



# Robocars 2

Lucas MCKENNA, Directeur Europe - Point One Navigation

# Agenda

- 1/ But du TP
- 2/ Device setup
- 3/ Recevoir les messages
- 4/ Protocole
- 5/ Calibration
- 6/ Idées d'applications

# 1/ But du TP

---

## Configuration du hardware

Flasher, mettre à jour et différences des images flash.

---

## Récupérer les données

Quels sont les données que l'on récupère? Comment configurer le RTK?

---

## Quels sont les données?

Comment interpréter les données? Comment les visualiser?

---

## Applications

Idées de projets.



## 2/ Device setup - Repo

---

### Répos github à cloner

Voici les répos à cloner:

- [Fusion Engine Client](#) -> Protocoles de communication/outils de dev
- [P1 Host Tools](#) -> Interfacer avec le hardware

---

### Configurer Fusion Engine client

Créer un environnement virtuel et installer les dépendances:

```
cd fusion-engine-client
python -m venv venv
source venv/bin/activate
pip install -r python/requirements.txt
```

---

### Configurer P1 Host Tools

Créer un environnement virtuel et installer les dépendances:

```
cd p1-host-tools
python -m venv venv
source venv/bin/activate
pip install -r requirements.txt
```

## 2/ Device setup - Flash

---

### Type d'images AM vs AP

Il y a deux types d'images:

- [SDK-AM](#) -> GNSS uniquement (commençons par ça)
- [SDK-AP](#) -> GNSS + IMU (nous verrons plus tard, cela requiert une calibration)

Lorsque le hardware est branché, il apparait sous /dev/ttyUSB0 et /dev/ttyUSB1.

---

### Flasher le device

Utilisez l'image SDK-AM (GNSS) pour flasher le hardware Point One:

```
pl-host-tools/firmware_tools/lg69t  
python firmware_tool.py ~/Downloads/quectel-lg69t-am.0.18.3.plfw
```

Ces commandes configurent le hardware pour qu'il tourne en mode GNSS.

Vu que le processeur est un MCU, il est monotâche, donc à partir du moment où le device est flashé, le hardware est continuellement en train de naviguer. C'est la seule chose qu'il sait faire.

---

### 3/ Recevoir les messages - interface avec le HW

---

#### Config\_tool

Testez votre configuration avec:

```
python pl-host-tools/bin/config_tool.py
```

S'il le script tourne, vous êtes bon.

Le script config\_tool.py permet de configurer les messages out, leurs frequences (**en ms**) et la calibration pour le IMU.

---

#### Récupérer les données

Pour récupérer les données de position:

```
python3 pl-host-tools/bin/runner.py --device-id LE_NOM_DE_VOTRE_DEVICE  
--polaris VOTRE_ID_POLARIS_VOIR_AVEC_COLIN
```

Le device ID est pour identifier votre device dans les logs (/logs/device\_id/log\_guid), votre ID polaris est très importante car elle permet de récupérer les corrections RTK (en format RTCM) et d'avoir le GPS centimétrique.

À chaque fois que vous faites tourner le script, un nouveau log avec des données de positionnement sera créé.

**Note: soyez sur que l'antenne est en extérieur avec une vue sur le ciel, sinon vous ne verrez pas les satellites.**

---

## 3/ Recevoir les messages - logs

---

### Les logs

Comme dit précédemment, chaque instance du script p1\_runner crée un log.

Les logs sont créés par défaut ici: **\$HOME/logs/**.

L'extension de nos logs sont: .p1log ou .raw, exemple:

~/logs/2024-12-07/my\_device/bb553782fd4e431387dcd96091681a64/output/diagnostic.playback.p1log < fichier contenant uniquement du protocole Fusion Engine

~/logs/2024-12-07/my\_device/bb553782fd4e431387dcd96091681a64/output/input.raw < fichier contenant plusieurs protocoles

La composition est la suivante:

- Root des logs: ~/Documents/logs/
- Date: 2024-12-07
- Identifiant du device: my\_device
- Log Guid (identifiant unique du log): bb5537...
- Répertoire "output", qui contient les fichiers binaires
- Puis, le fichier qui a les informations de positionnement: diagnostic.playback.p1log/input.raw

### Outils de lecture des logs

Vous pouvez changer de repo pour aller dans **fusion-engine-client**.

Il y a plusieurs applications qui nous permettent de lire les logs et de les déchiffrer (ils sont open source, vous pouvez donc vous en inspirer pour faire vos propres applications en temps réel).

Exemple (p1\_display):

```
python fusion-engine-client/python/fusion_engine_client/applications/p1_display.py  
~/logs/2024-12-07/my_device/bb553782fd4e431387dcd96091681a64/output/diagnostic.playback.p1log
```

Cela générera une page HTML avec toutes les métadonnées nécessaires à l'analyse du log.

### 3/ Recevoir les messages - interface

The screenshot shows a web interface with several sections. Arrows point from the following labels to their corresponding sections in the interface:

- Pages web.** points to the top navigation menu.
- Firmware.** points to the 'Type' and 'Version' table.
- Informations de durée.** points to the 'Last Crash Type Crash Count' and 'Description' / 'Time' table.
- Qualité de la position.** points to the 'Position Type Count Percent' table.
- Nombres de messages.** points to the 'Message Type Count' table.

**Navigation Menu:**

- [Calibration Status](#)
- [GNSS Azimuth & Elevation Vs. Time](#)
- [GNSS CNO vs. Time](#)
- [GNSS Corrections Status](#)
- [GNSS Dilution of Precision \(DOP\) vs. Time](#)
- [GNSS Signal Status](#)
- [GNSS Sky Plot](#)
- [Pose Displacement From Median Position: Top-Down \(Topocentric\)](#)
- [Pose Displacement From Median Position: vs. Time](#)
- [Solution Type](#)
- [Stationary Status](#)
- [Time Scale](#)
- [Vehicle Pose vs. Time](#)
- [Vehicle Trajectory \(Map\)](#)

**Type Version**

Firmware	1.9
FusionEngine	
OS	
GNSS Receiver	

**Last Crash Type Crash Count**

CRASH\_TYPE\_NONE 0

**Description Time**

Log Start Time	P1: 84362.000 sec System: POSIX: 2262-04-11 23:47:16.854776+00:00 (9223372036.855 sec) GPS: 2258:418539.200 (1366056939.200 sec) UTC: 2023-04-20 20:15:21.200
Total Log Duration	2278.0
Processed Start Time	P1: 84362.643 sec System: POSIX: 2262-04-11 23:47:16.854776+00:00 (9223372036.855 sec) GPS: 2258:418539.843 (1366056939.843 sec) UTC: 2023-04-20 20:15:21.843
Processed Duration	2278.0 seconds

**Position Type Count Percent**

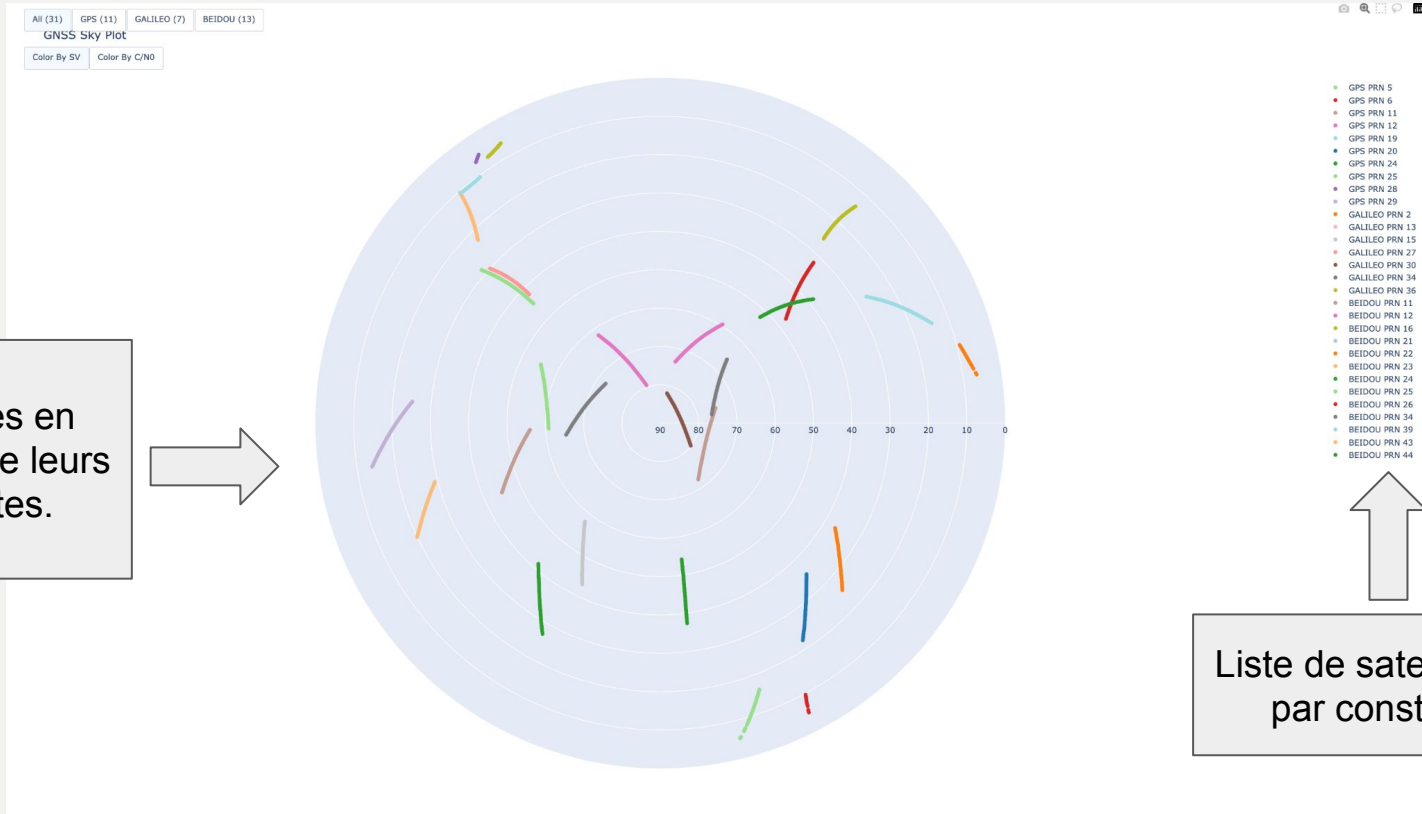
Invalid	0	0.0%
Integrated	2050	9.0%
Standalone	68	0.3%
DGPS	93	0.4%
RTK Float	1079	4.7%
RTK Fixed	19467	85.6%
PPP	0	0.0%
Vision	0	0.0%
Total	22777	

**Message Type Count**

POSE (10000)	22777
GNSS_INFO (10001)	22777
GNSS_SATELLITE (10002)	22777
POSE_AUX (10003)	22777
CALIBRATION_STATUS (10004)	228
IMU_OUTPUT (11000)	226362
WHEEL_SPEED_OUTPUT (11135)	179560
VERSION_INFO (13003)	76
DEVICE_ID (13007)	76
Total	497410

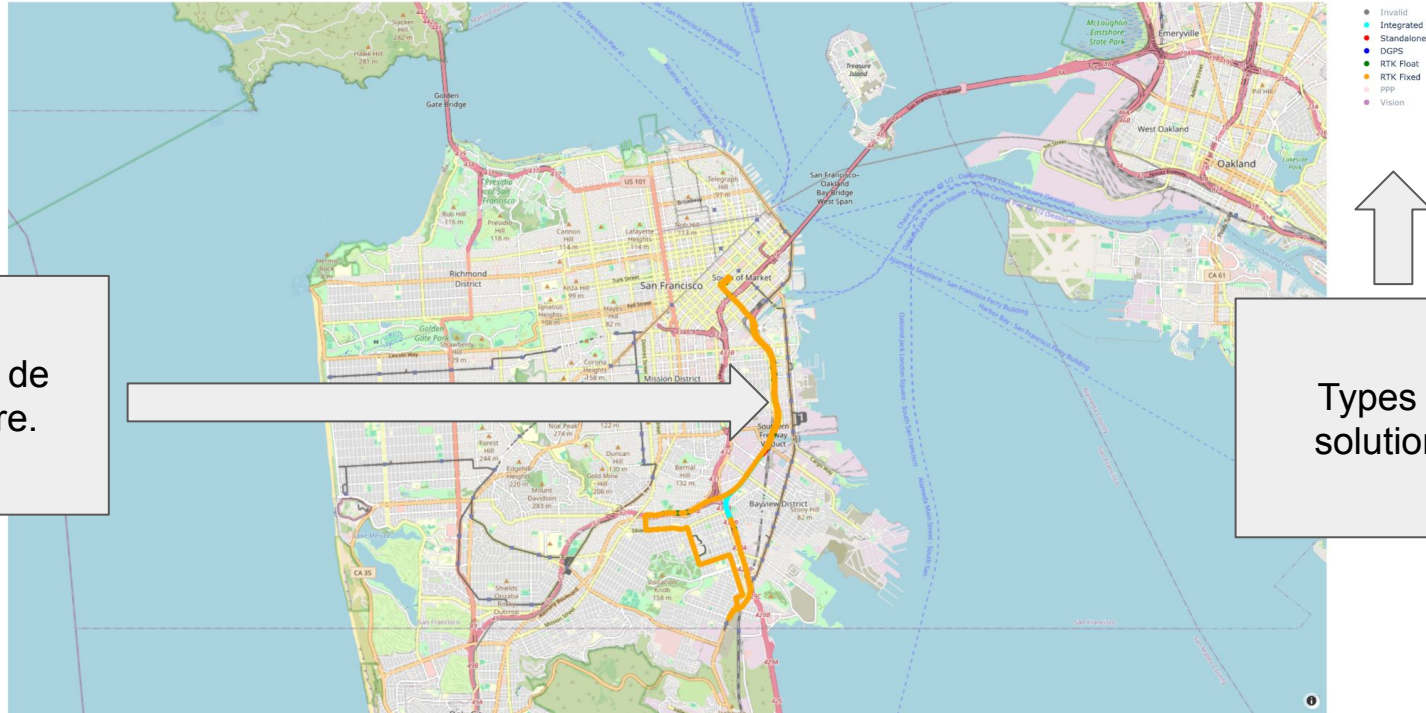


### 3/ Recevoir les messages - Sky plot



### 3/ Recevoir les messages - Map

Vehicle Trajectory  
For satellite imagery, please provide a Mapbox token using --mapbox-token or by setting MAPBOX\_ACCESS\_TOKEN.



Exemple de  
trajectoire.

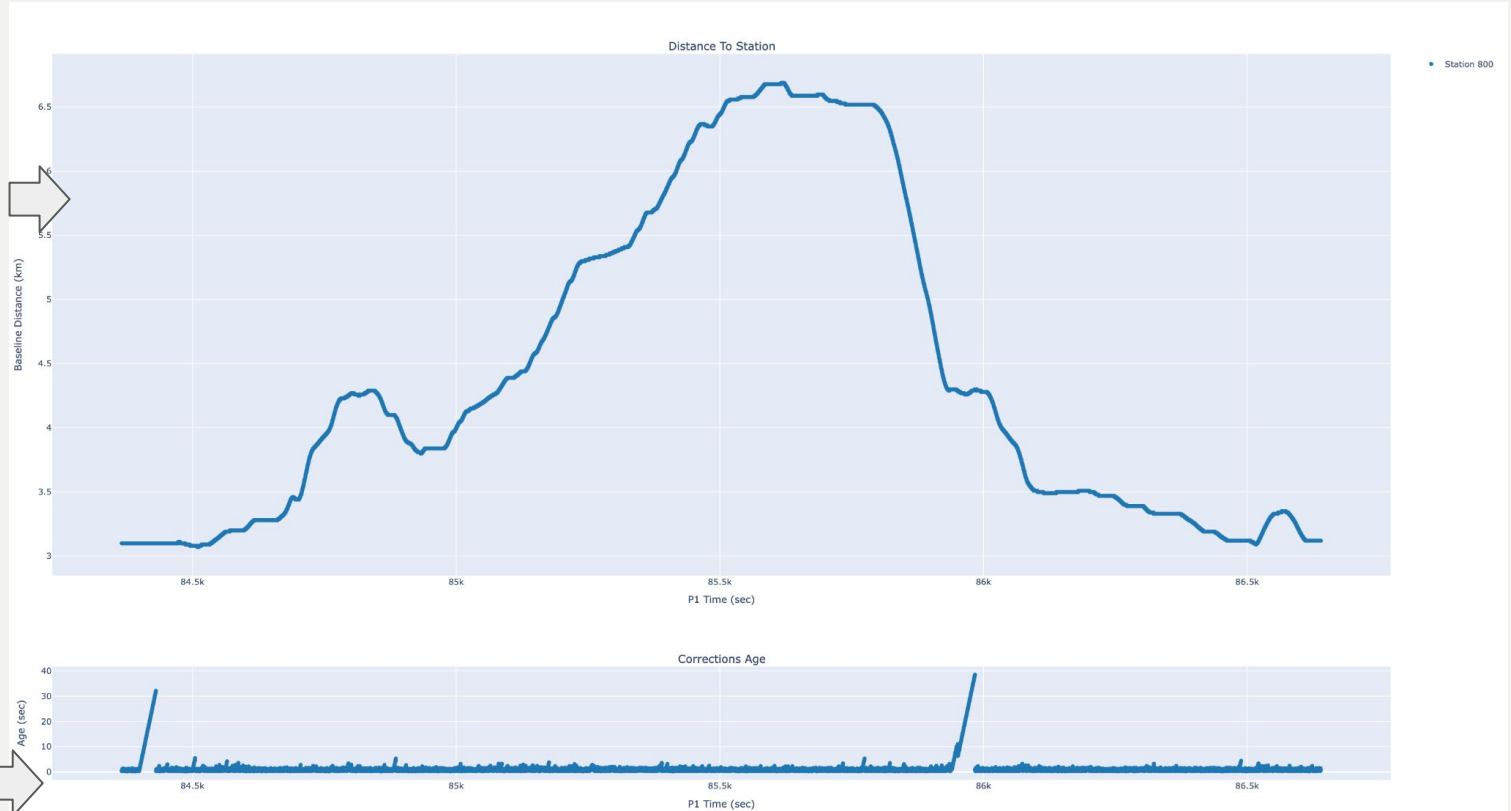
Types de  
solutions.

### 3/ Recevoir les messages - Base station correction age/distance

Distance de la  
balise la plus  
proche.

**Très important  
pour savoir si  
votre système  
utilise le RTK!**

Temps en  
seconde de la  
dernière  
correction  
utilisé.



### 3/ Recevoir les messages - Les solutions de navigation

Position Type	Count	Percent
---------------	-------	---------

Invalid	0	0.0%
Integrated	2050	9.0%
Standalone	68	0.3%
DGPS	93	0.4%
RTK Float	1079	4.7%
RTK Fixed	19487	85.6%
PPP	0	0.0%
Vision	0	0.0%

Total	22777
-------	-------

Solution invalide.
--------------------

Solution intégré sans GNSS (IMU en sous-sol)
----------------------------------------------

GNSS sans corrections.
------------------------

Solution GPS différentiel moins qualitative que RTK. (ancêtre du RTK),
------------------------------------------------------------------------

GNSS avec correction et précision 50cm ou plus.
-------------------------------------------------

Solution à 2-3cm.
-------------------

## 4/ Le protocol - Fusion Engine protocol

---

### Fusion Engine protocole

Vous pouvez voir [ici tous les messages du fusion engine protocol](#).

Le message le plus important pour vous, la position est le [PoseMessage](#).

Ce message contient:

- message\_type -> type de message, en fonction du message c'est PoseMessage, IMU, CMD etc...
- solution\_type -> type de solution vue dans le slide précédent.
- p1\_time -> le temps monotonic créé par Point One.
- gps\_time -> le temps GPS des horloges atomique des satellites.
- lla\_deg -> latitude, longitude, altitude en degré (permet d'afficher les coordonnées sur google maps par exemple).
- position\_std\_en\_u\_m -> [standard deviation](#) en mètre ([enu = East North Up, un système de référence local](#)).
- ypr\_deg -> ypr = yaw pitch roll, qui est [l'attitude](#) du véhicule (uniquement disponible lorsque l'IMU est disponible).
- ypr\_std\_deg -> [standard deviation](#) on degré.
- velocity\_body\_mps -> vitesse en mètre par secondes.
- velocity\_std\_body\_mps -> [standard deviation](#) de la vitesse.

Ce sont les attributs donc vous aurez besoin pour afficher les données sur une carte et comprendre l'orientation du véhicule.

### Autres messages

Vous pouvez voir [ici tous les messages du fusion engine protocol](#).

- [Solution Type](#) -> vue dans le slide precedent.
- [Message Type](#) -> tous les messages disponibles dans le protocol.
- [IMUOutput](#) -> inertie calibré.
- [RawIMUOutput](#) -> inertie non calibré.
- Messages [ROS](#) -> si vous voulez utiliser l'environnement de simulation robotique [ROS](#).

## 5/ La calibration - Utiliser le IMU

### Configurer l'IMU

Pour utiliser l'IMU il faut maintenant changer le repo: p1-host-tools. Pour utiliser le script: [config\\_tool.py](#).

**Note: Pour faire confiance à une solution GNSS + IMU, il faut avoir fait une route de calibration, c'est à dire une conduite de 15-20 min pour "exciter" les capteurs. Un message est défini pour ça: [CALIBRATION STATUS](#), qui donne un pourcentage de complétion de la calibration.**

Maintenant, configurons l'IMU:

```
%(command) = python p1-host-tools/bin/config_tools.py
```

Reset the device calibration and state:

```
%(command)s reset calibration
```

Read the active lever arm values.

```
%(command)s read
```

Apply changes to the GNSS and device (IMU) lever arms, then save the changes to persistent storage.

```
%(command)s apply gnss 0 0.4 1 -> antenna to body
```

```
%(command)s apply device 0 2.1 1 -> device to body
```

```
%(command)s save
```

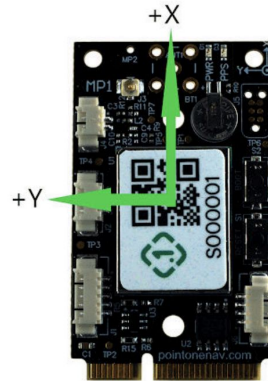
### L'orientation du device

Pour avoir l'orientation correct: mettez le device en +X et +Y comme l'image à côté.

Voir la documentation [ici](#).

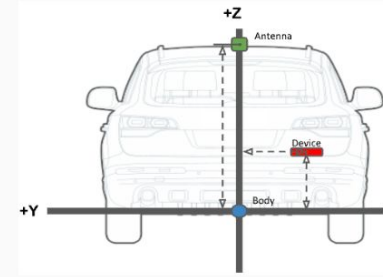
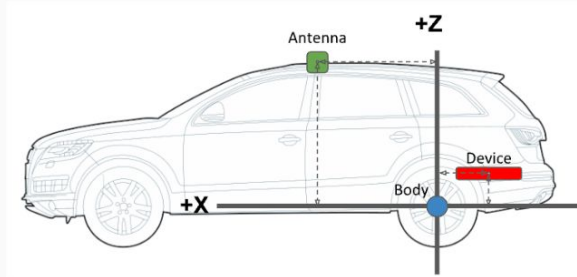
**Note: Vous devez obligatoirement flasher avec le firmware [AP](#) pour utiliser l'IMU!**

#### 6.3.1 Device (IMU) Orientation



# 5/ La calibration - Utiliser le IMU

## Orientation



### Primary GNSS Antenna Lever Arm

Specify the lever arm (in meters) of the GNSS antenna with respect to the vehicle body.

X*	Y*	Z*
<input type="text" value="0.00"/>	<input type="text" value="0.00"/>	<input type="text" value="0.00"/>

### Device Lever Arm

Specify the lever arm (in meters) of the Atlas device with respect to the vehicle body.

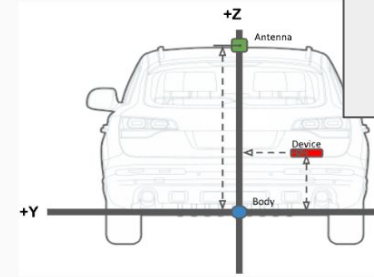
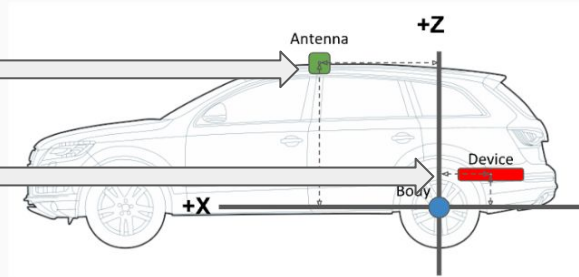
X*	Y*	Z*
<input type="text" value="0.00"/>	<input type="text" value="0.00"/>	<input type="text" value="0.00"/>

## 5/ La calibration - Utiliser le IMU

### Orientation

Il faudra mesurer en XYZ en m, la distance entre l'antenne et le centre de l'axe arrière.

Il faudra mesurer en XYZ en m, la distance entre le device et le centre de l'axe arrière.



**Assurez vous que l'orientation du device est bien X+ Y+ et Z+. En gros que l'avant face l'avant du véhicule.**

### Primary GNSS Antenna Lever Arm

Specify the lever arm (in meters) of the GNSS antenna with respect to the vehicle body.

X*	Y*	Z*
<input type="text" value="0.00"/>	<input type="text" value="0.00"/>	<input type="text" value="0.00"/>

### Device Lever Arm

Specify the lever arm (in meters) of the Atlas device with respect to the vehicle body.

X*	Y*	Z*
<input type="text" value="0.00"/>	<input type="text" value="0.00"/>	<input type="text" value="0.00"/>



## 6/ Idées d'applications

- Collecte de données initial (GNSS only).
- Collecte de données RTK fixed (prouver que vous pouvez fixer).
- Création d'outils de visualisation temps réels.
- Prouver que vous avez une calibration IMU qui fonctionne.
- Faire un suivi de trajectoire dans un espace sans structure en extérieur.
- Utiliser temps GPS pour prendre des décisions sur plusieurs robots en même temps (éviter les collisions).

# Questions?

## Resources:

- [Doc HW](#)
- [Ressources Point One](#)
- [Fusion Engine Client](#) (Specs)
- [Firmware Tools](#)
- [P1 Host Tools](#)
- [Fusion Engine Client](#) (Repo)
- [GraphQL](#)

## Contact:

Lucas MCKENNA, Directeur Europe

Email: [lucas@pointonenav.com](mailto:lucas@pointonenav.com)  
Tel: +33 7 69 59 33 34