

Synthèse Technique de Réalisation pour la brique Décision

Version : 0.3

Date de version : 20/05/2025

|  |  |
| --- | --- |
| Une image contenant signe, extérieur, ciel, rue  Description générée automatiquement | Opération réalisée avec le concours des Investissements d’avenir de l’Etat confiés à l’ADEME |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Projet | Zone | Lot | Phase | Générations | Type de document | Emetteur | Numéro Chrono | Version | Indice |
| FM-AT-L8-ESE-G0-RAP-SIC-00125-v0.1 | | | | | | | | | |

Informations du document

Périmètre de diffusion : xxx

Type : xxx

Date prévue de livraison : xxx

Statut : xxx

Lieu de stockage du document : xxxx

Auteurs :

|  |  |  |
| --- | --- | --- |
| Pilote(s) | Organisation | Rôle dans le projet |
| B. YAHIAOUI | 12/18/2024 | Création |
| M. BOURKAIB | 01/10/2024 | Création |
|  |  |  |
| Contributeurs | Organisation | Rôle dans le projet |
|  |  |  |
|  |  |  |
|  |  |  |

Table de révision

|  |  |  |
| --- | --- | --- |
| Version | Date | Contenu de la modification |
| 0.1 |  | Sprint 0 |
| 0.2 |  | Sprint 1 |
| 0.3 |  | Sprint 2 |
| 0.4 |  |  |
| 0.5 |  |  |
| 0.6 |  |  |
|  |  |  |
|  |  |  |

Table des matières

[Informations du document 2](#_Toc198617728)

[Table de révision 2](#_Toc198617729)

[Table des matières 3](#_Toc198617730)

[1. Caractéristique logiciel 4](#_Toc198617731)

[2. Description du document 4](#_Toc198617732)

[3. Environnement Software et Hardware 4](#_Toc198617733)

[4. Dispositions particulières 4](#_Toc198617734)

[4.1. Liste des paramètres nécessaires au lancement 4](#_Toc198617735)

[4.2. Descriptif des entêtes de messages 6](#_Toc198617736)

[5. Schéma Global du Module AD 7](#_Toc198617737)

[6. Contenu du module AD\_Core 8](#_Toc198617738)

[7. Brique de TopologyReader 9](#_Toc198617739)

[7.1. Entrées de la sous-brique 9](#_Toc198617740)

[7.2. Sortie de la sous-brique 9](#_Toc198617741)

[7.3. Description des principaux algorithmes 10](#_Toc198617742)

[7.3.1. Décodage du fichier topologique 10](#_Toc198617743)

[7.3.2. Processus de chargement des données 10](#_Toc198617744)

[8. Brique de Décision 12](#_Toc198617745)

[9. Schéma des sous-briques de la décision 13](#_Toc198617746)

[10. MotionPlanner 14](#_Toc198617747)

[10.1. Entrées de la sous-brique 14](#_Toc198617748)

[10.2. Sortie de la sous-brique 15](#_Toc198617749)

[10.3. Description des principaux algorithmes 16](#_Toc198617750)

[10.3.1. Principe de la méthode de génération du profil de vitesse 16](#_Toc198617751)

[10.3.2. Suppression des cas dégénérés 17](#_Toc198617752)

[10.4. Conclusion 18](#_Toc198617753)

[10.5. Bibliographie 18](#_Toc198617754)

[11. DecisionOrchestrator 19](#_Toc198617755)

[11.1. Entrées de la sous-brique 20](#_Toc198617756)

[11.2. Sortie de la sous-brique 23](#_Toc198617757)

[11.3. Description des principaux algorithmes 25](#_Toc198617758)

[11.3.1. Diagramme d’état pour les validations et la surveillance 25](#_Toc198617759)

[11.3.2. Diagramme d’état principal 25](#_Toc198617760)

[11.3.1. Extracteur de Mission 28](#_Toc198617761)

[11.4. Conclusion 29](#_Toc198617762)

[12. Signature 29](#_Toc198617763)

# Caractéristique logiciel

|  |  |
| --- | --- |
| Fonction | FO1.4 v3 : Contrôler le véhicule sur les rails, les hubs et les plateformes de croisement |
| Nom du logiciel | Decision |
| Version | 1 |
| Dates des livrable | 12/15/2024 |
| Date de validation | 12/15/2024 |
| Utilisateur | Sicef |
| Equipe/service | AD |
| Responsables | B. YAHIAOUI, M. BOURKAIB |

# Description du document

Ce document présente les détails réalisés à partir de la spécification fonctionnelle FO1.4 concernant la Fonction Décision pour le POC dit de novembre. Il retrace les méthodes utilisées pour construire chaque sous-brique ainsi que les interfaces (entrés/sorties) détaillés.

# Environnement Software et Hardware

**Configuration du calculateur embarqué :**

* Modèle Ecotron EAXVA04 distribuée par FAAR
* Processeur Infineon TC297 (architecture TriCore) a 0.3GHz
* 728ko de mémoire RAM

**Logiciels pour intégration :**

* Simulink
* Compilateur architecture TriCore
* Produit Ecotron (EcoCal, EcoFlash et EcoCoder)

# Driver Kvaser USB-CAN Exigences implémentées

|  |  |  |
| --- | --- | --- |
| Exigences | Version : Pas de versionning (Décision v0.3) | Commentaires |
| EX- FP1.4.2 |  |  |
| EX- FS1.4.3.1 |  |  |
| EX- FS1.4.3.2 |  |  |
| EX- FP1.4.4 |  |  |
| EX- FS1.4.5.1 |  |  |
| EX- FS1.4.5.2 |  | C'est la loc qui le fait directement aujourd'hui (Sprint2) |
| EX- FS1.4.5.3 |  |  |
| EX- FS1.4.6 |  |  |
| EX- FS1.4.7.1 |  |  |
| EX- FS1.4.7.2 |  |  |
| EX- FS1.4.8.1 |  |  |
| EX- FS1.4.8.2 |  |  |
| EX- FS1.4.9.1 |  |  |
| EX- FS1.4.9.2 |  |  |
| EX- FP1.4.14 |  |  |
| EX- FP1.4.15 |  |  |
| EX- FP1.4.16 |  |  |
| EX- FP1.4.17 |  |  |
| EX- FP1.4.18 |  |  |
| EX- FP1.4.20 |  |  |
| EX- FP1.4.22 |  |  |
| EX- FP1.4.29 |  |  |

# Dispositions particulières

## Liste des paramètres nécessaires au lancement

Que ce soit pour une simulation ou pour une compilation pour flashage sur ECU, il est nécessaire de définir les paramètres suivants :

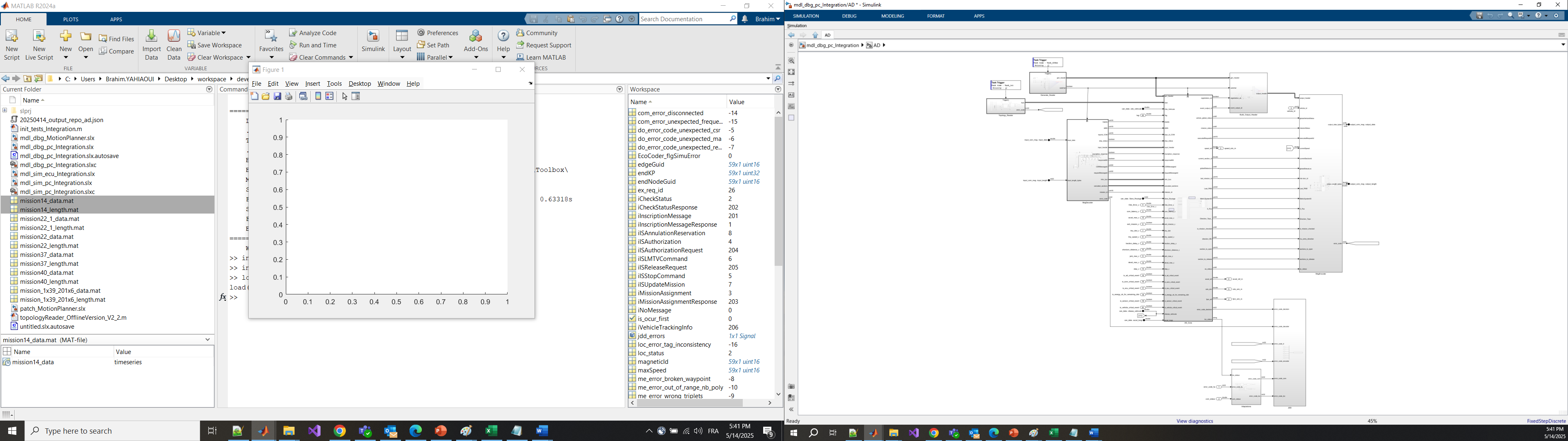
|  |  |  |
| --- | --- | --- |
| Nom | Valeurs | Description |
| **Constants** |  |  |
| null\_id | 65535 | L'identifiant concidéré comme l'élément nul |
| **Parameters** |  |  |
| msg\_version | 51 | Version du message |
| tcp\_msg\_size | 1330 | Taille maximale des trames tcp |
| nb\_max\_triplets | 32 | Nombre maximal de polylines |
| registration\_timeout | 1 | Délais de demande en boucle de l'inscription à la supervision |
| section\_entery\_timeout | 1 | Délais de demande en boucle d'autorisation pour occuper un section |
| nb\_max\_nodes\_mission | 100 | Nombre maximal de nœud dans une mission |
| nb\_max\_edges\_topo | 59 | Nombre maximal d'arc dans un fichier topo |
| nb\_poly\_sc | 16 | Nombre maximal de polynomes dans le MotionPlanner |
| nb\_corr\_sc | 1000 | Nombre maximal de corrections dans le cas d'une dégénéréscence d'une courbe dans le MotionPlanner |
| ex\_req\_id | 26 | Obsolète |
| loc\_status | 2 | Obsolète |
| **Precalculs** |  |  |
| nb\_max\_edges\_mission | 99 | nb\_max\_nodes\_mission - 1 |
| **Codes erreurs** |  |  |
| msg\_error\_invalid\_message | -1 | Impossible reçu est invalide ou peut être décalé. |
| msg\_error\_invalid\_message\_version | -2 | La version du message reçue est incompatible avec la version du decoder. |
| msg\_error\_unexpected\_request | -3 | Le message reçu est inattendue. |
| msg\_error\_unexpected\_response | -4 | La réponse à envoyer n'est pas connu par le module du encoder. |
| do\_error\_code\_unexpected\_csr | -5 | check status response dans le mission assignment est non compatible avec la dernière demande check status. |
| do\_error\_code\_unexpected\_ma | -6 | l'ouverture de section reçue est non compatible avec celle qui a été demandée. |
| do\_error\_code\_unexpected\_request | -7 | requète inattendue au moment où on attend une nouvelle mission |
| me\_error\_broken\_waypoint | -8 | chemin reçu par la supervision non compatible avec la topo dans le calculateur. |
| me\_error\_wrong\_triplets | -9 | les données triplets reçus sont invalides, incontinuité des triplets détectée. |
| me\_error\_out\_of\_range\_nb\_poly | -10 | Le nombre de polynome maximale prédéfinit ne suffit pas pour contenir toute la mission |
| mp\_error\_failed\_gen | -11 | Itération de correction maximale atteinte. Résultat invalide. |
| mp\_error\_failed\_pcs | -12 | Impossible de valider ou de produire des abscisses cuvilignes valides. Résultat invalide. |
| tr\_error\_wrong\_topo | -13 | Erreur dans la topologie au moment de la construction de la table des racourcis. |
| com\_error\_disconnected | -14 | Connexions non établie ou perdue. |
| com\_error\_unexpected\_frequency | -15 | Deux messages reçus à une fréquence plus petite que 100ms. |
| loc\_error\_tag\_inconsistency | -16 | Deux messages reçus à une fréquence plus petite que 100ms. |
| **Identifiant des messages** |  |  |
| iNoMessage | uint8(0) | Valeur nul pour les identifiants de message |
| **>Supervision** |  |  |
| iInscriptionMessageResponse | uint8(1) | Valeur correspondant à la spec de communication |
| iCheckStatus | uint8(2) | Valeur correspondant à la spec de communication |
| iMissionAssignment | uint8(3) | Valeur correspondant à la spec de communication |
| iISAuthorization | uint8(4) | Valeur correspondant à la spec de communication |
| iISStopCommand | uint8(5) | Valeur correspondant à la spec de communication |
| iISLMTVCommand | uint8(6) | Valeur correspondant à la spec de communication |
| iISUpdateMission | uint8(7) | Valeur correspondant à la spec de communication |
| iISAnnulationReservation | uint8(8) | Valeur correspondant à la spec de communication |
| **>AD** |  |  |
| iInscriptionMessage | uint8(201) | Valeur correspondant à la spec de communication |
| iCheckStatusResponse | uint8(202) | Valeur correspondant à la spec de communication |
| iMissionAssignmentResponse | uint8(203) | Valeur correspondant à la spec de communication |
| iISAuthorizationRequest | uint8(204) | Valeur correspondant à la spec de communication |
| iISReleaseRequest | uint8(205) | Valeur correspondant à la spec de communication |
| iVehicleTrackingInfo | uint8(206) | Valeur correspondant à la spec de communication |

## Descriptif des entêtes de messages

Les messages, quel que soit le type, ont un ensemble de champs commun qui sont les suivants : **TypeMessage**, **messageId**, **timeStamp** et **sequenceNumber**. L’idée dans la réalisation est d’unifier ces champs sous le même bus de données. Ce qui permet une modularité efficace du code développé. L’objet constituant dans l’ordre les champs est appelé **« header »**. Pour cela, on a remonté le besoin d’unifier les tailles des messagesId, on nous propose 24bits.

Il est à noter qu’un bloc spécifique permet aujourd’hui de générer l’entête de sortie en calculant champ par champ.

# Schéma Global du Module AD

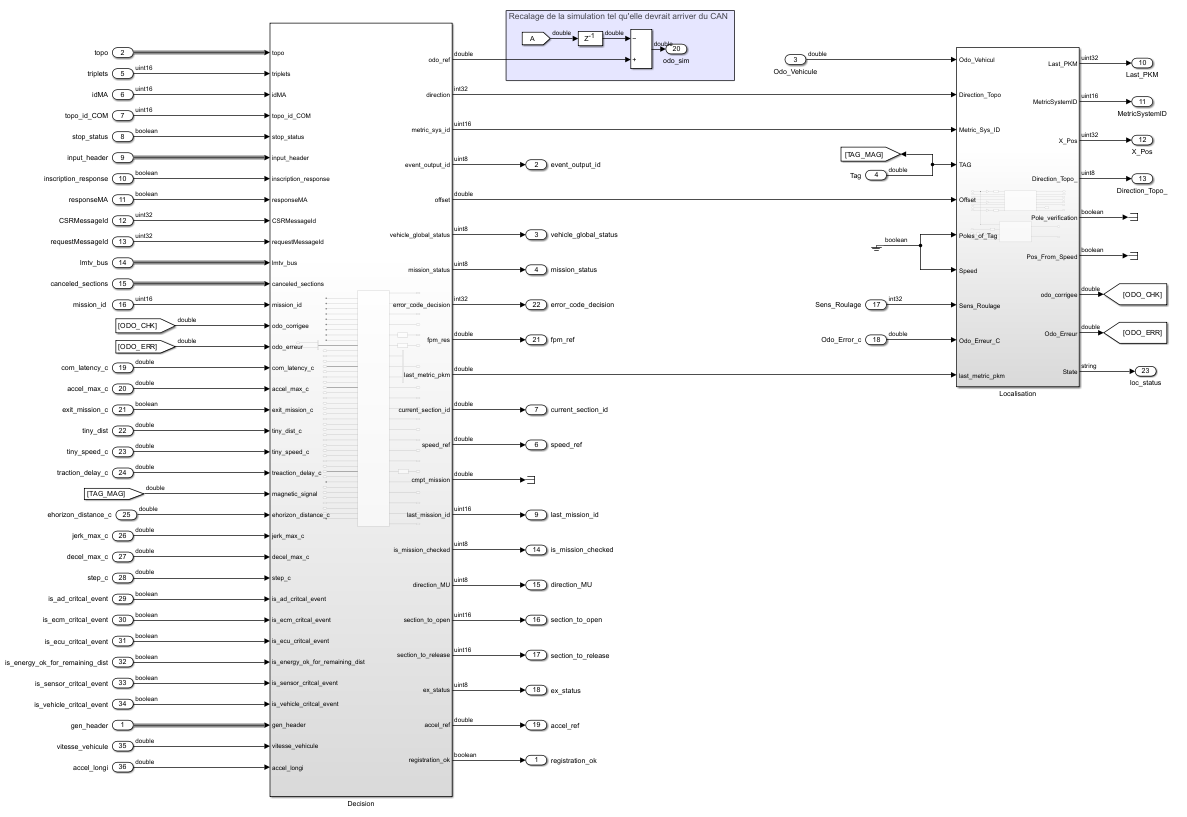


TopologyReader

AD\_Core

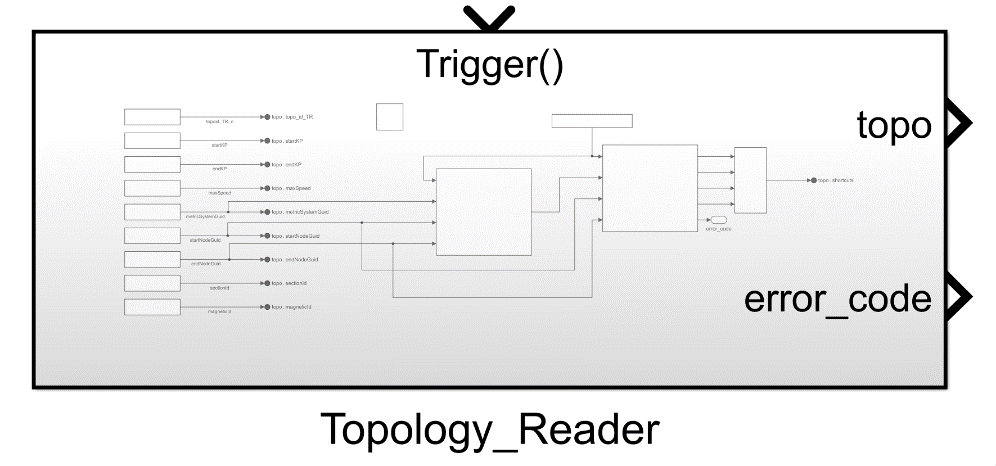
Modèle Simulink : vue globale

# Contenu du module AD\_Core



Modèle Simulink : Contenu AD\_Core

# Brique de TopologyReader



## Entrées de la sous-brique

Le **TopologyReader** est responsable du décodage, de l'interprétation et du stockage des données de topologie dans l'espace de travail (workspace). La version actuelle est une version hors ligne, car le décodage du fichier se fait hors ligne grâce à un script MATLAB. Ce script prend en entrée un fichier au format JSON et extrait ensuite les données. Il permet aussi de construire un tableau de raccourcis nécessaire à la construction des missions et à leurs mises à jour.

## Sortie de la sous-brique

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nom | Champs | Description | Type/taille | Destination | Unité et valeur(s) |
| topo |  | Données définissant la topologie complète d’une mission. | Bus | Decision |  |
|  | Topo\_id\_TR | Identifiant de la version topo. | uint16 |  | Sans unité, Identifiant |
|  | endKP | Vecteur colonne contenant les positions de fin de chaque arc | uint32, nombre max edge topo |  | mm, >=0 |
|  | startKP | Vecteur colonne contenant les positions de début de chaque arc | uint32, nombre max edge topo |  | Vecteur colonne, en mm |
|  | maxspeed | Vecteur colonne contenant les vitesses de chaque arc | uint16 , nombre max edge topo |  | Vecteur colonne, en Km/h |
|  | metricSystèmeGuid | Vecteur colonne contenant les identifiants des métriques systèmes de chaque arc | uint16 , nombre max edge topo |  | Sans unité, Identifiant |
|  | startNodeGuid | Vecteur colonne contenant les identifiants des nœuds de début de chaque arc | uint16 , nombre max edge topo |  | Sans unité, Identifiant |
|  | endNodeGuid | Vecteur colonne contenant les identifiants des nœuds de fin de chaque arc | uint16 , nombre max edge topo |  | Sans unité, Identifiant |
|  | sectionId | Vecteur colonne contenant les identifiants des sections de chaque arc | uint16 , nombre max edge topo |  | Sans unité, Identifiant |
|  | magneticId | Vecteur colonne contenant les identifiants des TAG de chaque arc | uint16 , nombre max edge topo |  | Sans unité, Identifiant |
|  | shortcuts | Tableau des racourcis défini dans la section [Processus de chargement des données](#_Processus_de_chargement) | Tableau de 4xnombre\_max\_d’arcs en uint16. |  | Sans unités, Identifiants de nœud et orientations (0, 1 ou 2) |
| error\_code |  | Renvoie une valeur négative si une erreur de topologie est détectée. Vaut 0 si tout se passe bien. | int32 |  | Sans unité, <=0 |

## Description des principaux algorithmes

### Décodage du fichier topologique

Le référentiel topologique est un fichier au format JSON qui décrit la carte sur laquelle le véhicule peut effectuer ses missions. Pour décoder ce fichier, la bibliothèque **Jsondecode** a été utilisée. Elle permet de décoder le fichier JSON et de créer une structure MATLAB au format .mat, qui sera ensuite utilisée pour extraire les données souhaitées.

### Processus de chargement des données

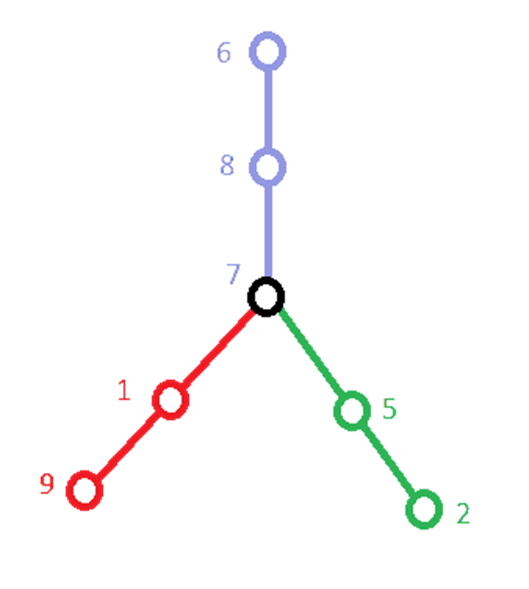
Après extraction, toutes les données présentées sous forme de chaînes de caractères sont chargées dans des tableaux constants tout au long de l’exécution. La brique qui correspond dans le diagramme Simulink s’exécute une seule fois, seulement au démarrage à la lecture du nouveau fichier contenant les informations topologiques.

Un tableau de raccourcis est constitué à ce moment-là. Il permet de récupérer les chemin complets des missions où à les mettre à jour dans la brique Decision. Le principe consiste utiliser ce tableau lors du balayage des arcs, lorsqu’on arrive à une bifurcation ou une fin de métrique par exemple, le tableau de raccourcis nous envoie vers le bon arc qui correspond pour assurer la continuité.

Plus précisément, l ’objectif est qu’on puisse relier un croisement, une fin ou un début de métrique système à un autre nœud si ce dernier existe. On va donc spécifier pour chaque ligne qui correspond à un arc sa correspondance, par défaut les valeurs sont à zéro si pas de correspondance. Ce tableau de raccourcis est un tableau qui a comme nombre de lignes le maximum de nombre d’arcs et chaque ligne est décrite ainsi :

1. Indice de la ligne à laquelle est raccordée le nœud de début de l’arc.
2. 1 signifie que le nœud de début de l’arc est raccordé aussi à un nœud de début d’arc, sinon s’il vaut 2, il est raccordé au nœud de fin.
3. Indice de la ligne à laquelle est raccordée le nœud de fin de l’arc.
4. 1 signifie que le nœud de fin de l’arc est raccordé à un nœud de début d’arc, sinon s’il vaut 2, il est raccordé au nœud de fin

Dans le cas d’un croisement étoilé en plusieurs branches, on stocke les raccourcis d’une manière circulaire. C’est à dire que lorsqu’on balaye les arcs, on va chercher premier arc qui est relié à lui, si celui-ci ne correspond pas au chemin qui nous intéresse, on regarde l’arc qui relie le dernier arc qu’on a trouvé. Et ainsi de suite jusqu’à trouve le chemin souhaité. Cela nous permet de garantir une taille statique du tableau des raccourcis.



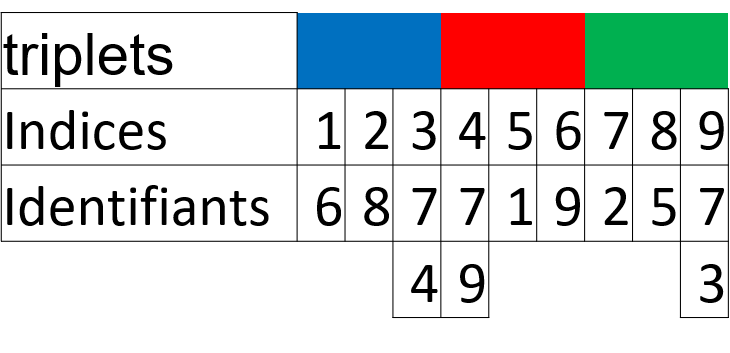
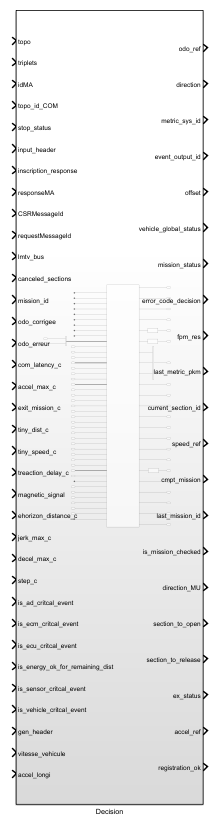


Illustration d’un remplissage d’un tableau de raccourcis - cas de bifurcation

Dans l’illustration ci-dessus, le nœud qui à l’identifiant 7 est un nœud de croisement (en particulier de bifurcation). Supposons qu’on arrive du nœuds 6 et qu’on veuille aller au nœud 2. En balayant 6, 8 puis 7, on trouve une correspondance pour l’indice de tableau 4 qui nous ramène au chemin rouge. On rejette donc ce chemin et on lit la correspondance qui correspond à l’identifiant 7 dans la partie rouge et qui nous donne l’indice 9. Celui-ci nous renvoie à l’identifiant 7 du chemin vert. On continu ainsi en balayant 7, 5 et 2 pour finir le parcours.

# Brique de Décision



# Schéma des sous-briques de la décision

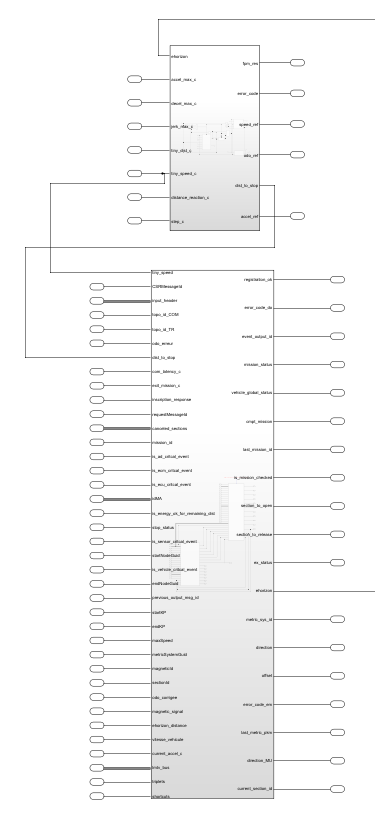
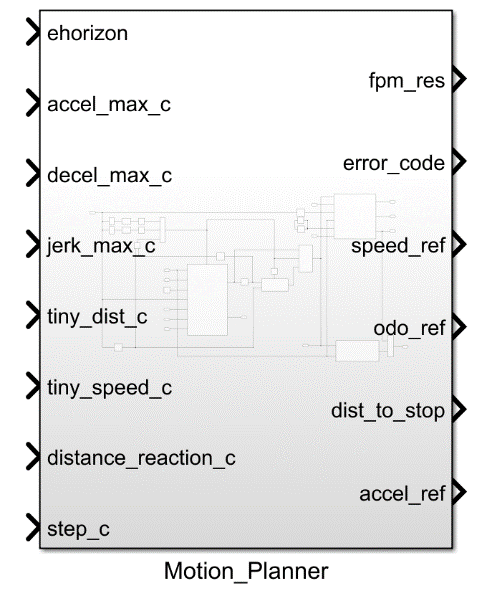


Schéma global à l’intérieur de la brique décision

# MotionPlanner



## Entrées de la sous-brique

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nom | Description | Type | Provenance | Unité et valeur(s) |
| ehorizon | Contient un ensemble de données permettant de construire le profil de vitesse sur un nombre fini de mètres.  Cette donnée est sous forme d’une matrice où chaque ligne est de la décrite comme suit :  **sB,sC,deg,b3,b2,b1,b0,aB**  avec **sB** est l’abscisse curviligne de debut de segment, **sC** est sa fin, **deg** degré de polynôme, **b3** coefficient de la puissance 3 du polynôme (respectivement pour **b2**, **b1** et **b0**) et **aB** l’accélération au début du segment. | Matrice double | MissionExtractor | Unité respectant [SI], valeurs réelles sauf pour le degré qui est un entier valant 0, 2 ou 3. |
| accel\_max\_c | Accélération maximale autorisée. | double | Calibration | m/s², >0 |
| decel\_max\_c | Décélération maximale autorisée. | double | Calibration | m/s², >0 |
| jerk\_max\_c | A-coup maximal autorisée. | double | Calibration | m/s3, >0 |
| tiny\_dist\_c | Distance négligeable. | double | Calibration | Mètre, >0 |
| tiny\_speed\_c | Vitesse négligeable. | double | Calibration | m/s, >0 |
| reaction\_delay\_c | Temps de réaction permettant le calcul de la vitesse de référence à suivre pas le pilote. | double | Calibration | s, >0 |
| step\_c | Pas en espace permettant d’estimer la distance à parcourir pendant le temps de réaction. Plus cette valeur est petite, plus on est précis mais plus on consomme du temps de calcul et vice versa. | double | Calibration | m, >0 |

## Sortie de la sous-brique

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nom | Description | Type | Destination | Unité et valeur(s) |
| fpm\_res | Définit une fonction polynomiale par morceau décrivant le profil de vitesse à suivre tout au long de l’ehorizon. Les polynômes sont définis par la même structure que la variable ehorizon ; c-à-d comme suit :  **sB,sC,deg,b3,b2,b1,b0,aB**  avec **sB** est l’abscisse curviligne de debut de segment, **sC** est sa fin, **deg** degré de polynôme, **b3** coefficient de la puissance 3 du polynôme (respectivement pour **b2**, **b1** et **b0**) et **aB** l’accélération au début du segment. | Matrice double | DEBUG | Unité respectant [SI], valeurs réelles sauf pour le degré qui est un entier valant 0, 2 ou 3. |
| error\_code | - Impossible de valider ou de produire des abscisses curvilignes valides. Résultat invalide.  - Génération des conditions aux limites impossibles. Résultat invalide. | Int32 | Brique de journalisations des codes erreurs | Sans unité, <=0 |
| speed\_ref | Vitesse recommandée tenant compte d’un temps de réaction. | double | Mesure | Simulation | Km/h, >=0 |
| odo\_ref | Simule l’odométrie courante à partir de la vitesse recommandée. | double | Simulation | Mètre, >=0 |
| dist\_to\_stop | Distance jusqu’à la position où doit décélérer le véhicule pour s’arrêter. | double | DecisionOrchestrator | Mètre, [0,+∞] |
| accel\_ref | Simule l’accélération à partir du profil de vitesse. | double | Simulation | m/s², nombre réel |

## Description des principaux algorithmes

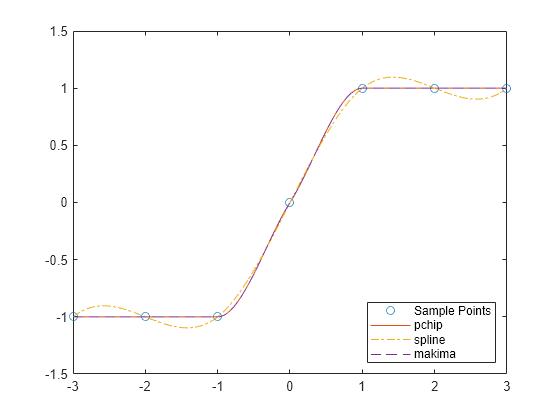
### Principe de la méthode de génération du profil de vitesse

Ce module correspond à un générateur de profil de vitesse qui permet d’éviter des "overshooting" pouvant être provoqué par le module de régulation. Il prend en entrée un ensemble de limitations défini à partir du fichier topo et de la mission envoyée par la supervision afin de générer une fonction polynomiale par morceau.

Notons , , et respectivement l’abscisse curviligne, la vitesse, l’accélération et l’à-coup. Le problème local d’interpolation se fait sur l’intervalle .

Les polynômes sont générés en utilisant les formules suivantes :

* L’interpolation d’Hermite [HERMITE] donne le polynôme d’ordre 3 : avec, selon si ou inconnu, la contrainte liée à l’accélération est donnée par et celle de l’à-coup maximale est .
* Polynôme d’ordre 2 par calcul : avec, selon si ou inconnu,
* Les polynômes d’ordre 0 pour les parties constantes



Comparaison des méthodes d’interpolation :

PCHIP (Piecewise Cubic Hermite Interpolating Polynomial),

spline et MAKIMA (Modified Akima piecewise cubic Hermite interpolation) [PCHIP]

### Suppression des cas dégénérés

Il se peut lorsqu’on souhaite générer une courbe entre deux limites de vitesse que la distance ne soit pas suffisante pour atteindre la seconde vitesse. Cela apparait lorsqu’on impose des contraintes d’accélération, de décélération ou d’à-coup plus fortes ou lorsque la condition initiale n’épouse pas exactement la courbe à l’approche du prochain morceau de polynôme.

Mettons l’hypothèse qu’on soit dans le cas de profile de vitesse sous forme d’une colline. Soit un segment de l’abscisse curviligne, l’abscisse curviligne à laquelle on atteint la vitesse associée au segment et l’abscisse curviligne à laquelle on commence à changer de vitesse pour atteindre la prochaine.

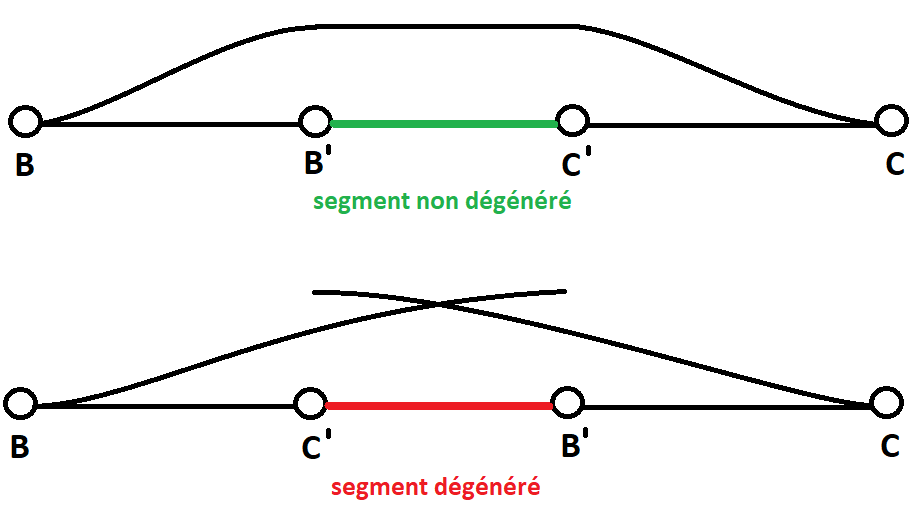


Illustration de changement des contraintes dynamiques

sur des longueurs similaires

La solution est de diminuer la vitesse limite afin que le profil puisse assurer sa continuité et ceux en tenant compte des contraintes. Plus précisément, on calcul un paramètre de dilation qu’on applique à la contrainte d’accélération pour trouver la valeur de la vitesse de correction.

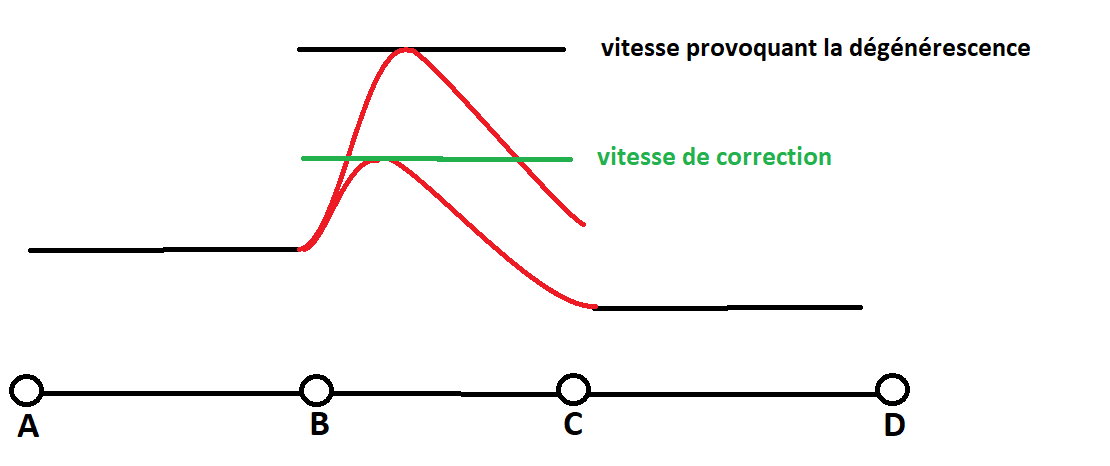


Illustration de la méthode de correction d’une dégénérescence

Dans le cas plus général, seuls les profils de type montée ou descente peuvent provoquer une dégénérescence. Nous considérons dans ces cas respectivement que ou .

Il faut noter cependant que changer la vitesse sur un segment impacte les segments voisins et peut ainsi causer de nouvelles dégénérescences. Pour cela l’algorithme est itératif et comme à chaque itération on baisse la vitesse, la convergence est assurée, donc la méthode est consistante.

## Conclusion

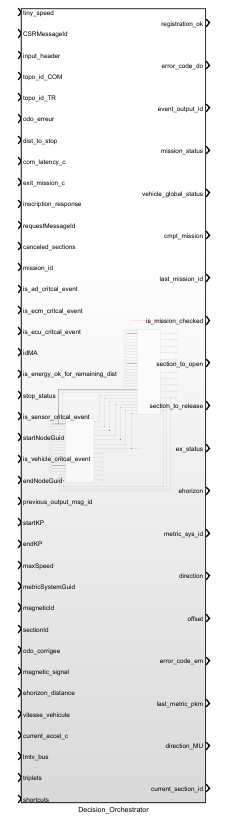
La construction du profil de vitesse passe par trois phases : L’analyse et correction, la construction des conditions aux limites pour chaque morceau et enfin l’interpolation polynômiale. Le profil résultant prend en compte la condition initiale et permet ainsi une adaptation temps réel pour que le véhicule ne fasse jamais de « overshooting » de la vitesse.

## Bibliographie

[HERMITE] <https://fr.wikipedia.org/wiki/Interpolation_d%27Hermite>

[PCHIP] <https://fr.mathworks.com/help/matlab/ref/pchip.html>

# DecisionOrchestrator



## Entrées de la sous-brique

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nom | Description | Type | Provenance | Unité et valeur(s) |
| tiny\_speed\_c | Vitesse négligeable. | double | Calibration | m/s, >0 |
| CSRMessageId | L’identifiant de message envoyé à la supervision au moment d’un CheckStatusResponse et retourné par le MissionAssignment pour assurer la traçabilité de la demande de mission. | uint16 | MsgDecoder | Sans unité, Identifiant de message |
| input\_header | Entête du message d’entrée. | Bus (voir header [Descriptif des entêtes de messages](#_Descriptif_des_entêtes)) | MsgDecoder |  |
| topoId\_COM | Identifiant du référentiel topologique utilisé par la Supervision | uint16 | MsgDecoder | Sans unité, Identifiant de topologie |
| Topo\_id\_TR | Identifiant de la version topo. | uint16 | TopologyReader | Sans unité, Identifiant |
| odo\_erreur | Erreur de localisation. | double | Localisation | Mètre, >= 0 |
| dist\_to\_stop | Distance jusqu’à la position où doit décélérer le véhicule pour s’arrêter. | double | MotionPlanner | Mètre, [0,+∞] |
| com\_latency\_c | Estimation d’un temps de vol total d’une requête du moment de l’envoie du calculateur embarqué jusqu’à la réception de sa réponse par la supervision. | double | Calibration | s, >0 |
| exit\_mission\_c | Si vaut 1, sort de la mission en court et revient à l’état initial. Remettre à 0 pour pouvoir affecter de nouvelles mission. | boolean | Calibration |  |
| inscription\_response | Vaut 1 si l’inscription vient d’être accordée par la supervision. | Boolean | MsgDecoder | Sans unité, 0 ou 1 |
| requestMessageId | L’identifiant de message envoyé à la supervision au moment d’une demande d’autorisation d’ouverture de section et retourné par ISAuthorization pour assurer la traçabilité de la demande de la demande. | Uint16 | MsgDecoder | Sans unité, Identifiant de message |
| canceled\_sections | Non utilisée. Incohérence remontée. | Bus{  uint16,  uint16,  uint16  } | MsgDecoder | Sans unités, identifiants de section |
| mission\_id | Identifiant de mission. | uint16 | MsgDecoder | Sans unité, identifiant de mission |
| is\_ad\_critical\_event | Si 1, alerte de la brique AD sur une situation critique. | boolean | Calibration | Sans unité, 0 ou 1 |
| is\_ad\_ecm\_event | Si 1, alerte de la brique mécanique sur une situation critique. Remplacement de ecm par meca au prochain sprint. | boolean | Calibration | Sans unité, 0 ou 1 |
| is\_ad\_ecu\_event | Si 1, alerte de la brique ECU sur une situation critique. | boolean | Calibration | Sans unité, 0 ou 1 |
| idMA | Identifiant de la section pour laquelle le véhicule demande l’autorisation et la réponse de la supervision. Si cette réponse vaut 1, on peut occuper la section. | Bus {  uint16,  boolean  } | MsgDecoder | Sans unités, Identifiant de section et 0 ou 1 |
| is\_energy\_ok\_for\_remaining\_ dist | Vaut 1 lorsqu’il y a assez d’energie (batterie) pour finir la mission | boolean | Calibration | Sans unité, 0 ou 1 |
| stop\_status | 1 si l’arrêt, 0 sinon. | boolean | MsgDecoder | Sans unité, 0 ou 1 |
| is\_sensor\_critcal\_event | Si 1, alerte d’un capteur sur une situation critique. | boolean | Calibration | Sans unité, 0 ou 1 |
| startNodeGuid | Liste ordonnée de tous les identifiants de nœuds en début de segment chargés du fichier de topo. | Matrice uint8 | TopologyReader | Sans unité, Id Nœud |
| is\_vehicle\_critcal\_event | Si 1, alerte du véhicule sur une situation critique. | boolean | Calibration | Sans unité, 0 ou 1 |
| endNodeGuid | Liste ordonnée de tous les identifiants de nœuds en fin de segment chargés du fichier de topo. | Matrice uint8 | TopologyReader | Sans unité, Id Nœud |
| startKP | Liste ordonnée de tous les pkm de nœuds en début de segment chargés du fichier de topo. | Vecteur double | TopologyReader | mm, >=0 |
| endKP | Liste ordonnée de tous les pkm de nœuds en fin de segment chargés du fichier de topo. | Vecteur double | TopologyReader | mm, >=0 |
| maxSpeed | Liste ordonnée de toutes les vitesses limites des segments chargés du fichier de topo. | Vecteur double | TopologyReader | Km/h, >0 |
| metricSystemGuid | Liste ordonnée de tous les noms de système métrique par segments chargés du fichier de topo. | Matrice uint8 | TopologyReader | Sans unité, Id système métrique |
| magneticId | Liste ordonnée de tous les identifiants de nœuds correspondant à un tag magnétique et en début de segment chargés du fichier de topo. Si le nœud ne correspond pas à un tag magnétique, la valeur de la cellule est nulle. | Matrice uint8 | TopologyReader | Sans unité, Id nœud ou 0 |
| odo\_corrigee | Odométrie corrigée cohérente avec la localisation dans la supervision. | double | Localisation | Mètre, >= 0 |
| magnetic\_signal | Vaut 1 pendant qu’un tag magnétique est détecté. | boolean | ADC/A14 avec transformation en boolean | Sans unité, 1 ou 0 |
| ehorizon\_distance\_c | Distance maximum du e-horizon. | double | Calibration | mètre, >0 |
| vitesse\_vehicule | Vitesse longitudinale du véhicule. | double | CAN/HS1 | Km/h, >=0 |
| current\_accel\_c | Accélération longitudinale du véhicule. Changer pour le prochain sprint le nom car ce n’est pas une calibration. | double | CAN/HS1 | m/s², Nombre réel |
| lmtv\_bus | Données pour limiter la vitesse sur un intervalle définies par la supervision. Cette sortie est composée deux trois données : la vitesse de limitation, l’identifiant du nœud de départ de la limitation et celui du nœud de la fin. | Bus{  uint16,  uint16,  uint16  } | MsgDecoder | (Km/h,Sans Unité, Sans unité), >0, Identifiant de nœud, Identifiant de nœud |
| triplets | Défini un chemin décomposé en un ensemble de « sous-chemins » appelés triplets. Chaque triplet correspond à trois identifiants de nœud : le premier, le second et le dernier. Le triplet a pour particularité de ne contenir de nœud de bifurcation qu’a ses bords. | Tableau de 3x32 en uint16. | MsgDecoder | Sans unités, Identifiants de nœuds |
| shortcuts | Tableau des racourcis défini dans la section [Processus de chargement des données](#_Processus_de_chargement) | Tableau de 4xnombre\_max\_d’arcs en uint16. | TopologyReader | Sans unités, Identifiants de nœud et orientations (0, 1 ou 2) |

## Sortie de la sous-brique

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nom | Description | Type | Destination | Unité et valeur(s) |
| registration\_ok | Vaut 1 lorsque le véhicule souhaite s’inscrire dans la supervision. | boolean | MsgEncoder | Sans unité, 1 ou 0 |
| code\_error\_do | - une requête inattendue a été reçu.  - véhicule est en incapacité d’effectuer la mission | int32 | Brique de journalisations des codes erreurs | Sans unité, <=0 |
| event\_output\_id | Type de requête à envoyer à la supervision. | uint16 | MsgEncoder | Sans unité, exemple : "MissionAssignmentRespose" |
| mission\_status | Si vaut 1 : Si une mission est en cours. | boolean | MsgEncoder | Sans unité, 1 ou 0 |
| vehicle\_global\_status | Vaut true si le véhicule est opérationnel. | boolean | MsgEncoder | Sans unité, 1 ou 0 |
| cmpt\_mission | Compteur de mission depuis le lancement du programme. | int32 | Non Utilisé | Sans unité, >=0 |
| last\_mission\_id | Identifiant de la dernière mission effectué. | uint16 | MsgEncoder | Sans unité, Id Mission |
| is\_mission\_checked | Vaut 1 lorsque le véhicule est apte à effectuer la mission demandée par la supervision. | boolean | MsgEncoder | Sans unité, 1 ou 0 |
| section\_to\_open | Identifiant de la section à demander à occuper. | uint16 | MsgEncoder | Sans unité, Id Section ou 0 |
| section\_to\_release | Identifiant de la section à libérer. | uint16 | MsgEncoder | Sans unité, Id Section ou 0 |
| ex\_status | Manque d’explication pour cette variable. Demande remontée. | uint8 | MsgEncoder | Sans unité, ND |
| ehorizon | Contient un ensemble de données permettant de construire le profil de vitesse sur un nombre fini de mètres.  Cette donnée est sous forme d’une matrice où chaque ligne est de la décrite comme suit :  **sB,sC,deg,b3,b2,b1,b0,aB**  avec **sB** est l’abscisse curviligne de debut de segment, **sC** est sa fin, **deg** degré de polynôme, **b3** coefficient de la puissance 3 du polynôme (respectivement pour **b2**, **b1** et **b0**) et **aB** l’accélération au début du segment. | Matrice double | MotionPlanner | Unité respectant [SI], valeurs réelles sauf pour le degré qui est un entier valant 0, 2 ou 3. |
| metric\_sys\_id | Identifiant du système métrique en cours. | string | Localisation | Sans unité, id système métrique |
| direction | Direction du véhicule dans le repère topologique. | int32 | Localisation | Sans unité,  1 : sens de la topologie  2 : sens contraire par rapport à la topologie  0 : sinon |
| offset | Valeur de correction de la localisation dans le repère topologique. | double | Localisation | Mètre, >=0 |
| code\_error\_em | Vaut :  0 si pas d’erreur détecté.  -1 si le nombre maximal de changement de vitesse est dépassé.  -2 si on n’arrive pas à extraire la suite d’identifiant de nœuds. | int32 | Brique de journalisations des codes erreurs | Sans unité, <=0 |
| last\_metric\_pkm | Limite maximale à ne pas dépasser par la localisation dans une métrique. | double | Localisation | Mm, >=0 |
| direction\_MU | Direction topologique dans la prochaine section ferrmée. | uint8 | MsgEncoder | Sans unité,  1 : sens de la topologie  2 : sens contraire par rapport à la topologie  0 : sinon |
| current\_section\_id | Nom de la section courante. Vaut zéro si pas de section. | int16 | DecisionOrchestrator | Sans unité, Id Section ou 0 |

## Description des principaux algorithmes

### Diagramme d’état pour les validations et la surveillance

Ce diagramme a pour but de vérifier si une mission ou une mise à jour de celle-ci est opérationnelle et que le véhicule a la capacité de pourvoir répondre à la demande.

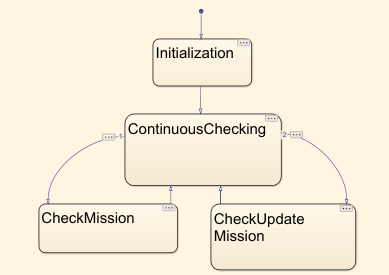


Diagramme d’état de validation et de surveillance

Juste après une initialisation, on rentre dans une vérification en continu des composants :

* Energy restante par rapport à la distance
* Signalement critiques de l’ECU (limites atteintes du calculateur, anomalie)
* Signalement critiques des briques AD (Impossibilité de traiter la demande, problème de COM)
* Signalement critiques de la brique ECM (dysfonctionnement, anomalie)
* Signalement critiques du Véhicule (état des pneu, AEB)
* Signalement critiques d’un capteur (dysfonctionnement du capteur d’alignement ou autre)
* Incompatibilité du référentiel topologique avec celui de la supervision

Quand on reçoit une nouvelle mission on vérifie en plus si la vitesse est nulle. En outre, dans le cas d’une mise à jour de mission on vérifie si celle-ci est cohérente et on vérifie aussi si nous ne sommes pas dans une partie où on est en train de demander l’ouverture d’une section. On considère qu’il est trop tard de demander une quelconque modification de parcours de la mission à ce moment-là pour éviter toute demande tardive.

### Diagramme d’état principal

Ce bloque interagit avec le reste afin d’assurer un bon enchainement. Il s’assure que les demandes provenant de la supervision sont bien cohérentes avec les données stockées dans le véhicule et que la suite de cette demande est bien dans l’ordre. Il contribue aussi à l’ATP en signalant qu’il souhaite occuper ou libérer une section selon les cas.

Les fonctionnalités implémentées sont les suivantes :

* Il permet de lancer une mission et d’envoyer au module de communication que la mission est en cours ou pas.
* Il prend en compte le stop command.
* Demande d’entrée dans une nouvelle section à une distance où est la distance à laquelle le véhicule doit commencer à s’arrêter, la vitesse courante, la latence de la communication avec la supervision et l’erreur courante de la localisation.
* Demande de libération de section si la distance depuis que la dernière section a été quittée est plus grande que l’erreur courante de la localisation.

Afin d’enchainer les fonctionnalités prévues dans ce module (calculés à l’intérieur ou à l’extérieur de ce dernier), on applique un diagramme d’état assurant la cohérence des décision prise par cette sous-brique.

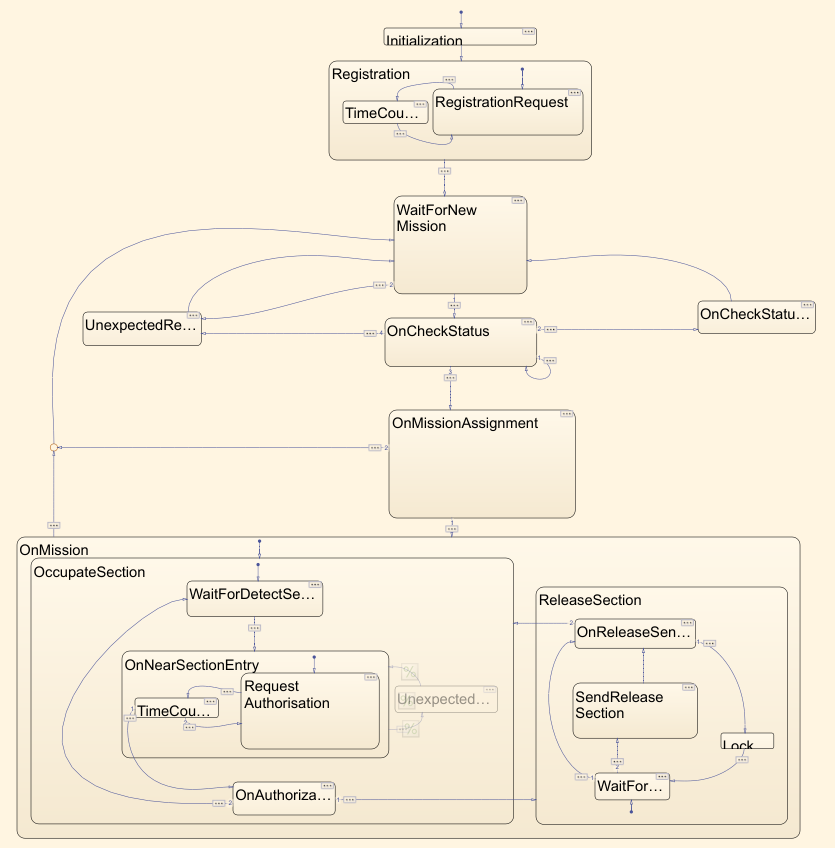


Diagramme d’état de la sous-brique DecisionOrchestrator

Dans ce diagramme, Le premier état est celui de "Initialisation" qui permet d’initialiser les variables dans le scope du cycle de vie du stateflow, tandis que l’état qui suit "WaitForNewMission" réinitialise les variables dépendant du cycle de vie d’une mission. L’état "UnexpectedRequest" est invoqué lorsque la demande sollicitée par la supervision est inattendue, de même pour l’état "OnCheckStatus" qui correspond à une demande de la supervision pour valider l’acceptabilité de la mission. Si la validation n’aboutit pas, on invoque l’état "OnCheckStatusFailed". Dans le cas où on reçoit une nouvelle demande de la part de la supervision, pendant qu’on est encore dans l’état "OnCheckStatus", et que cette demande comporte une autre validation, on réitère les opérations liées à cet état (on considère ou bien qu’une nouvelle mission soit demandée, ou bien que la réponse de la validation précédente ne soit pas arrivée chez la supervision). On continue après l’état "OnCheckStatus" seulement si la validation a abouti et qu’une nouvelle mission a été affectée. Dans ce cas on passe à l’état "OnMissionAssignment" afin de répondre à la supervision qu’on a bien pris en compte l’affectation de la mission et on passe tout de suite après à l’état "OnMission". On sort de cet état si on invoque par le biais de la calibration- la sortie de mission (debug) ou si on vérifie les trois conditions suivantes :

* On est sur le dernier segment de la mission
* La vitesse est inférieure à un seuil assez petit pour décider que le véhicule est arrêté
* Le véhicule est à une distance inférieure à où est la distance à laquelle le véhicule doit s’arrêter et l’erreur courante de la localisation.

L’état "OnMission" contient des sous-états qui permettent d’échanger avec l’ATP pour l’occupation ou la libération des sections. On dispose dans un premier niveau deux état :

* "OccupateSection" : qui gère l’entrée en section. C’est l’état initial d’une mission car dans le cas où on ne se trouve pas dans une section, on aura forcément au moins une section a occuper si celle-ci fait partie de la mission. Le second cas qui reste est lorsqu’on reçoit une mission alors qu’on se trouve dans une section. Dans ce cas la convention est de demander à la supervision l’occupation de la section en cours.
* "ReleaseSection" : qui permet de gérer la libération d’une section

Le passage d’un état à un autre se fait selon les conditions suivantes :

* **"OccupateSection" à " OccupateSection " :** si le véhicule se trouve dans un segment qui ne contient pas de section à libérer et qu’il y a prochainement une section à occuper.
* **"OccupateSection" à "ReleaseSection" :** si on a reçu l’autorisation d’occuper la prochaine section et si ensuite le véhicule est rentré dans cette section.
* **"ReleaseSection" à " ReleaseSection" :** si le véhicule se trouve dans une section à libérer et qu’il n’y a prochainement pas de section à occuper.
* **"ReleaseSection" à "OccupateSection" :** Si on ne se trouve pas dans la condition du passage "ReleaseSection" à " ReleaseSection".

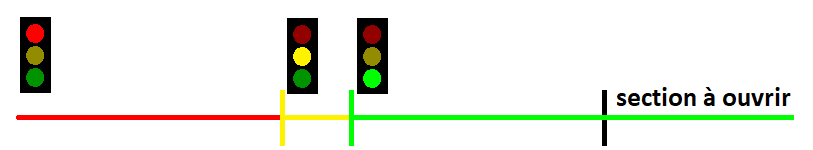


Illustration d’ouverture de section avec une réponse positive

Attention, le jaune signifie ici qu’une demande a été envoyé et qu’on peut passer ensuite au rouge comme au vert.

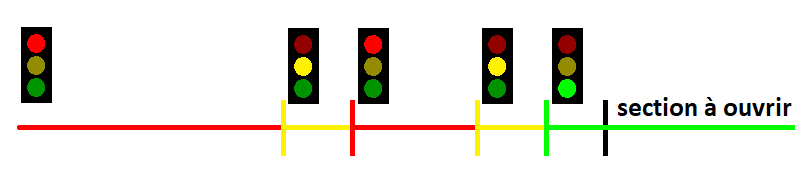


Illustration d’ouverture de section avec une réponse négative suivi d’une réponse positive  
La redemande est périodique et contrôlé par le paramètre section\_entery\_timeout

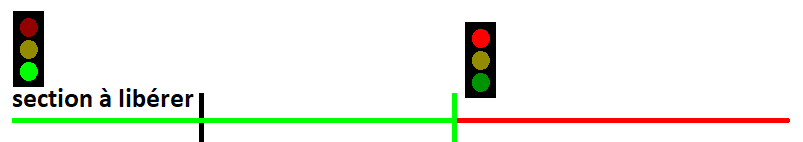


Illustration de la libération d’une section

Il est a noté que dans l’état "OccupateSection", on envoie en boucle la demande d’occupation, selon un pas de temps prédéterminé, tant qu’on n’a pas reçu l’autorisation d’occuper la nouvelle section. A contrario, l’état "ReleaseSection" permet d’envoyer le message une seule fois, une fois la section libérée.

### Extracteur de Mission

Afin que les sous-briques de décision et la brique de la Localisation puissent exploiter les données provenant simultanément de la lecture du fichier de topologie ainsi que les données provenant de la communication, un algorithme a été prévu pour associer les données en prenant en compte le sens de la mission défini par la supervision et récupère le reste des informations de la partie topologie. L’algorithme effectue pour cela une réorientation des arcs dans le sens de la mission et prépare des données aux autres sous bloques de la décision et la brique de localisation : e-horizon, nom de la section à ouvrir, systèmes métrique courant…etc.

Afin de reconstruire le chemin complet de la mission, l’extracteur de mission va utiliser le tableau de raccourcis défini dans la partir TopologyReader et les triplets qui proviennent de la supervision. Un triplet un sous ensemble d’un chemin de mission qui ne dispose de nœud de croisement (nombre de connections supérieur à 2) que sur éventuellement ses bords. Il est défini par trois nœud :

* Nœud de début : permet d’avoir un nœud de départ.
* Nœud suivant celui du début : celui-ci permet de définir l’orientation.
* Nœud de fin : celui la permet de s’arrêter à la fin d’un triplet.

Une mission doit être entièrement définie par un ensemble de triplets.

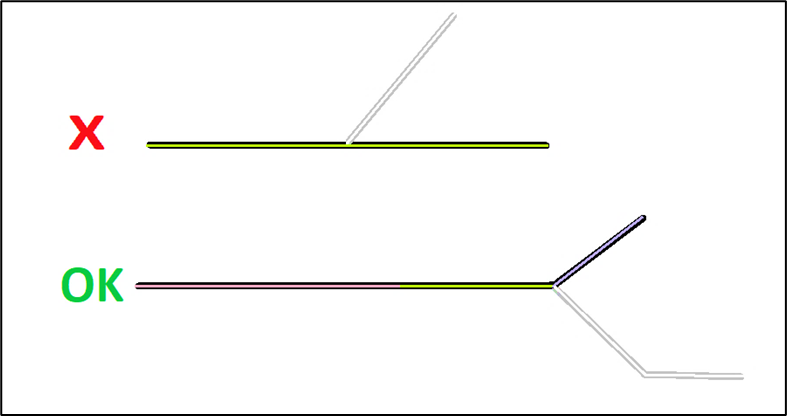


Illustration de ce qui est considéré comme un triplet  
En couleurs les triplets et en noir ou gris les systèmes métriques

Cette brique adapte selon l’exploitation à mené par la suite les données :

* Elle calcul le chemin à partir de triplets.
* Elle s’occupe de la mise à jour de la mission.
* Dans le cas d’une commande stop, elle va annuler toutes les vitesses de l’e-horizon qui peut produire par la suite un profil de vitesse cohérent avec la situation.
* Quand une section est fermée elle annule les vitesses dans les e-horizon à partir de cette section.
* Elle va préparer un e-horizon valide pour le MotionPlanner afin d’éliminer les intervalles successifs qui ont la même vitesse.
* Elle produit la liste de TAG à balayer dans l’ordre pour le module de localisation.
* Elle fournit le système métrique et l’offset de calcul de la localisation courants.
* Elle Applique la limitation de vitesse.
* Elle calcule des données nécessaires à l’ouverture et la fermeture de sections.
* Elle contribue à définir la fin de mission.

Lorsqu’on reçoit une nouvelle mission, le véhicule est à l’arrêt, les missions sont entièrement chargées avant que le véhicule ne démarre. Dans le cas d’une mise à jour de mission, on ne reçoit que la partie à mettre à jour. L’algorithme est prévu pour ne mettre à jour qu’une partie de la mission pour optimiser le temps de calcul lorsque le véhicule roule à des vitesses de croisière.

## Conclusion

Cette brique est une brique de haut niveau, elle contient des stateflows qui permettent de piloter et un extracteur de mission qui adapte les données, ce qui la rend facile à maintenir. En même temps c’est une partie qui contient des algorithmes cohérents avec les autres composants qu’elle soit à intérieur de l’AD (comme la Localisation et le MotionPlanner) ou à l’extérieur (comme la supervision).

# Signature

|  |  |  |  |
| --- | --- | --- | --- |
| Prénom et nom | Version | Date | Signature |
| Brahim YAHIAOUI | 1 | 01/10/2025 |  |
|  |  |  |  |