour appeler le contrôleur

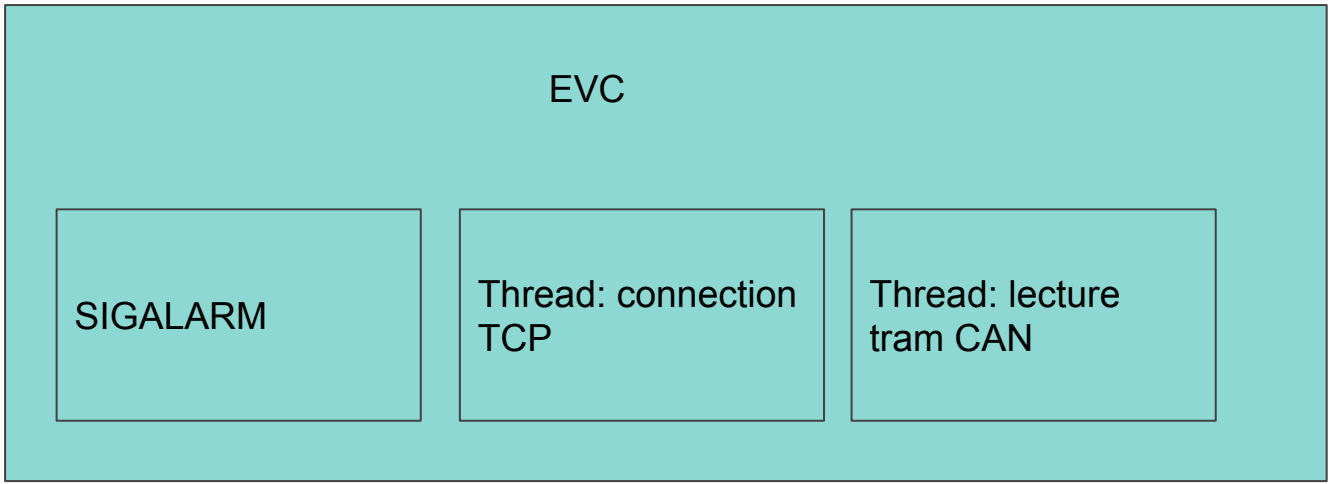


Structure de la liste chaînée de train

```
typedef struct train {
        int id;                     // Id du train
        int pos;                    // Position du train
        int speed;                  // Vitesse du train calculé par le controller
        int sock;                   // Socket pour la communication avec le train
        int speedConsigne;          // Vitesse consigne (entrée pour le controller)
        int speedMeasured;          // Vitesse mesurée (entrée pour le controller)
        int order;                  // Ordre (dans l'objectif de démarrer en déterminant un leader)
        int initialized;            // Variable afin de voir si le controller a été initialisé
        int connected;              // Variable afin de voir si on a une connection avec le train
        int initText;               // Variable afin de voir si on a commencé à écrire le CSV
        struct timeval lastDisplay; // Date du dernier message envoyé à l'affichage
        struct train *nextTrain;    // Train suivant dans la liste chaîné
} Train;
```

Structure des processus EVC

- Un thread pour la lecture de la tram CAN
- Une clock fonctionnant par SIGALARM qui fonctionne toutes les 20ms pour envoyer la position et la vitesse au RBC
- Un thread pour la communication avec le RBC (notamment réception de la consigne)



Protocole de communication

Fonctionnement en TCP/IP

Avantage: mode connecté

Entre le RBC (serveur) et les EVC (clients) :

Un message: **code**:**id**:**position**:**speed** séparateur: :

Plusieurs messages: **code**:**id**:**position**:**speed$$code**:**id**:**position**:**speed** séparateur: $$

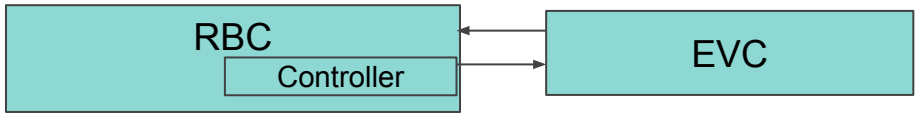
Codes utilisés

1 : Demande de l'ajout d'un train au RBC
2 : Train ajouté à la liste (ack code 1)
3 : Envoie de la position et vitesse vers le RBC
4 : Position et vitesse reçue par le RBC (ack code 3)
5 : Consigne reçue par l'EVC (ack code 6)
6 : Envoie de la consigne vers l'EVC

Code impaire: EVC -> RBC
Code paire: RBC -> EVC

Protocole de communication

Entre le RBC (serveur) et les EVC (clients) :



Un message: **code:id:position:speed** séparateur: :

Plusieurs messages: **code:id:position:speed$$code:id:position:speed** séparateur: $$

Gestion des messages Définition de fonctions communes au RBC et à l'EVC

Pour chaque EVC on utilise un thread pour communiquer avec lui:

```
int sendData(int socket, int code, int id, int position, int speed)
```

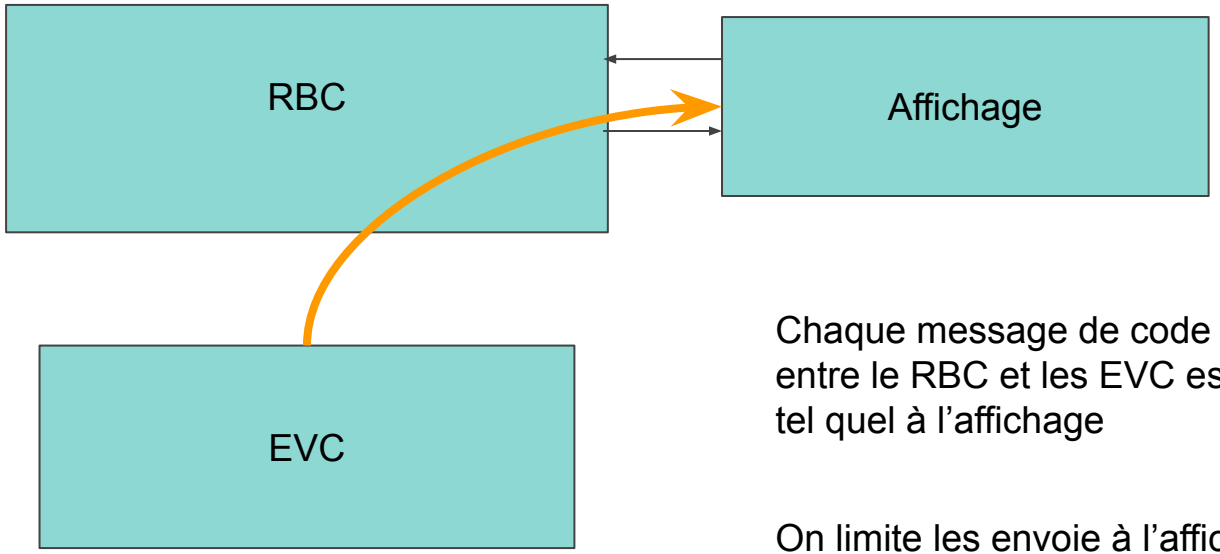
```
T_list splitMessages(T_list list, char data[])
```

On reçoit les messages dans une chaine de caractère et on les mets dans une liste FIFO.

```
T_list parseMessage(T_list list, int* code, int* id, int* position, int* speed)
```

On parse une chaîne de caractère on extrait les différentes informations.

Protocole de communication entre l'affichage et le RBC

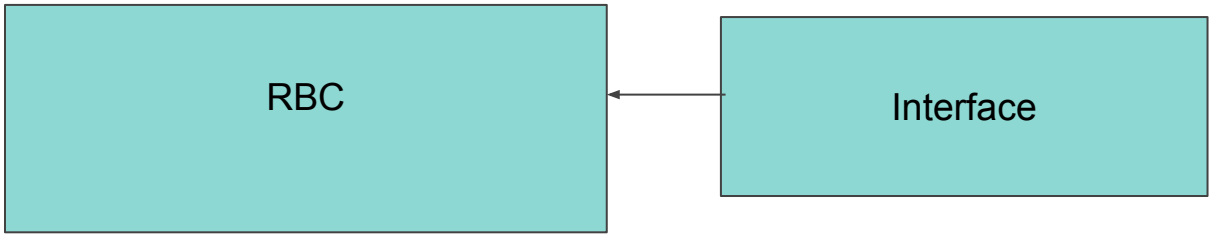


Chaque message de code 4 et 6 entre le RBC et les EVC est transmis tel quel à l'affichage

On limite les envoie à l'affichage à un message toute les 0.02s

Protocole de communication entre l'affichage et le RBC

Transmission de la vitesse consigne à respecter depuis l'interface



Un message: **10**:**-1**:**id_train**:**speed** :

où **id_train** est l'id du train à contrôler
et **speed** est la vitesse

-1 correspond à l'id assigné par défaut à l'interface
10 correspond au code pour le RBC

Période d'appel du controller

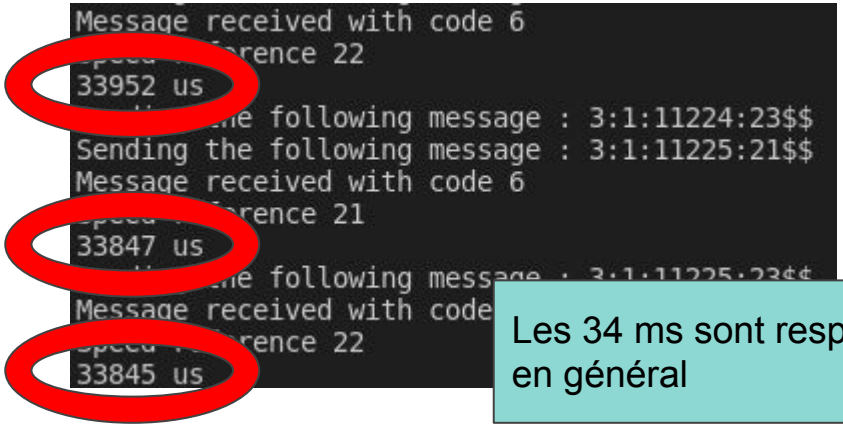
2 méthodes de vérifications

Création côté RBC de CSV

onestep_duration	period	distance	vitesse_consigne	vitesse_reelle	vitesse_envoyee
0	0.034	11610	20	38	34.8
0	0.034	11610	20	38	34.7
0	0.034	11610	20	38	34.7
0	0.034	11610	20	38	34.7
	0.034	11600	20	39	35.6

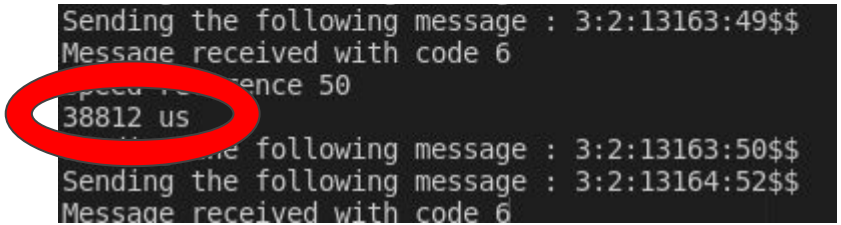
Côté RBC tout est effectué en 34ms

Affichage côté EVC : entre deux réceptions de consignes



Les 34 ms sont respectés en général

En fonction du réseau cela peut être plus long



situation observé lorsque 7 machines y étaient connectés

Détection de balise

Diagramme de séquence: démarrage

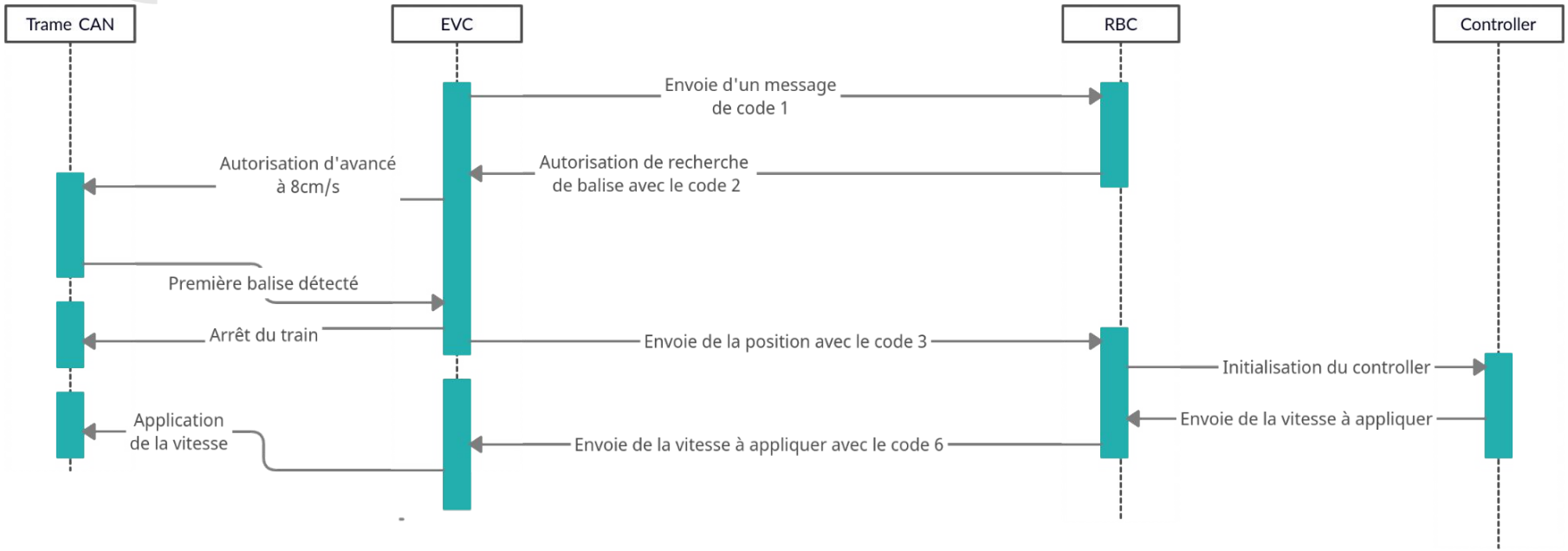


Diagramme de séquence: en fonctionnement nominal

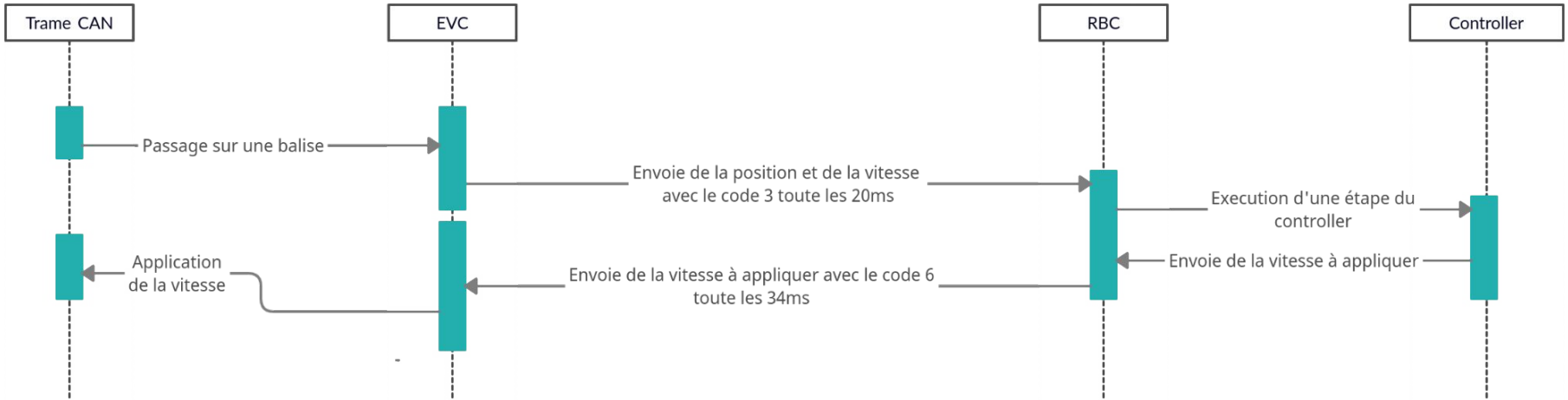
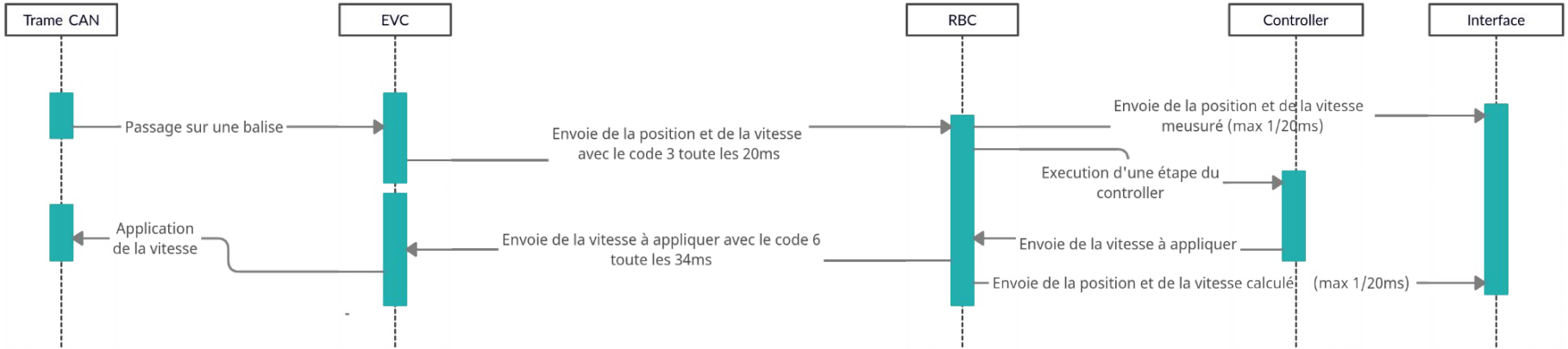
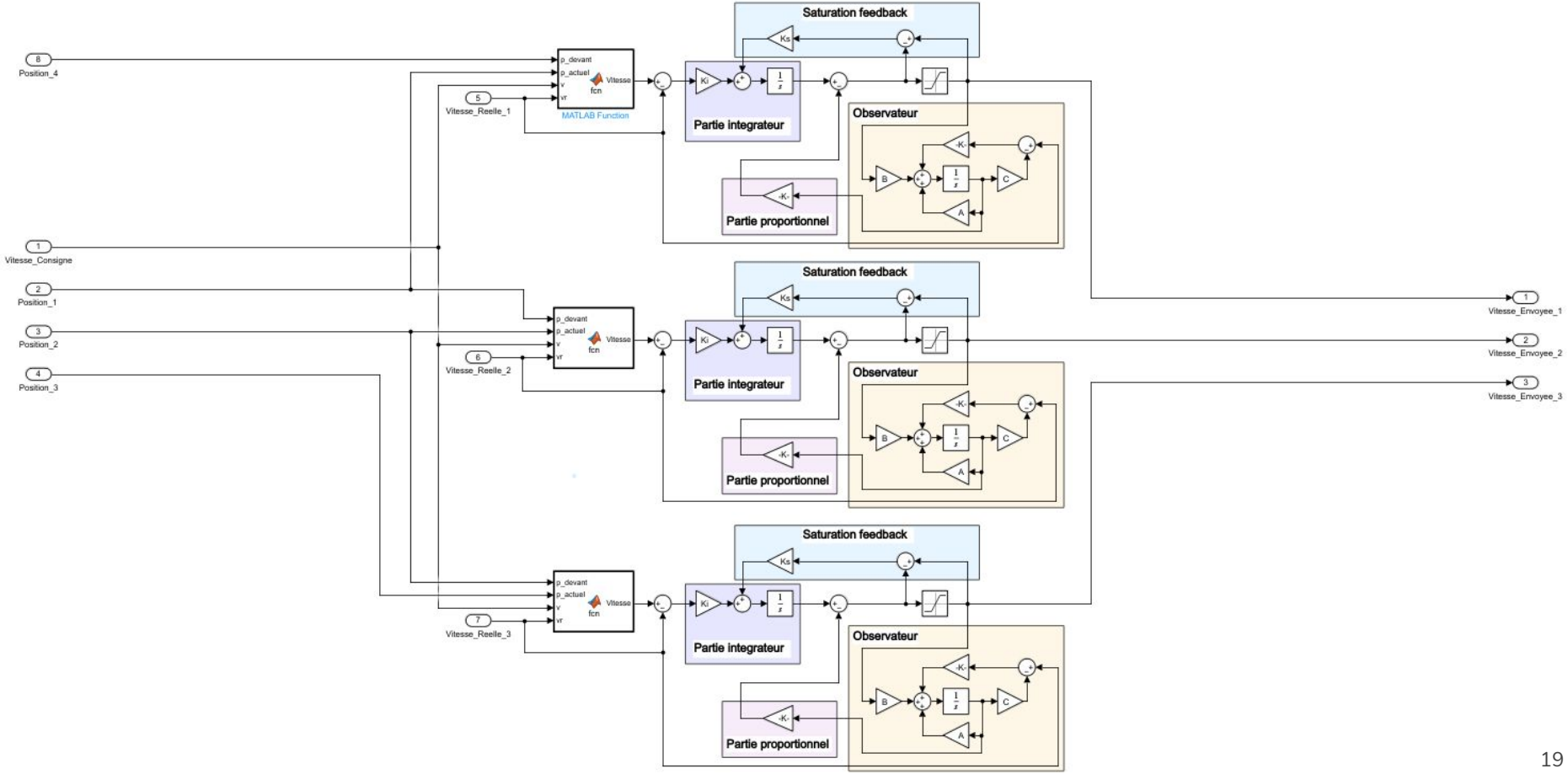


Diagramme de séquence: en fonctionnement nominal avec interface



II) Commande et Automatique



Identification du modèle

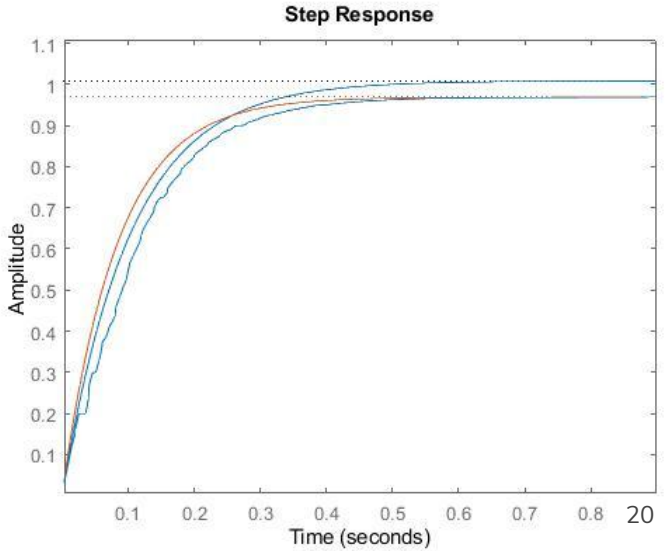
Objectif:

1. Obtenir une modèle de premier ordre pour chaque train avec des simulations

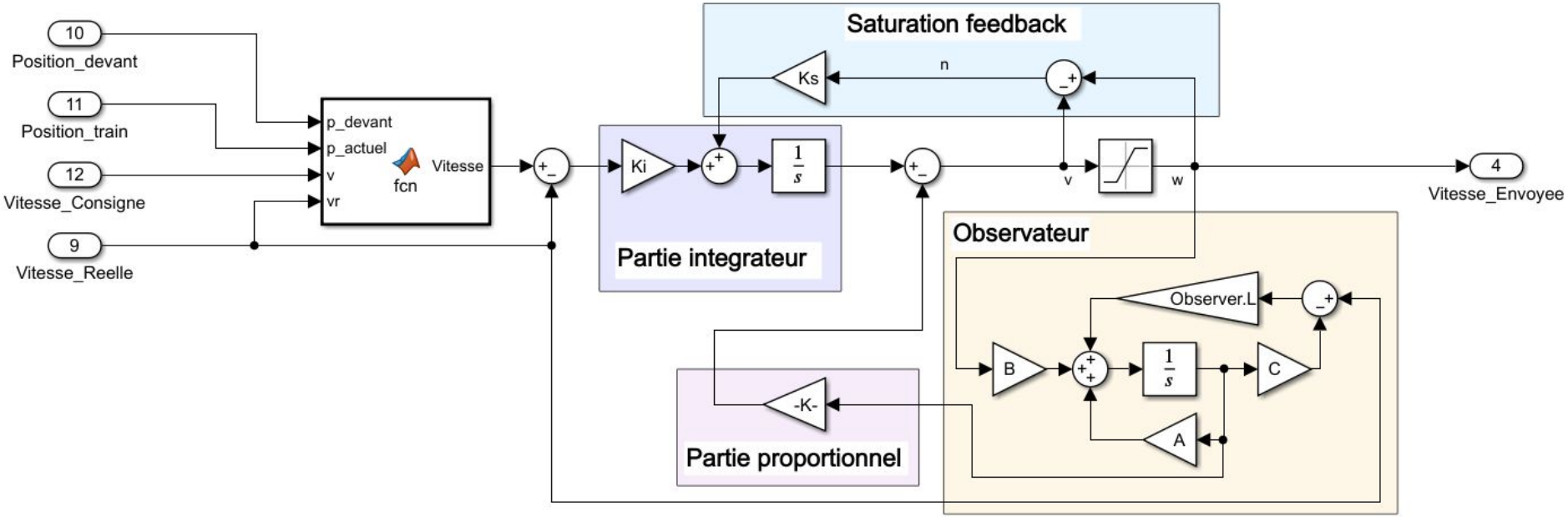
$$G(s) = \frac{Y(s)}{U(s)} = \frac{K}{\tau s + 1},$$

$$y(t) = KA(1 - e^{-t/\tau}),\ t \geq 0$$

$$\frac{11.59}{s + 11.96}$$

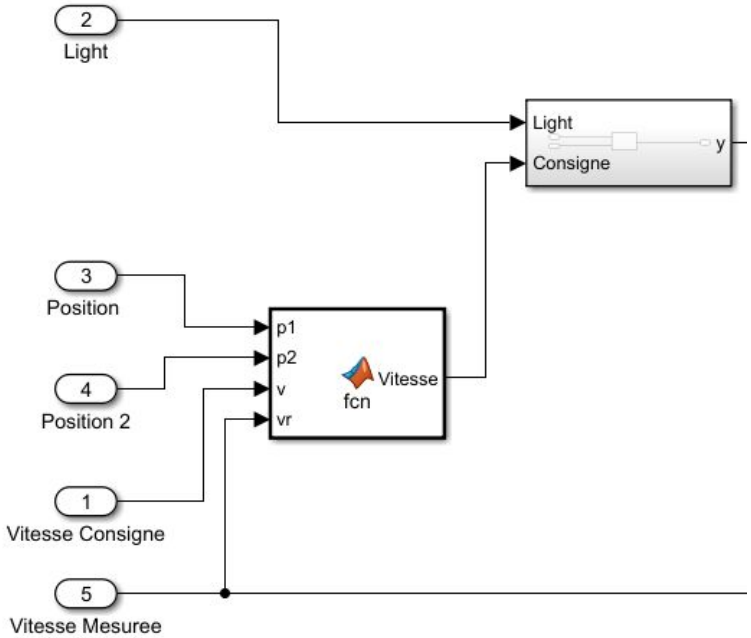


Projet du contrôleur

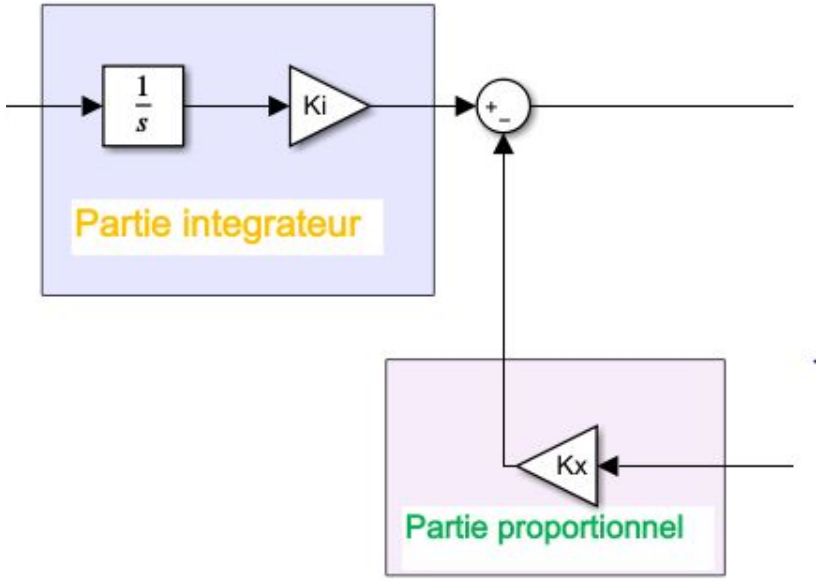


Calcul du gain du contrôleur

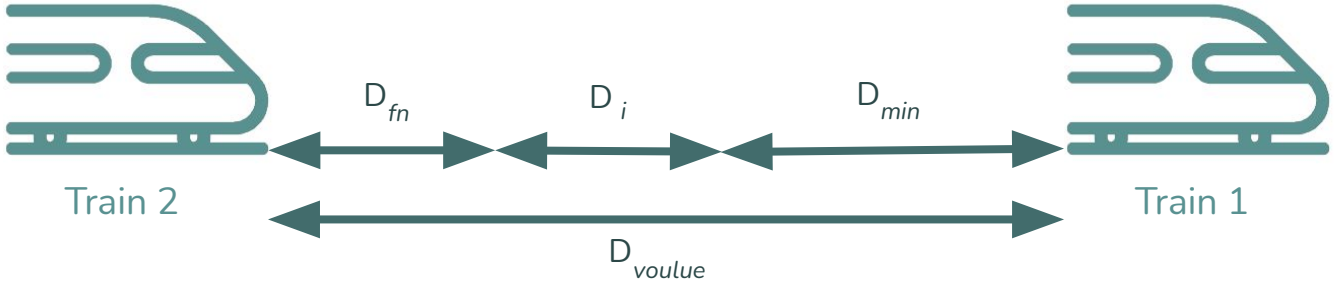
Commande par position:



Commande par vitesse:



Commande par position:



$D_{fn} = t_b * v + \frac{v^2}{2 * \gamma}$: distance due au temps de réaction

$D_i = 0{,}03 * v^2 + 0{,}28 * v$: distance de freinage due à l'inertie

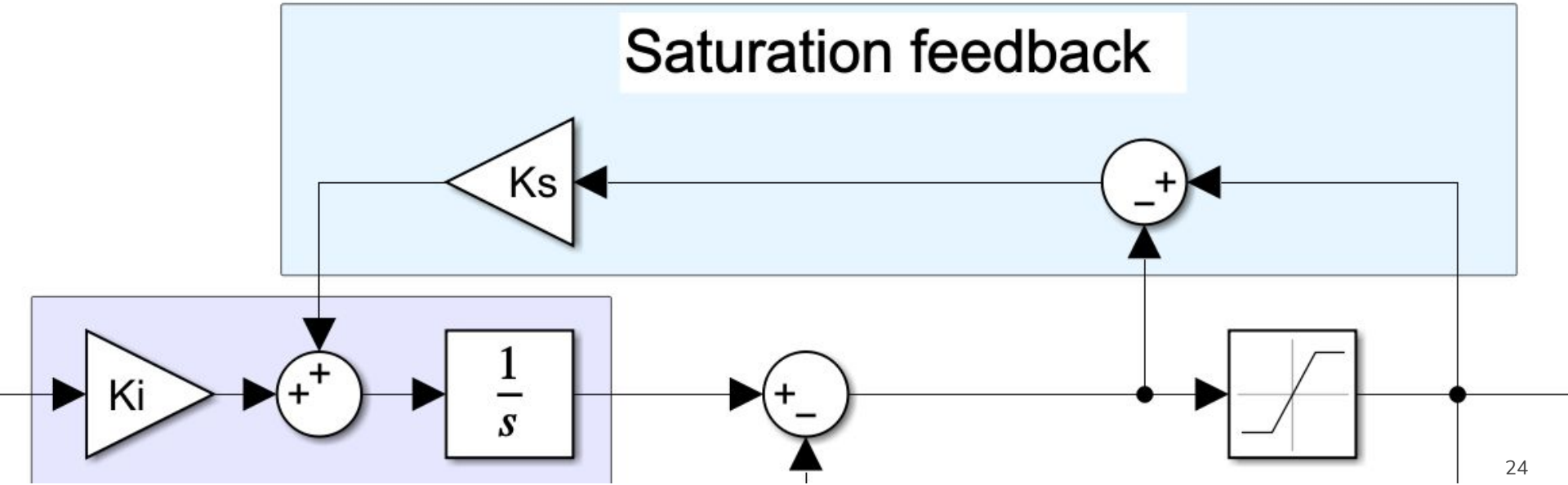
D_{min} écart entre deux trains souhaité

$$V = V_{ideal} \frac{\left(x^2 - D_{min}^2\right)}{D_{frenaje}^2 + 2D_{min} * D_{frenaje}}$$

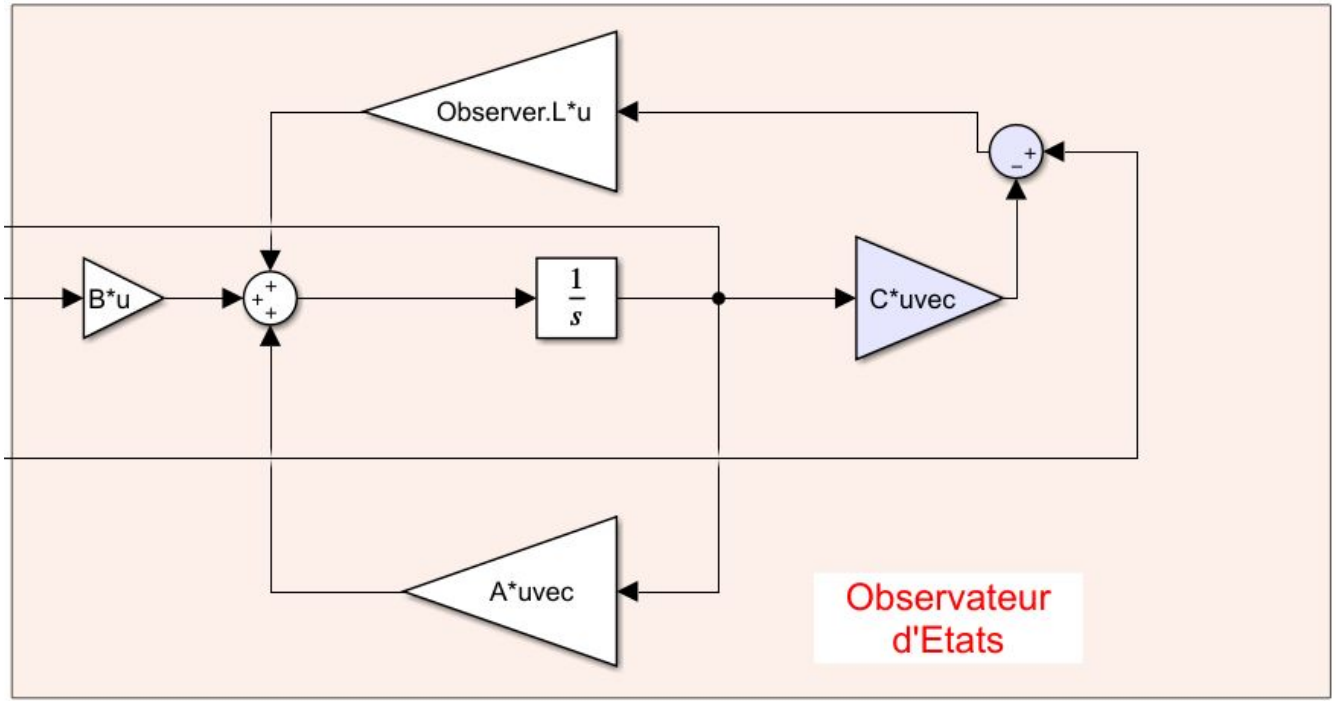
23

Compensation de saturation

Ks=10*Ki;



Projet de l'Observateur

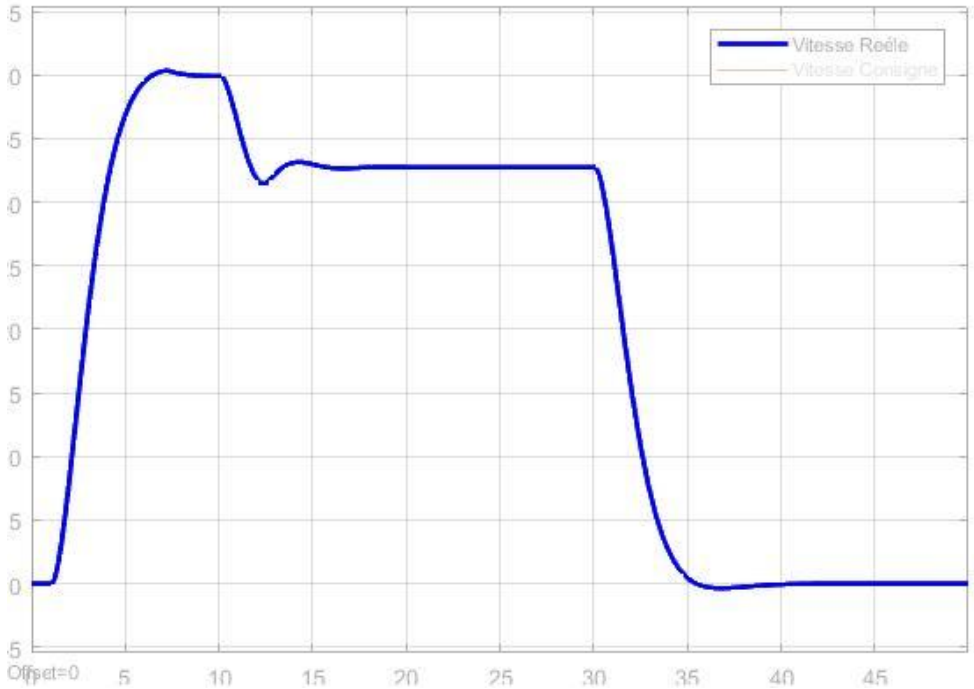


Test du Commande

1. Premièrement, il y a une distance nécessaire;

2. Deuxièmement, il y a la distance minimale autorisée;

3. Dans la distance minimale.



Exportation du code

Inports

[-]

Block Name	Code Identifier	Data Type
<Root>/Vitesse_Consigne	controle_v7_1_U.Vitesse_Consigne	real_T
<Root>/Light	controle_v7_1_U.Light	real_T
<Root>/Distance	controle_v7_1_U.Distance	real_T
<Root>/Vitesse_Reelle	controle_v7_1_U.Vitesse_Reelle	real_T

Outports

Block Name	Code Identifier	Data Type
<Root>/Vitesse_Envoyee	controle_v7_1_Y.Vitesse_Envoyee	real_T

Entry-Point Functions

Function: controle_v7_1_initialize

Prototype	**void controle_v7_1_initialize(void)**
Description	Initialization entry point of generated code
Timing	Must be called exactly once
Arguments	None
Return value	None
Header file	controle_v7_1.h

Function: controle_v7_1_step

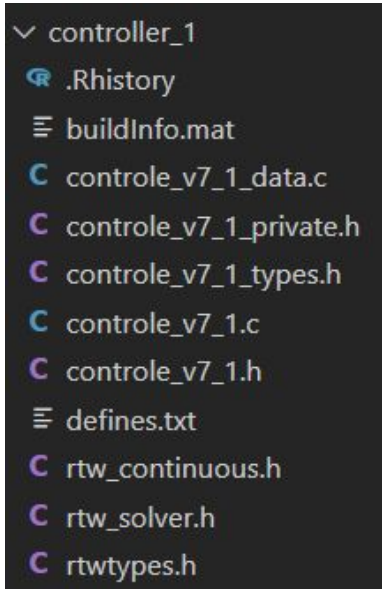
Prototype	**void controle_v7_1_step(void)**
Description	Output entry point of generated code
Timing	Must be called periodically, every 0.034 seconds
Arguments	None
Return value	None
Header file	controle_v7_1.h

Function: controle_v7_1_terminate

Prototype	**void controle_v7_1_terminate(void)**
Description	Termination entry point of generated code
Timing	Must be called exactly once
Arguments	None
Return value	None
Header file	controle_v7_1.h

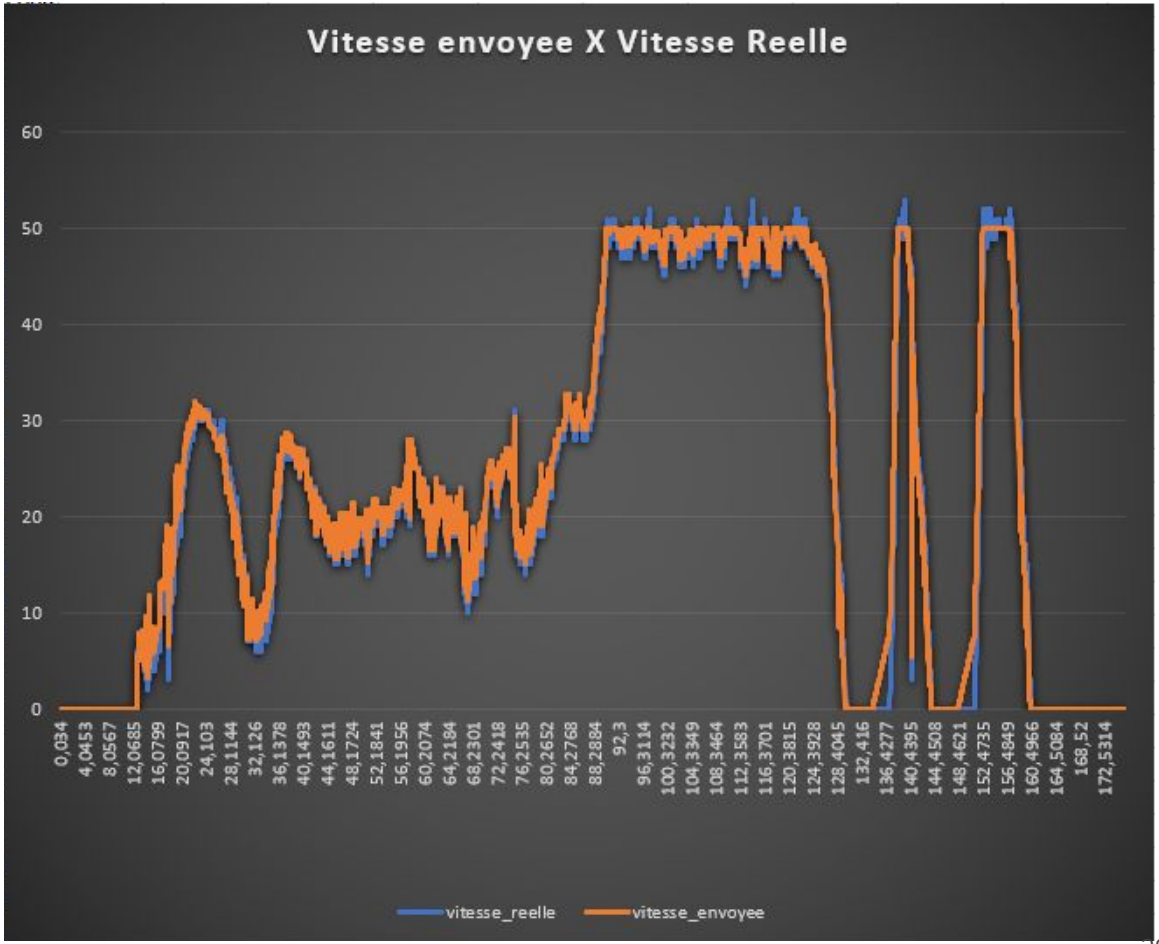
Exportation du code

```
#include "rtw/rtw_continuous.h"
#include "rtw/rtw_solver.h"
#include "rtwtypes.h"


#include "controller_1/controle_v7_1.h"
#include "controller_2/controle_v7_2.h"
```

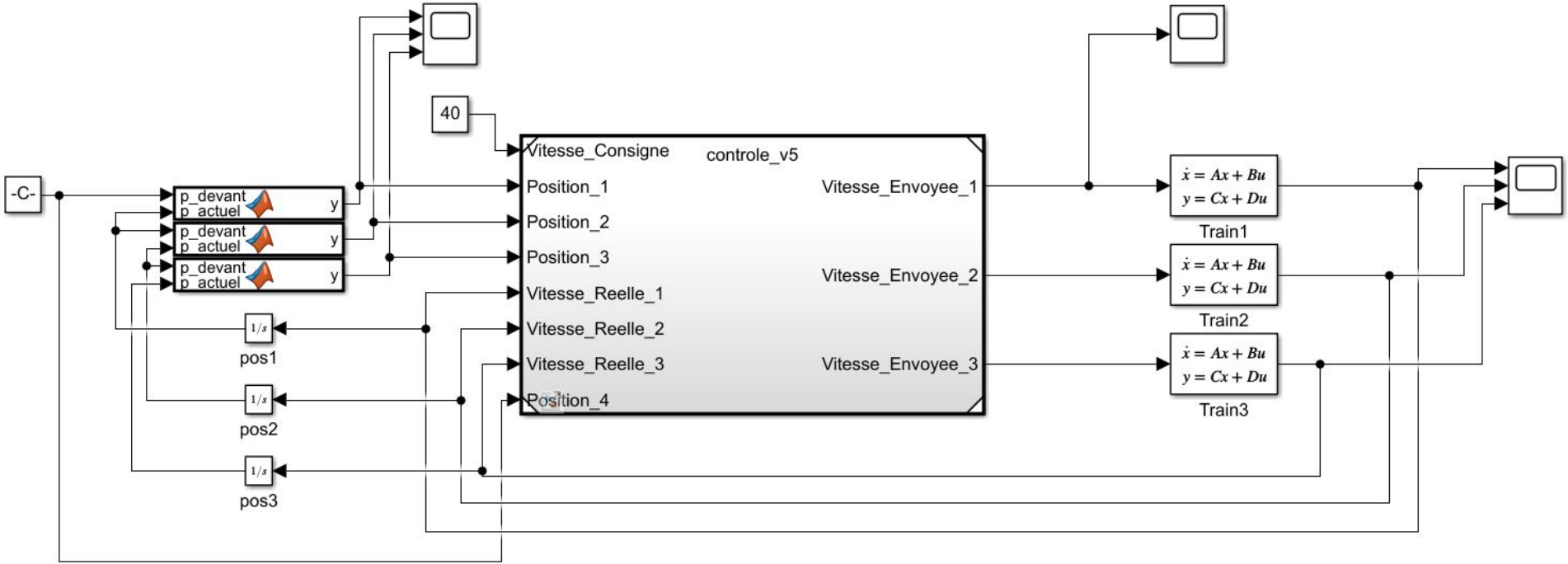



```
controle_v7_1_U.Distance = defaultDistance;
controle_v7_1_U.Vitesse_Consigne = defaultConsigne;
controle_v7_1_U.Vitesse_Reelle = train->speedMeasured;
controle_v7_1_U.Light = 0;

printf("Distance 1->2 %f\n", controle_v7_1_U.Distance);

/* Step the model for base rate */
controle_v7_1_step();

/* Get model outputs here */
// Log all outputs here
//printf("Vitesse consigne (Output) %d\n", (int) controle_v7_1_Y.Vitesse_Envoyee);

train->speed = (int)  controle_v7_1_Y.Vitesse_Envoyee;
```

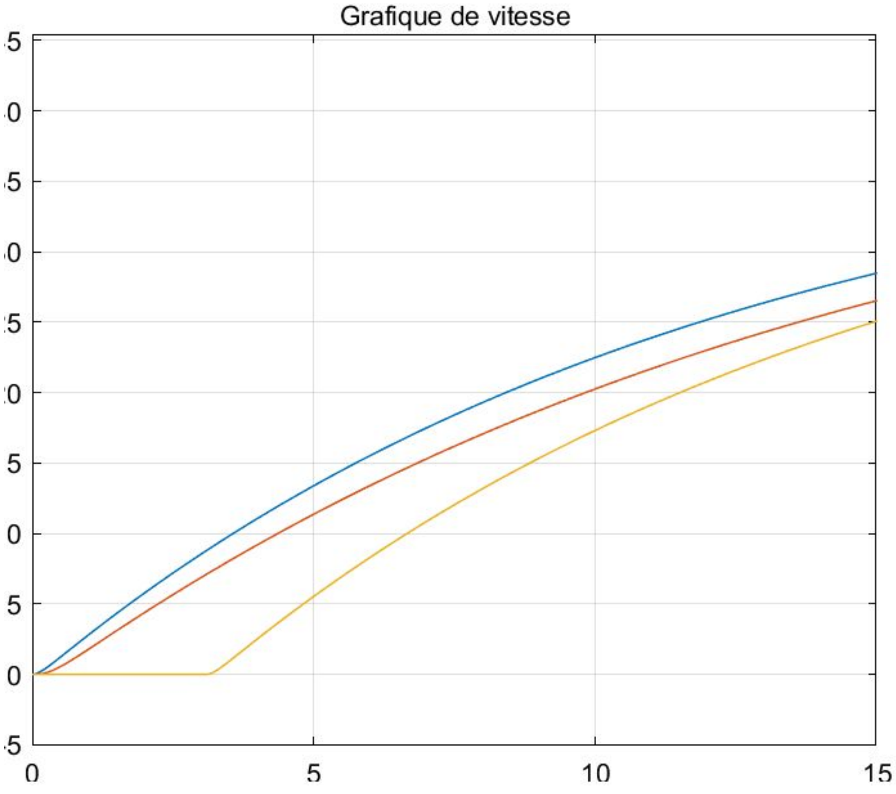
Résultats

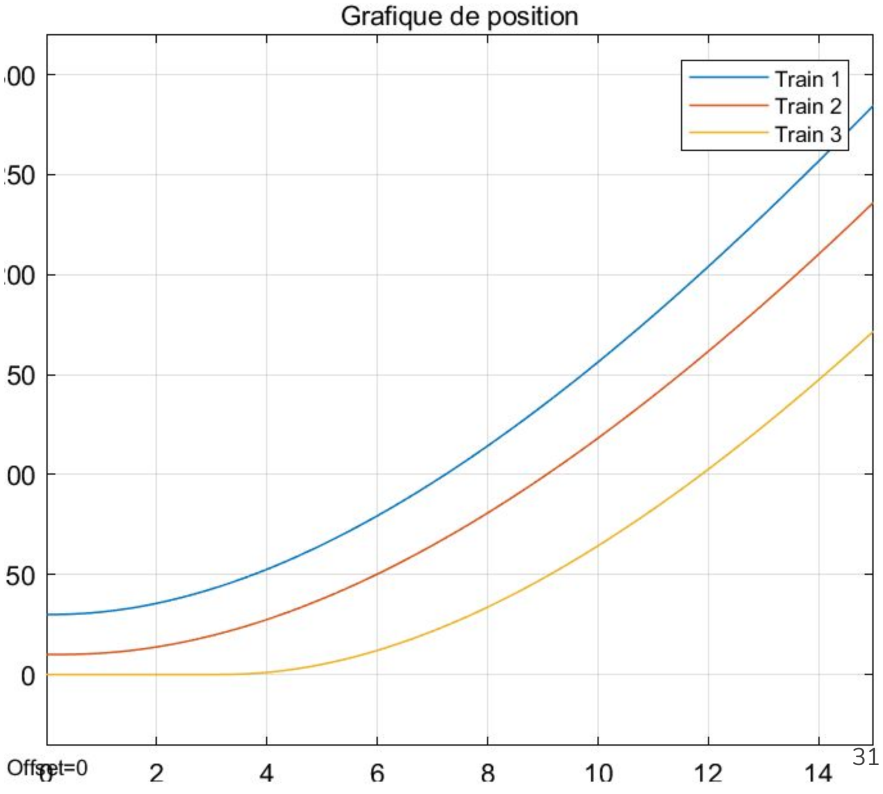

Vitesse envoyee X Vitesse Reelle
60
50
40
30
20
10
0
0,034
4,0453
8,0567
12,0685
16,0799
20,0917
24,103
28,1144
32,126
36,1378
40,1493
44,1611
48,1724
52,1841
56,1956
60,2074
64,2184
68,2301
72,2418
76,2535
80,2652
84,2768
88,2884
92,3
96,3114
100,3232
104,3349
108,3464
112,3583
116,3701
120,3815
124,3928
128,4045
132,416
136,4277
140,4395
144,4508
148,4621
152,4735
156,4849
160,4966
164,5084
168,52
172,5314
vitesse_reelle
vitesse_envoyee

Simulation 3 trains



Simulation 3 trains





III) Détection d’obstacles

Cahier des charges :

➜ Constituer un dataset

➜ Implémenter un algorithme permettant de détecter des objets sur la voie ferrée

➜ Arrêter le train en fonction du risque

Pistes de solutions d'implémentation :

➜ Soustraction d’images (problème de précision)

➜ YOLO (obstacles non prévisibles)

➜ ***Réseau de neurones*** 

Constitution de la dataset

DATASET COHÉRENTE

➜ Images capturées par la PiCaméra placée à l'avant du train

➜ Images capturées à différentes vitesses du train

⇒ *Conditions exactement identiques - contexte optimal*

↳ Plus de 1000 images

↳ Environ 50% avec obstacle et -50% sans obstacle

AVEC OBSTACLES

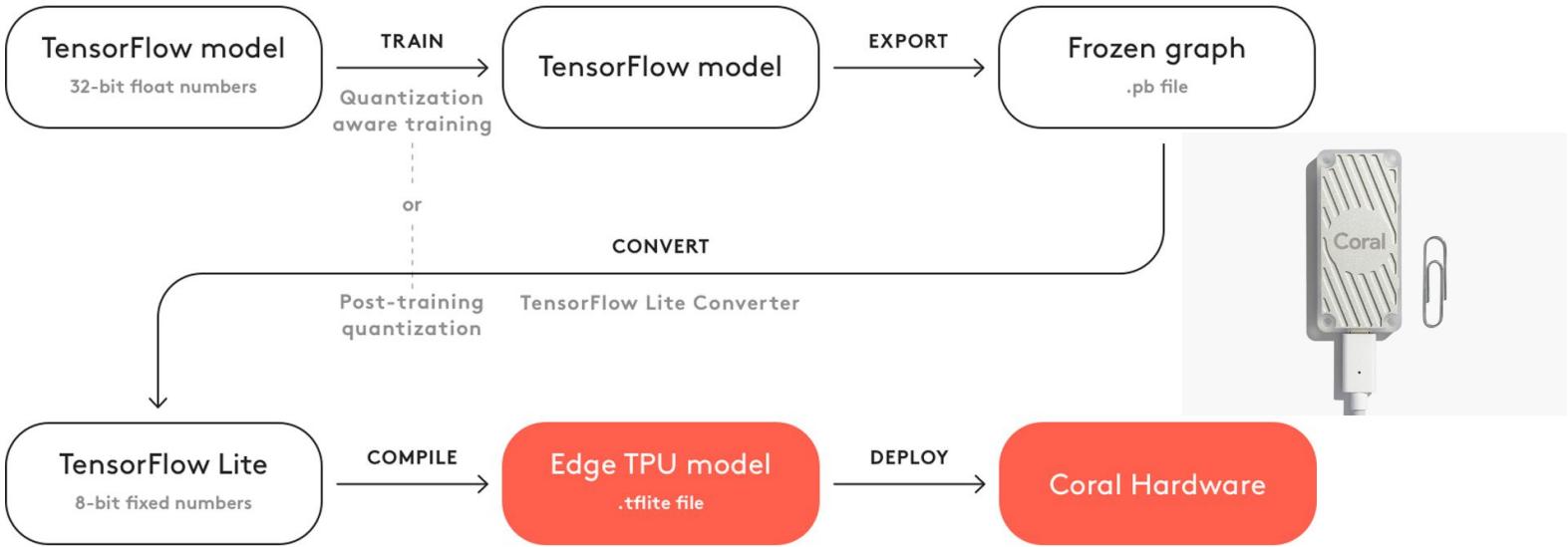
SANS OBSTACLE

Exemples : Cas complexes

➛ *Avec obstacle lointain:*

➛ *Sans obstacle dans un virage:*

Modèle


TensorFlow model
32-bit float numbers
TRAIN
Quantization aware training
TensorFlow model
EXPORT
Frozen graph
.pb file
or
CONVERT
Post-training quantization
TensorFlow Lite Converter
Coral
TensorFlow Lite
8-bit fixed numbers
COMPILE
Edge TPU model
.tflite file
DEPLOY
Coral Hardware

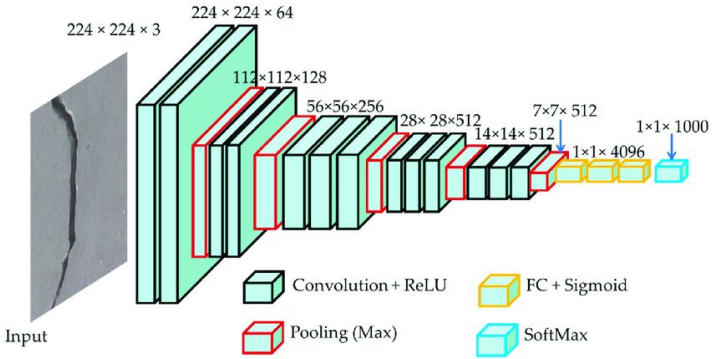
Modèle

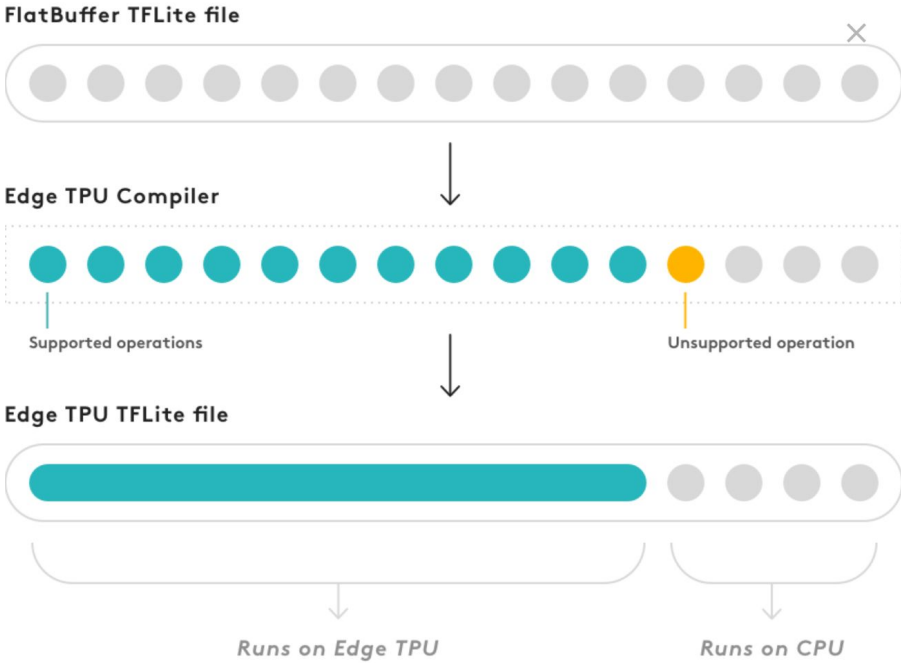
- Quantification
  - post-training quantization
    - Full integer post-training quantization
      - representative dataset
  - quantization sensitive training

Technique	Benefits	Hardware
Dynamic range quantization	4x smaller, 2x-3x speedup	CPU
Full integer quantization	4x smaller, 3x+ speedup	CPU, Edge TPU, Microcontrollers
Float16 quantization	2x smaller, GPU acceleration	CPU, GPU

Modèle

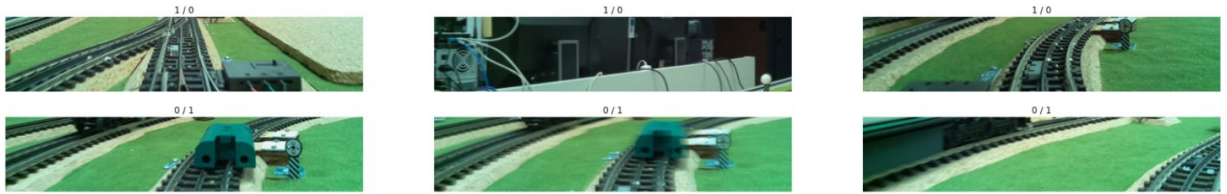
- Adaptabilité des opérations
  - VGG16





Modèle

```
plt.figure(figsize=(30,25))
n_all = X.shape[0]
i=1
for j in range(len(X)) :
    if (y_cnn_all[j].argmax(axis=-1) != y[j].argmax(axis=-1)):
        plt.subplot(10,3,i)
        plt.axis('off')
        plt.imshow(X[j])
        plt.title('%s / %s' % (Classes[y_cnn_all[j].argmax(axis=-1)], Classes[y[j].argmax(axis=-1)]))
        i+=1
```



[28]:
```
confusion_matrix(y.argmax(axis=-1),y_cnn_all.argmax(axis=-1),labels=[1,0])
```

[28]:
```
array([[404,   3],
       [  3, 565]])
```

Algorithme

Importation des librairies et du modèle

```python
import time
import numpy as np
import tensorflow as tf # ou MobileNet
from PIL import Image
from tflite_runtime.interpreter import load_delegate
from pycoral.utils import edgetpu
from pycoral.utils import dataset
from pycoral.adapters import common
from pycoral.adapters import classify
import os
import picamera
import picamera.array
from numpy.lib.function_base import average

#model_file=os.path.join('/home/pi/','colab_1_edgetpu.tflite')
model_file = os.path.join('/home/pi/', 'mobilenet_v2_1.0_224_quant_edgetpu.tflite')
interpreter = edgetpu.make_interpreter(model_file)

#interpreter = tflite.Interpreter(model_path='mobilenet_v2_1.0_224_quant_edgetpu.tflite',experimental_delegates=[load_delegate('libedgetpu.so.1.0')])
interpreter.allocate_tensors()
# Get input and output tensors.
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()
```

Algorithme

Démarrage de la Pi caméra et prise de photo

```
25 with picamera.PiCamera() as camera:
26   camera.resolution = (480, 300)
27   camera.framerate=50
28   camera.start_preview()
29   camera.rotation = 180
30   j=0
31
32   while (True) :
33     time_start=time.time()
34     print("photo capturee")
35     camera.capture_sequence(['./images_cam/image_cam%d.jpg'%j],use_video_port=True)
36     t = str(j)
37     while not(os.path.exists('./images_cam/image_cam%d.jpg'%j)):
38       p=0
39     img=Image.open('./images_cam/image_cam%s.jpg'%j)
```

Algorithme

Traitement de l’image et prédiction par le modèle

```
40    X1,X2,X3, = np.split(np.array(img), 3, axis=0)
41    input_array=(X3).astype("uint8").reshape([1,100,480,3])#uint8
42    interpreter.set_tensor(input_details[0]['index'], input_array)
43    interpreter.invoke()
44
45    output_data = interpreter.get_tensor(output_details[0]['index'])
```

Algorithme

Affichage des résultats

```
46      print(output_data)
47
48      print(output_data[0][1]>3)
49      print(time.time()-time_start)
```

```
0.44749975204467773
photo capturee
[[255   0]]
False
0.43076276779174805
photo capturee
^[[2~[[255   0]]
False
0.43857908248901367
photo capturee
[[254   2]]
False
0.43706822395324707
photo capturee
[[125 131]]
True
```

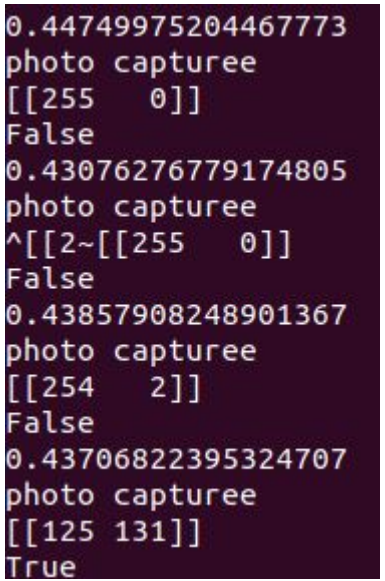
Résultats

➜ Temps d’exécution total : 0.45 secondes environ

➜ Distance voie observée : 10cm - 50cm

Probabilité de présence d’obstacle :

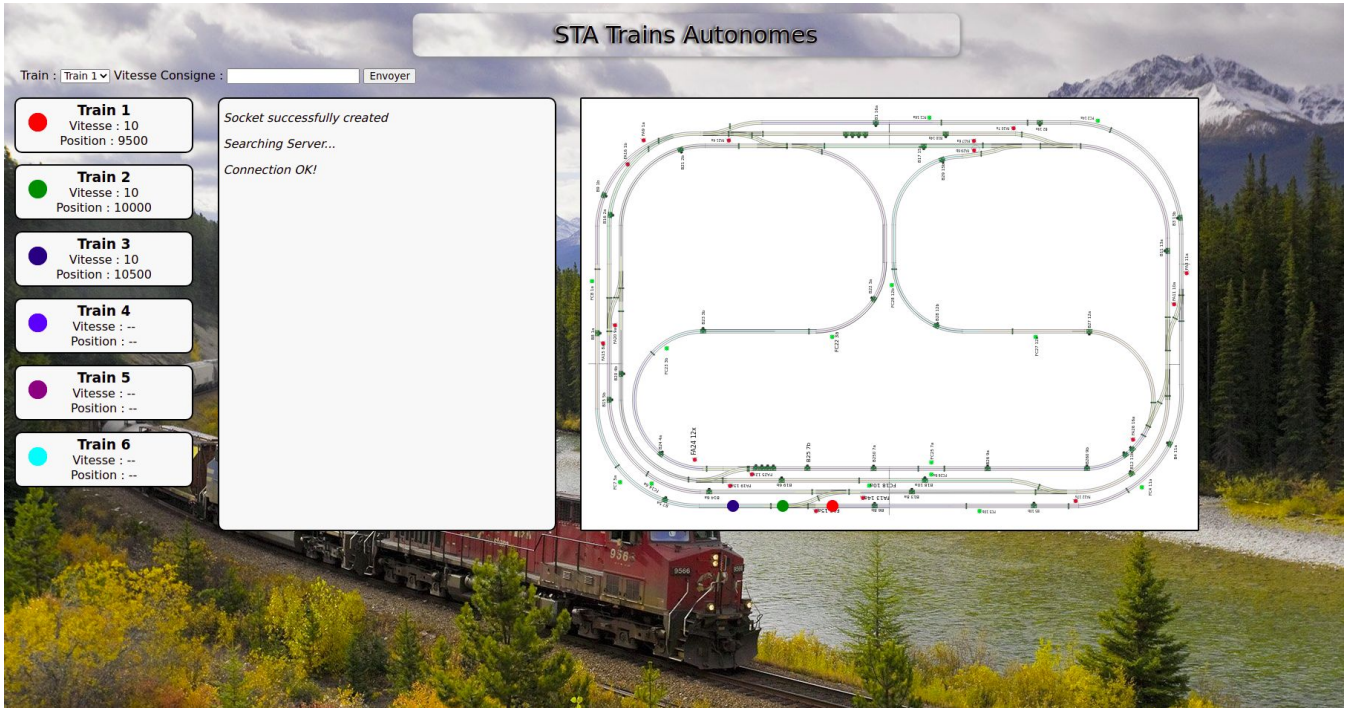
- Si absence d’obstacle : p < 10%
- Si obstacle éloigné (35-50cm) : 10% < p < 50%
- Si obstacle proche : 50% < p



Améliorations possibles

1. Intégration au processus EVC - multi-threading
2. Adaptation la résolution de la capture en fonction du placement du train (ligne droite, virage, ...) ⇨ réduire la taille de l'image ⇒ l'optimisation
3. Passer de 40 ms de temps de redimensionnement à < 10ms (ne plus générer X1,X2,X3 avec la fonction split, mais utiliser X = img [200:,:,])
4. Augmenter la taille de la dataset

IV) Interface Homme Machine



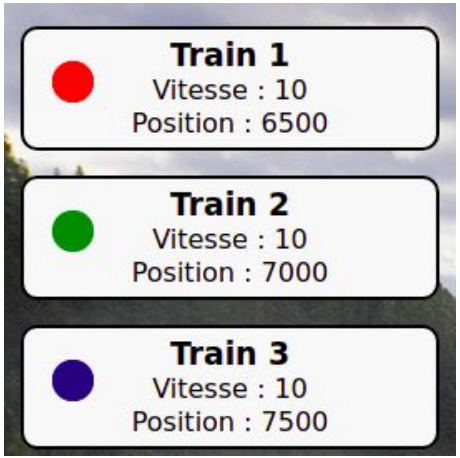
IHM : Objectifs

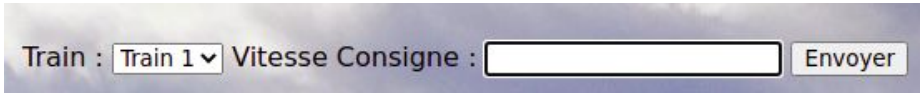
1. Permettre à l'opérateur de visualiser la position de chaque train sur la carte
2. Permettre à l'opérateur d'influencer le système
3. Permettre à l'opérateur de voir si le système de contrôle est efficace
4. Faciliter le traitement des informations par l'opérateur

IHM : Fonctionnalités

1. Affiche la vitesse actuelle et la position de chaque train

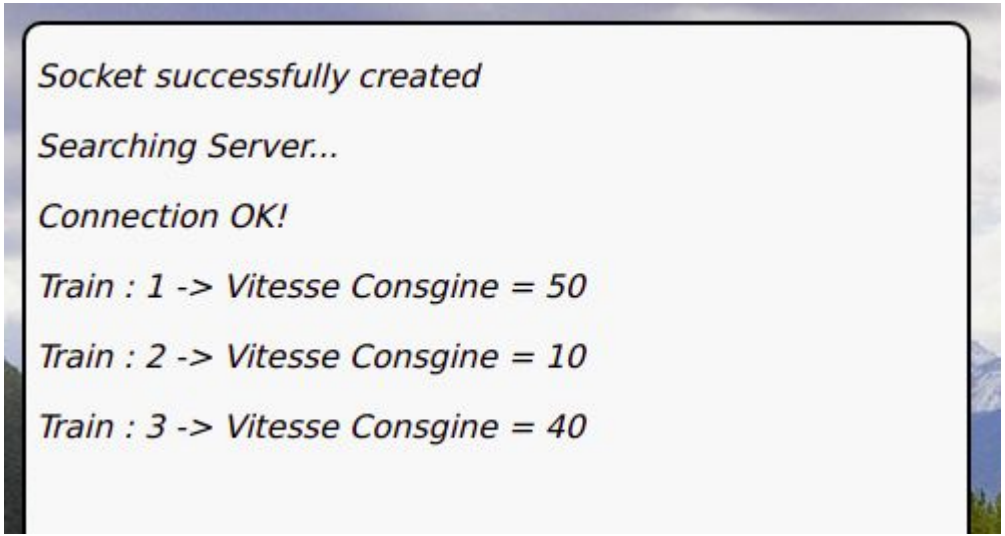
2. Permet à l'opérateur de décider de la vitesse consigné de chaque train





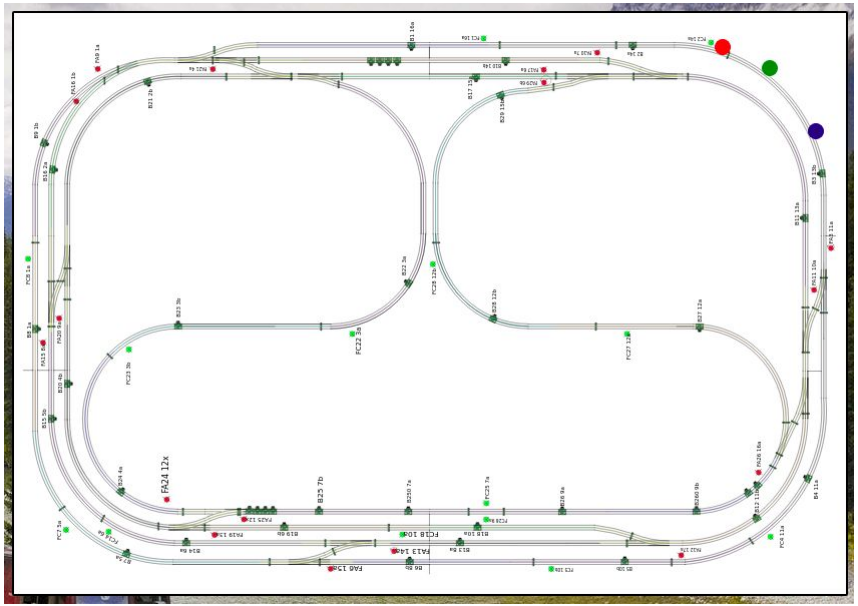
IHM : Fonctionnalités

3. Afficher des messages à l'opérateur sur la connexion du système, les vitesses, etc.



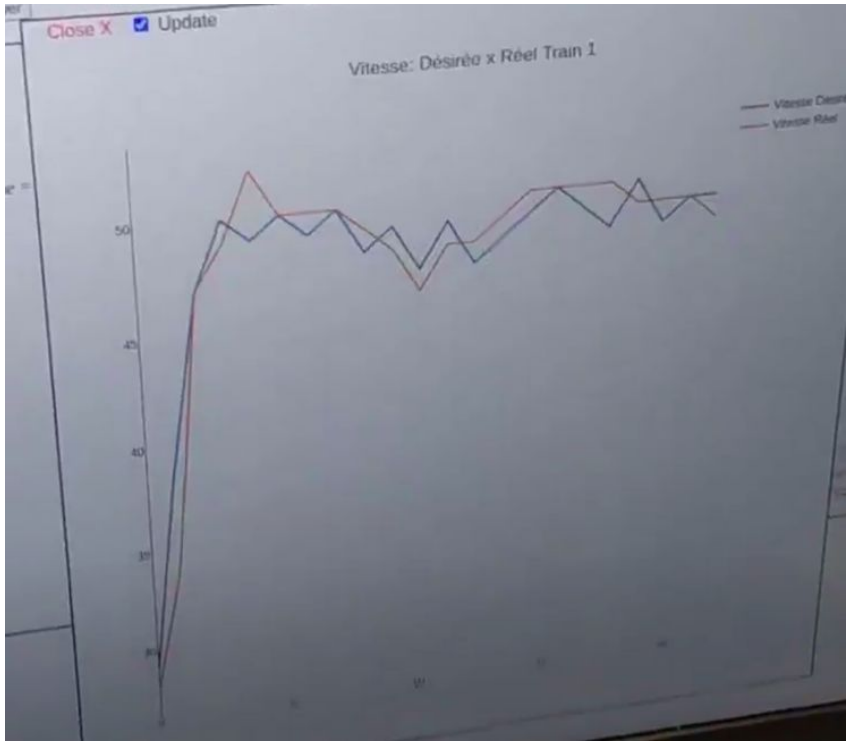
IHM : Fonctionnalités

4. Il montre la position de chaque train par rapport à la carte, ce qui permet à l'opérateur de voir plus facilement sur quel tronçon se trouve chaque train.



IHM : Fonctionnalités

5. Tracez un graphique avec la vitesse réelle par rapport aux valeurs de vitesse consigne (désirée). Vous permettant de visualiser l'historique des valeurs et de faire une comparaison entre les valeurs.



IHM : Développement

- *Parallélisme :*

  -> Thread Flask :

  Générée l'interface Web

  Gère les demandes du Front-End

  -> Thread Socket :

  Responsable pour la communication TCP/IP

  Traitement des données reçues

IHM : Développement

- *“Back - end” :*

IHM : Développement

- *“Front - end” :*

Démonstration