

# Défi Signal, Système & Simulation

Projet pipeau

2021-2022

Henri Bauer

[henri.bauer@ensma.fr](mailto:henri.bauer@ensma.fr)

Brice Chardin

[brice.chardin@ensma.fr](mailto:brice.chardin@ensma.fr)

Michaël Richard

[michael.richard@ensma.fr](mailto:michael.richard@ensma.fr)

Frédéric Ridouard

[frederic.ridouard@ensma.fr](mailto:frederic.ridouard@ensma.fr)



# Table des matières

Table des matières	iii
<b>1 Compétition D3S</b>	<b>1</b>
1.1 Objectifs et organisation . . . . .	1
1.2 Architecture . . . . .	2
1.3 Tâches à réaliser . . . . .	3
1.4 Rendu . . . . .	3
1.5 Évaluation . . . . .	3
<b>2 Environnement technique</b>	<b>4</b>
2.1 Connexion en SSH . . . . .	4
2.2 Utilisation du terminal . . . . .	4
2.3 Échange de fichiers . . . . .	5
<b>3 Format d'échange des messages</b>	<b>7</b>
3.1 Format des trames . . . . .	7
3.2 Identifiant des notes . . . . .	8
3.3 Chiffrement des transmissions . . . . .	9
<b>4 Analyse du son</b>	<b>10</b>
<b>5 Simulateur</b>	<b>11</b>





# 1 Compétition D3S

## 1.1 Objectifs et organisation

L'objectif de ce projet est de concevoir et implémenter un rover capable d'avancer *à l'oreille*, c'est-à-dire en suivant des instructions qui lui sont transmises par l'intermédiaire de signaux sonores – en l'occurrence des séquences de notes de musiques.

Ce projet forme un cadre de travail collaboratif par demi-groupe de TP, sur quatre séances encadrées de trois heures chacune. Le matériel utilisé pourra également être mis à disposition sur demande afin de vous permettre d'avancer entre les séances.

Chaque groupe possède une liste de séquences de notes qui lui est propre, c'est-à-dire que les différents groupes n'ont pas tous les mêmes correspondances entre notes et instructions. Chaque séquence de notes est associée à une puissance moteur maximale (pouvant être négative), consigne que votre rover devra respecter si la séquence musicale est jouée. L'objectif est alors de faire avancer le rover le plus rapidement possible tout en respectant ces contraintes. Une séquence consiste en une ou plusieurs notes jouées successivement, sans note intermédiaire additionnelle – mais des silences peuvent être présents.

Si l'avancement des différents groupes s'y prête, une compétition amicale entre les quatorze demi-groupes de la promotion sera organisée en fin de projet (la date et les modalités seront précisées ultérieurement).

**Objectifs pédagogiques** Ce projet fait le lien entre les différentes matières de première année portant sur le traitement numérique de l'information au sens large. Il permet de mettre en pratique les notions abordées au cours des différentes matières dans un cadre applicatif unique – et, il faut l'avouer, un peu pipeau.

## 1.2 Architecture

La figure 1.1 donne une vue d'ensemble des composants logiciels et matériels du projet et de leurs interactions.

Chaque groupe devra donc faire interagir, au minimum :

- une Raspberry Pi (A) commune à l'ensemble des groupes, sur laquelle le microphone USB sera branché et hébergeant les différents programmes de traitement des signaux sonores,
- un rover PiCar,
- une Raspberry Pi (B), propre à votre groupe, hébergeant votre simulateur,
- une Raspberry Pi hébergeant l'arbitre (qui sera fourni et n'est donc pas à développer).

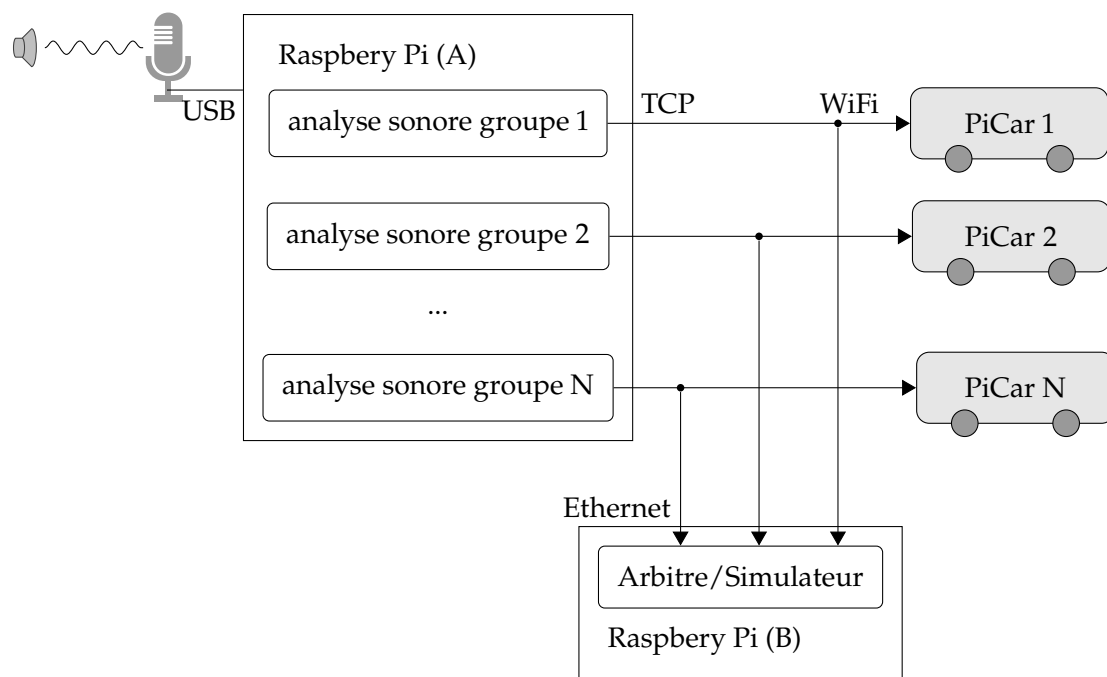


FIGURE 1.1 – Schéma global de l'architecture physique et des communications

## 1.3 Tâches à réaliser

Cinq grandes parties ont été identifiées pour ce projet.

1. Analyser le signal sonore issu du microphone pour détecter des séquences de notes (voir chapitre 4).
2. Définir le format de transmission de l'information entre les différents composants logiciels et implémenter les fonctions de transfert des données (voir chapitre 3).
3. Implémenter un simulateur permettant de visualiser le bon fonctionnement de l'ensemble, en particulier lorsque vous n'aurez pas de PiCar sous la main (voir chapitre 5).
4. Implémenter le contrôle du PiCar avec suivi de piste ou de cible lumineuse (à déterminer par la suite du projet) en fonction des ordre reçus depuis l'analyse sonore.
5. Sélectionner les partitions qui seront jouées lors de la compétition, en privilégiant celles qui favoriseront votre groupe par rapport aux séquences de notes qu'elles contiennent.

À l'exception de la tâche numéro 5 qui débutera un peu plus tard dans ce projet, les travaux de développement peuvent être menés en parallèle dès la première séance. La répartition du travail entre vous est laissée libre.

## 1.4 Rendu

Le rendu attendu pour ce projet est une archive contenant :

- l'intégralité des réalisations logicielles,
- un court document listant les contributions de chaque participant.

La compétition prévue en fin de projet servira également de démonstrateur.

## 1.5 Évaluation

Les principaux critères d'évaluation du projet sont :

- votre présence et implication lors des séances encadrées,
- la qualité des réalisations,
- le respect des bonnes pratiques de programmation, telles que vues dans les cours d'ASN et de BCL.

Le système d'exploitation installé sur les Raspberry Pi est Raspberry Pi OS Lite bullseye (64-bit).

Les associations entre adresses MAC et adresses IP sont les suivantes<sup>1</sup> :

1. dc:a6:32:65:7d:23 → 193.55.163.68
2. dc:a6:32:65:43:7e → 193.55.163.69
3. b8:27:eb:36:70:ca → 193.55.163.71
4. dc:a6:32:65:7d:32 → 193.55.163.74
5. dc:a6:32:65:7d:3e → 193.55.163.75
6. dc:a6:32:65:7c:a8 → 193.55.163.76
7. e4:5f:01:5f:79:f4 → 193.55.163.81 (microphone)
8. dc:a6:32:65:7c:63 → 193.55.163.83
9. b8:27:eb:fa:d7:1f → 193.55.163.84

1: Ces adresses ne sont accessibles que depuis le réseau local de l'ENSMA.

## 2.1 Connexion en SSH

Pour ce projet, l'accès aux Raspberry Pi sera réalisé avec SSH (*Secure Shell*). Le nom de l'unique utilisateur configuré par défaut est pi, avec le mot de passe raspberry.

```
1 | ssh pi@<ip_address>
```

Pour pouvoir utiliser des applications avec interface graphique (par exemple xclock), il faudra utiliser un client SSH intégrant un *serveur* X, par exemple MobaXterm (fourni dans l'archive [MobaXterm\\_v8.6.zip](#) sur moodle). Ce type de déport d'affichage pourra éventuellement servir pour le développement du simulateur (voir chapitre 5).





### Conventions d'écriture

Dans ce document, les paramètres entre chevrons <...> représentent des valeurs à remplacer avant l'exécution d'une commande. Les deux chevrons ne doivent pas être inclus. Un exemple de remplacement ici serait :



```
ssh pi@193.55.163.68
```

## 2.2 Utilisation du terminal

Les interactions avec la Raspberry Pi passant principalement par le terminal, cette section en présente les bases du fonctionnement.

Les flèches directionnelles ( et ) permettent de naviguer dans l'historique des commandes exécutées. La touche  permet de compléter automatiquement le nom d'une commande ou d'un fichier. Lorsque plusieurs solutions existent, l'appui une seconde fois sur la touche  permet de les lister.

### Exemple d'auto-complétion

L'auto-complétion de pyth donne python.   renseigne la liste des commandes existantes possédant ce préfixe (python, python3, etc.).



La combinaison **Ctrl** + **R** (*reverse-search*) permet de rechercher dans l'historique de commandes. Cette historique est conservée entre les différentes sessions. La répétition des touches **Ctrl** + **R** remonte dans cette historique en sélectionnant la commande correspondante précédente.

La combinaison **Ctrl** + **C** (*cancel*) annule :

- une commande en cours d'écriture (plutôt que de tout effacer)
- une commande en cours d'exécution (par exemple `sleep 10` si vous souhaitez tester ce fonctionnement)

La combinaison **Ctrl** + **D** (*end-of-file* ou EOF) peut être utilisé pour mettre fin à une session interactive, par exemple pour les commandes `python3` ou `ssh`.

La commande `man` (pour *manual*) donne la documentation associée à une commande ou une bibliothèque. La navigation dans cette aide se fait avec les touches de défilement (**PgUp** et **PgDown**), et **Q** pour quitter.

```
1 | man scp
```

**Ctrl** + **D**

Vous pouvez ainsi essayer d'exécuter puis de quitter l'interpréteur Python.

## 2.3 Échange de fichiers

### Secure Copy (scp)

La commande `scp` permet de copier un ou plusieurs fichiers depuis ou vers une machine distante.

Par exemple, la commande ci-dessous copie tous les fichiers finissant par `.wav` du répertoire courant vers le répertoire `~/sound_files` de la machine distante.

Le répertoire cible `~/sound_files` doit être créé au préalable si nécessaire (à l'aide de la commande `mkdir`).

```
1 | scp *.wav pi@<ip_address>:~/sound_files
```

#### Répertoire personnel

Le symbole tilde (`~`) désigne le répertoire personnel (*home*) de l'utilisateur courant, synonyme ici de `/home/pi`.

### Montage d'un sous-répertoire de Z :

La commande `mount` ci-dessous permet de rendre accessible votre répertoire Z: depuis la Raspberry Pi, dans le répertoire local `/home/pi/<target_dir>`. Il vous faudra remplacer les deux `<username>` de cette commande par votre login ENSMA (généralement votre nom suivi de l'initiale de votre prénom), et créer au préalable le répertoire `<target_dir>`.

#### sudo

La commande `sudo` signifie *superuser do*. Elle permet d'exécuter une commande en tant qu'administrateur.

```
1 | sudo mount -t cifs //DataEtu/<username> /home/pi/<target_dir>/
2 | -o vers=2.0,username=<username>,rw,uid=$(id -u),gid=$(id -g)
```

Comme ce répertoire est accessible par n'importe qui ayant accès à la Raspberry Pi – c'est-à-dire tout le monde à l'ENSMA ou presque – en lecture comme en écriture, il est (fortement) recommandé de créer un sous-répertoire spécifique au projet (<path> dans les commandes suivantes), et de ne monter que celui-ci.

Il faudra, au préalable, annuler le montage réalisé précédemment à l'aide de la commande `umount` :

```
1 | sudo umount /home/pi/<target_dir>

1 | sudo mount -t cifs //DataEtu/<username>/<path> /home/pi/<target_dir>/
2 | -o vers=2.0,username=<username>,rw,uid=$(id -u),gid=$(id -g)
```

La commande `mount` sans argument permet de consulter la liste des montages actifs, par ordre chronologique de création (le plus récent devrait donc se trouver à la fin de la liste).

Les montages réalisés en ligne de commande ne sont pas conservés après un redémarrage.

### Sauvegarde sur le cloud ENSMA (recommandé)

La sauvegarde de votre travail sur le cloud ENSMA vous permet de récupérer vos fichiers de projet depuis n'importe quel poste connecté à Internet (y compris depuis chez vous). Pour en bénéficier, il suffit que votre dossier de travail soit un sous-dossier de l'emplacement `Z:\Documents`. En effet, ce dossier est directement accessible en se connectant au cloud via l'adresse <https://cloud.ensma.fr>.

Le stockage sur le cloud de l'ENSMA vous permet de créer facilement des partages avec d'autres utilisateurs (par exemple les membres de votre groupe) avec des possibilités de paramétrage du type de partage (lecture, écriture, création, suppression de fichiers) ou de sa durée.

#### Collisions

Comme l'utilisateur des Raspberry Pis est unique, pensez à vous coordonner avec les autres personnes y accédant pour ne pas avoir le même répertoire cible de montage. Sinon, le montage le plus récent écrase et désactive temporairement les montages précédemment réalisés sur la même cible.

#### Protection des données

Le service informatique de l'ENSMA s'assure de la sauvegarde régulière (toutes les heures) des données qui sont déposées sur votre disque personnel et dans le cloud. En cas de perte, il est donc possible de demander la restauration d'une version précédente d'un fichier ou d'un dossier.

## 3.1 Format des trames

La figure 3.1 précise le format des messages échangés entre l'application de détection des séquences de notes et les autres acteurs du système (arbitre, simulateur, PiCar). Les messages transmis à plusieurs destinataires devront être strictement identiques.

Ces messages seront encapsulés dans des flux TCP, gérés par des *sockets*.

Les différents champs de ces messages sont listés dans cette section. Des indications de types de données sont fournis entre parenthèses. Les conventions de communication réseau imposent de transmettre les entiers avec une représentation *big-endian*.

**group id** (uint8) identifiant numérique du groupe de TP, c'est-à-dire un entier entre 1 et 8.

**half-group id** (char) identifiant alphabétique du demi-groupe de TP, en majuscule et codé en ASCII, c'est-à-dire le caractère 'A' (code ASCII 65) ou 'B' (code ASCII 66).

**reserved** (binary flags) tableau de booléen réservé à un usage futur. Les sept bits doivent être mis à zéro.

**E (encryption)** (binary flag) booléen indiquant si le reste du message est chiffré (1) ou transmis en clair (0). Le chiffrement est optionnel et est décrit dans la section 3.3.

**magic number** (uint8) constante numérique permettant d'identifier le protocole, ici 230 (0xE6).

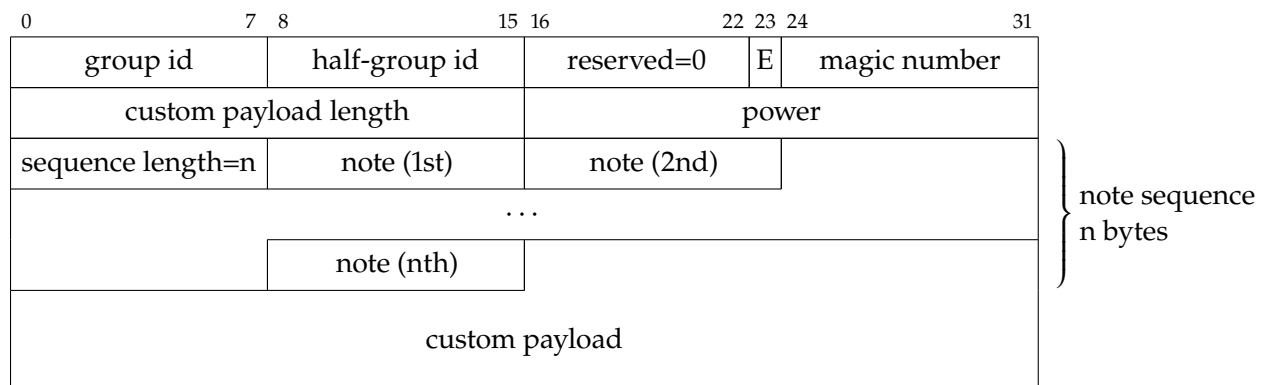


FIGURE 3.1 – Format des trames

**custom payload length** (uint16) longueur, en octets, du champ **custom payload**. Doit être mis à zéro si le **custom payload** est absent.

**power** (int16) instruction de puissance à envoyer au PiCar. La valeur maximale 32 767 (0x7FFF) correspond à une marche avant à vitesse maximale. La valeur 0 (0x0000) à l'arrêt, et la valeur minimale -32 768 (0xFFFF) à la marche arrière à vitesse maximale. Les autres valeurs possibles servent à désigner des régimes de vitesse intermédiaires avec une correspondance linéaire.

**sequence length** (uint8) nombre de notes de la séquence ayant permis de définir la puissance du champ **power**. Cette valeur indique le nombre de répétitions du champ **note**.

**note** (uint8) identifiant numérique d'une note. La génération de cet identifiant est détaillée dans la section 3.2.

**custom payload** champ libre dont la taille est précisée par le champ **custom payload length**. Son contenu est ignoré par l'arbitre mais peut servir à échanger des informations complémentaires avec votre simulateur ou votre PiCar.

La longueur totale d'une trame est donc de *custom payload length* + *sequence length* + 9 octets d'en-tête.

0. (0x0) do
1. (0x1) do♯ (ou réb)
2. (0x2) ré
3. (0x3) ré♯ (ou mib)
4. (0x4) mi
5. (0x5) fa
6. (0x6) fa♯ (ou solb)
7. (0x7) sol
8. (0x8) sol♯ (ou lab)
9. (0x9) la
10. (0xA) la♯ (ou sib)
11. (0xB) si

FIGURE 3.2 – Valeurs numériques associées aux demi-tons

## 3.2 Identifiant des notes

Une note est caractérisé par une hauteur, qui peut être elle-même décomposée en octave et en demi-ton sur cette octave.

L'octave de référence, contenant le *la* de fréquence 440 Hz, possède différentes notation selon la convention utilisée. Par exemple, il s'agit de l'octave numéro 3 dans la notation latine (en usage en France), et de l'octave numéro 4 dans la notation grégorienne. Pour éviter les valeurs négatives, la convention utilisée dans ce projet sera de considérer l'octave 0 comme l'octave incluant la note MIDI la plus grave, le *do*<sub>2</sub> (ou C-1 en notation grégorienne), de fréquence 8.18 Hz. Avec cette convention, le *la* de référence sera donc inclus dans l'octave numéro 5. La table 3.1 donne les associations entre notes et fréquences avec ce système de notation.

Au sein d'une octave, chaque note sera numérotée de 0 à 11, par demi-ton de fréquence croissante. Ces demi-tons sont listés sur la figure 3.2.

L'identifiant d'une note sera alors la concaténation de l'octave et du demi-ton, tous deux représentés avec des entiers non signés sur quatre bits.

Par exemple le *la* de référence aura l'identifiant 89 (0x59), et le *ré♯* de la deuxième octave aura l'identifiant 35 (0x23). La valeur 28 (0x1C) n'est pas un identifiant valide.

TABLE 3.1 – Fréquences (en Hz) des notes de musique dans le référentiel du projet

	octave										
	0	1	2	3	4	5	6	7	8	...	15
do	8.2	16.4	32.7	65.4	130.8	261.6	523.3	1047	2093	...	267 905
do#	8.7	17.3	34.6	69.3	138.6	277.2	554.4	1109	2218	...	283 835
ré	9.2	18.4	36.7	73.4	146.8	293.7	587.3	1175	2349	...	300 713
ré#	9.7	19.4	38.9	77.8	155.6	311.1	622.3	1245	2489	...	318 594
mi	10.3	20.6	41.2	82.4	164.8	329.6	659.3	1319	2637	...	337 539
fa	10.9	21.8	43.7	87.3	174.6	349.2	698.5	1397	2794	...	357 610
fa#	11.6	23.1	46.2	92.5	185.0	370.0	740.0	1480	2960	...	378 874
sol	12.2	24.5	49.0	98.0	196.0	392.0	784.0	1568	3136	...	401 403
sol#	13.0	26.0	51.9	103.8	207.7	415.3	830.6	1661	3322	...	425 272
la	13.8	27.5	55.0	110.0	220.0	<b>440.0</b>	880.0	1760	3520	...	450 560
la#	14.6	29.1	58.3	116.5	233.1	466.2	932.3	1865	3729	...	477 352
si	15.4	30.9	61.7	123.5	246.9	493.9	987.8	1976	3951	...	505 737

### 3.3 Chiffrement des transmissions

De manière optionnelle, votre solution pourra intégrer une solution de chiffrement avec clé secrète (cryptographie symétrique) afin d'éviter que des adversaires puissent analyser votre comportement ou se faire passer pour vous. Pour que l'arbitre puisse tout de même déchiffrer les messages, la stratégie de chiffrement est imposée, et est décrite succinctement dans cette section.

L'algorithme de chiffrement choisi est AES 128-bit en mode *Cipher Block Chaining* (CBC). Les trois premiers octets de la trame restent transmis en clair, le chiffrement ne commence qu'à partir de l'octet 3, c'est-à-dire du *magic number*. La clé de chiffrement et le vecteur d'initialisation des chaînes (*initialisation vector*, ou *iv*), qui sont tous deux à fournir en entrée des algorithmes de chiffrement et de déchiffrement et à conserver secrètement, vous seront communiqués par votre encadrant de séance.

Si la longueur de la portion de message original à chiffrer n'est pas un multiple de 16 bits, il faudra prévoir un octet de *padding*, c'est-à-dire compléter votre message avec la valeur 0x00 pour obtenir une longueur compatible avec le chiffrement par blocs.

#### Vecteur d'initialisation

Le mode CBC consiste à chaîner le chiffrement des blocs par des opérations xor (ou exclusif) afin d'éviter de pouvoir identifier, après chiffrement, les blocs de contenus identiques. Le vecteur d'initialisation sert pour le premier bloc. Normalement il s'agit d'une valeur aléatoire à transmettre avec le message mais, pour faire simple ici, cette valeur sera fixée à l'avance.

La bibliothèque Python `sounddevice` est une solution envisageable pour récupérer les enregistrements sonores sans passer par des fichiers intermédiaires. Un exemple (minimaliste) d'utilisation de cette bibliothèque est donné dans le listing 4.1.

```
1 import sounddevice as sd
2
3 with sd.InputStream(channels=1,
4                     blocksizes=2048,
5                     samplerate=44100,
6                     dtype='int16') as stream:
7     while True:
8         data, overflow = stream.read(2048)
9         # do something...
```

**Listing 4.1:** Exemple de lecture du flux sonore du microphone par défaut

La configuration ALSA (*Advanced Linux Sound Architecture*) de la Raspberry Pi fournie définit le périphérique réel (le microphone USB) comme périphérique d'enregistrement par défaut. Pour tester vos algorithmes de détection sans casser les oreilles des autres participants, huit microphones virtuels *loop1\_out*, ..., *loop8\_out* (associés aux sorties audio *loop1\_in*, ..., *loop8\_in*) permettent de boucler des sorties audio sur des entrées.

Il suffit alors de jouer un morceau sur une des entrées des boucles, par exemple ici sur l'entrée *loop1\_in* pour un fichier nommé *scale.wav*.

```
1 while true; do aplay -f S16_LE -D loop1_in scale.wav; done
```

Le microphone virtuel (*device* pour la bibliothèque `sounddevice`) *loop1\_out* aura alors comme flux sonore le contenu du morceau joué en boucle.

L'implémentation du simulateur est laissée libre. Si vous souhaitez concevoir une application avec un rendu graphique vous pourrez vous référer à la section 2.1 pour voir comment réaliser un déport d'affichage au dessus d'une connexion SSH.

Pour une visualisation en mode console, vous pourrez utiliser la bibliothèque ANSI\_Console manipulée lors des TP de BCL.

Optionnellement, votre simulateur pourra également servir de station sol, c'est-à-dire d'outil de communication au PiCar de consignes manuelles complémentaires comme, par exemple, en forcer l'arrêt.

Votre simulateur devra au minimum prendre en compte les messages échangés entre vos propres composants, mais vous pourrez également accepter les messages des autres groupes si vous souhaitez simuler une course entre plusieurs rovers.