Les Annexes

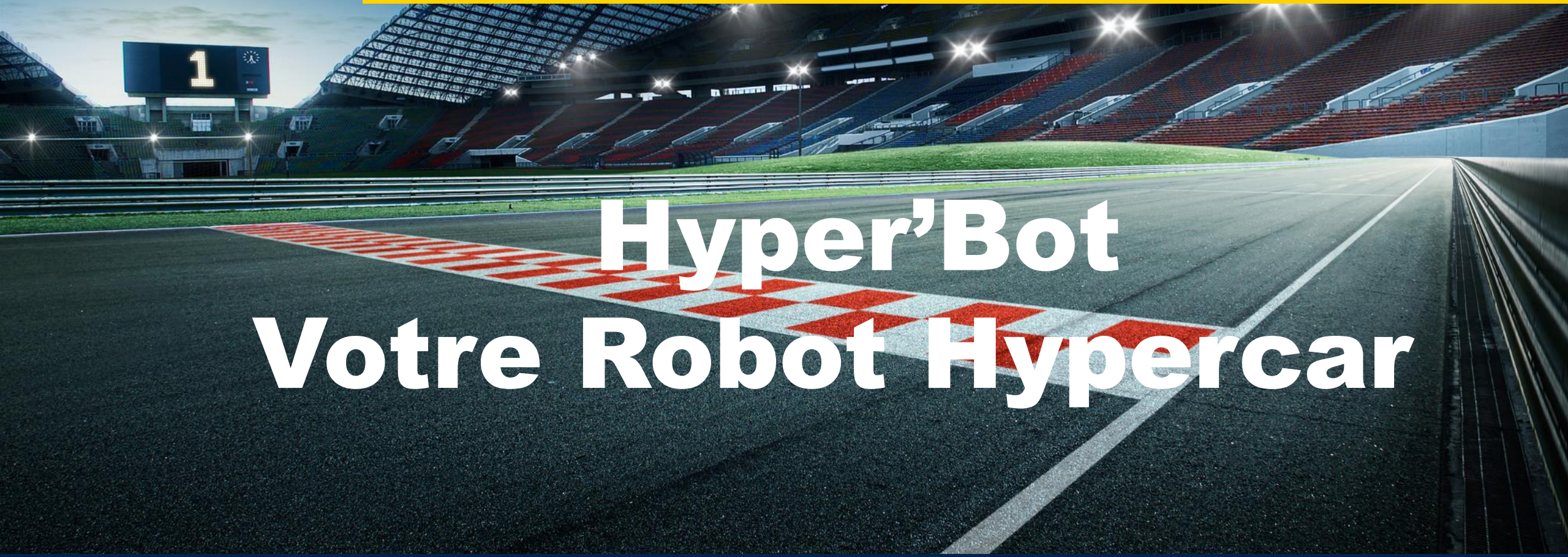
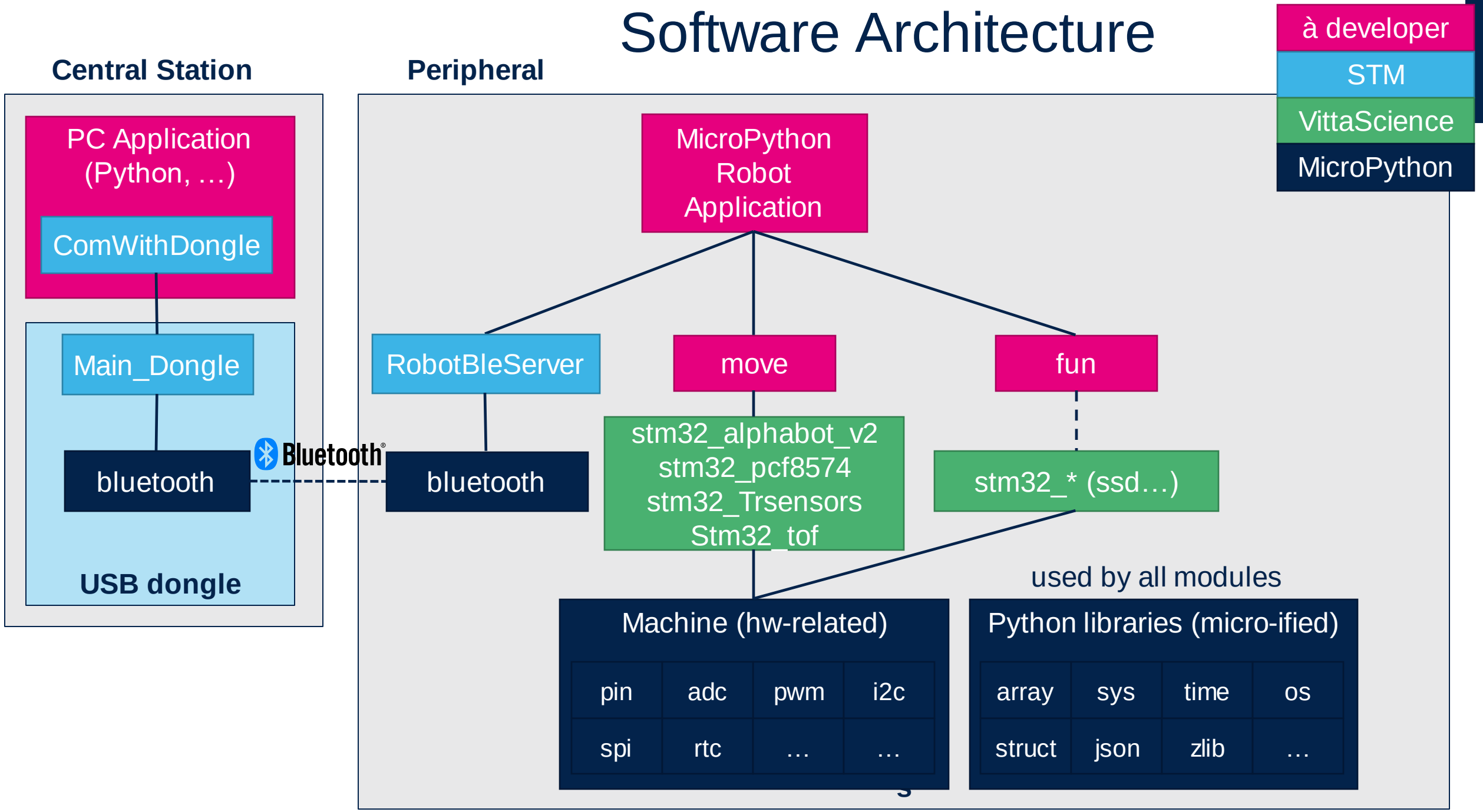


Table des matières

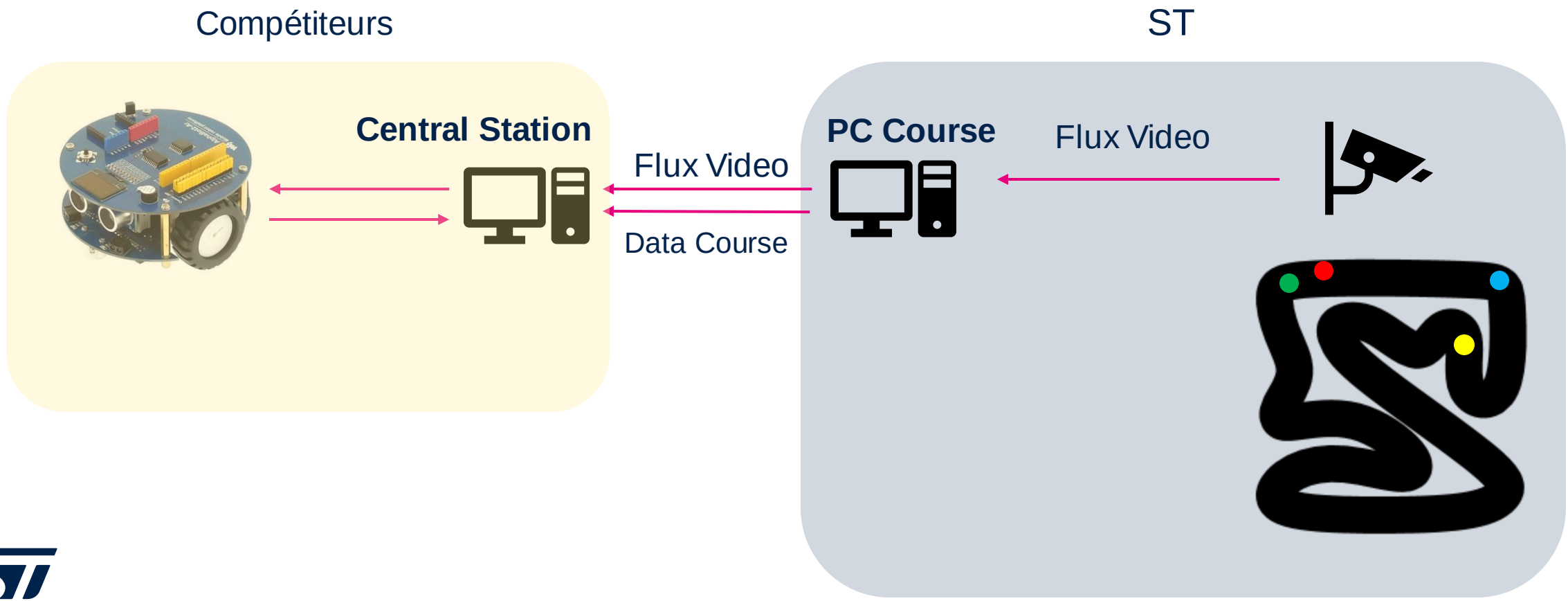
life.augmented

Informations générales

Software Architecture



System Architecture - Flux de données



life.augmented

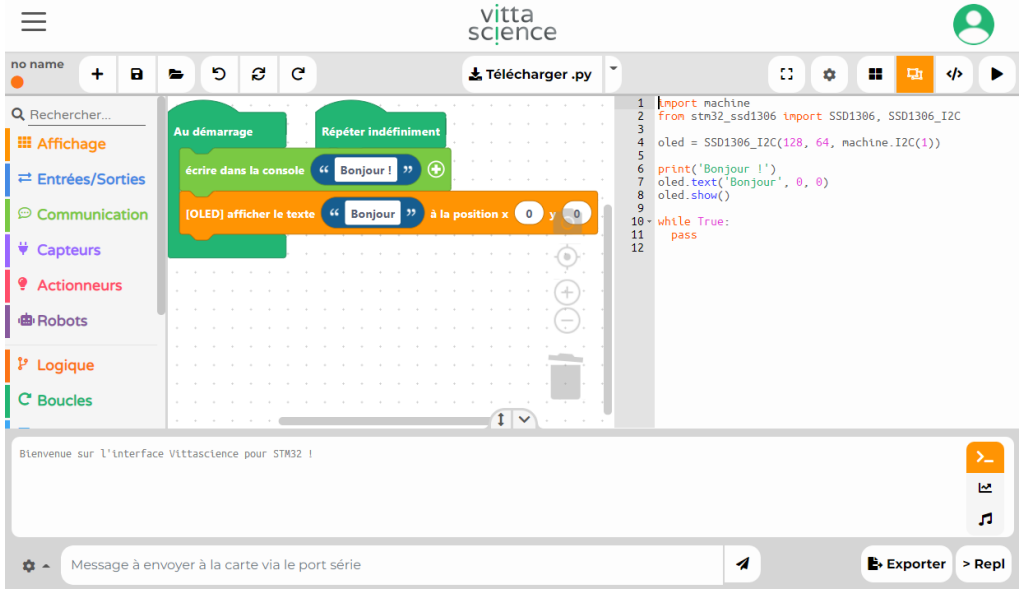
MicroPython useful libraries

- MicroPython official web site
- Documentation
- All available library list
  - **Python standard libraries**
    - sys, os, struct… **but only a subset of standard libraries are available!**
  - **MicroPython-specific libraries**
    - mainly linked to embedded hw
    - Machine: reset, i2c, spi, adc, pwm, gpio...
    - Bluetooth Low Energy (BLE): only BLE mode is supported on STM32WB55
  - **MicroPython port-specific libraries**
    - Stm32 mcu

VittaScience

VittaScience useful libraries (including stm32 ones)

- VittaScience official web site
- VittaScience stm32 online simulator
- VittaScience stm32 github
  - Libraries for the Time of Flight (ToF), lcd, sensors…
  - and everything for the stm32 AlphaBot v2!



life.augmented

STM

STMicroelectronics libraries & code examples

- **MainPcTestBLE.py:** a PC Python script used to demonstrate the communication between the PC and the robot thanks to the USB dongle.
- **ComWithDongle.py (for PC), mainDongle.py (for dongle), RobotBleServer.py (for robot):** BLE communication and messages management library.
- **MainRobot.py**: a python script (to run on robot) to demonstrate the communication between the PC and the robot thanks to the USB dongle
- **USB dongle:** in charge of the BLE communication (initiation, attachment, send & receive messages…) between the PC and the robot.

BLE communication and messages management library should not be modified by the participants.

life.augmented

à developer

Ce que vous avez à développer

- **Application en MicroPython sur votre robot:** c'est votre code source en MicroPython qui va vous permettre de piloter les différentes parties du robot ainsi que de communiquer avec votre application PC.
- **Application sur votre PC (Station Centrale):** c'est votre code source qui va vous permettre de récupérer les données de course (et le flux vidéo) de suivre les ordres de la Direction de Course et de communiquer avec votre robot afin de le piloter.
- **Move:** c'est à vous de le développer ! En s'appuyant sur les caractéristiques du robot.
- **Fun**: à vous d'être créatif :-)
- Explorez le code source qui vous est fournit sur le gitlab : https://gitlab.com/st24hducode/2023

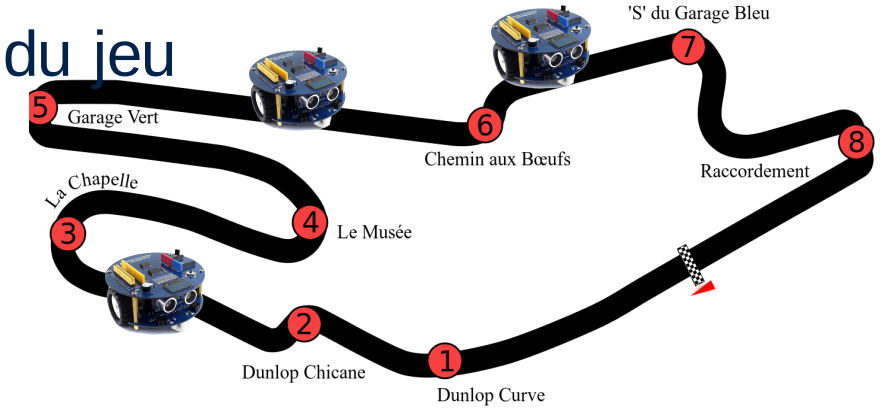
Quelques réponses à vos questions…

- **Le robot est utilisé par mes collègues, comment faire pour travailler sur cette épreuve en parallèle?** Plusieurs solutions:
  - Dans votre kit, il y a une carte supplémentaire Nucleo STM32WB55 et vous pouvez donc travailler dessus en MicroPython.
  - Vous pouvez aussi développer votre code en Python sur votre PC et vous ferez le portage plus tard. Attention de ne pas utiliser de dépendance à des bibliothèques Python non disponibles en MicroPython.

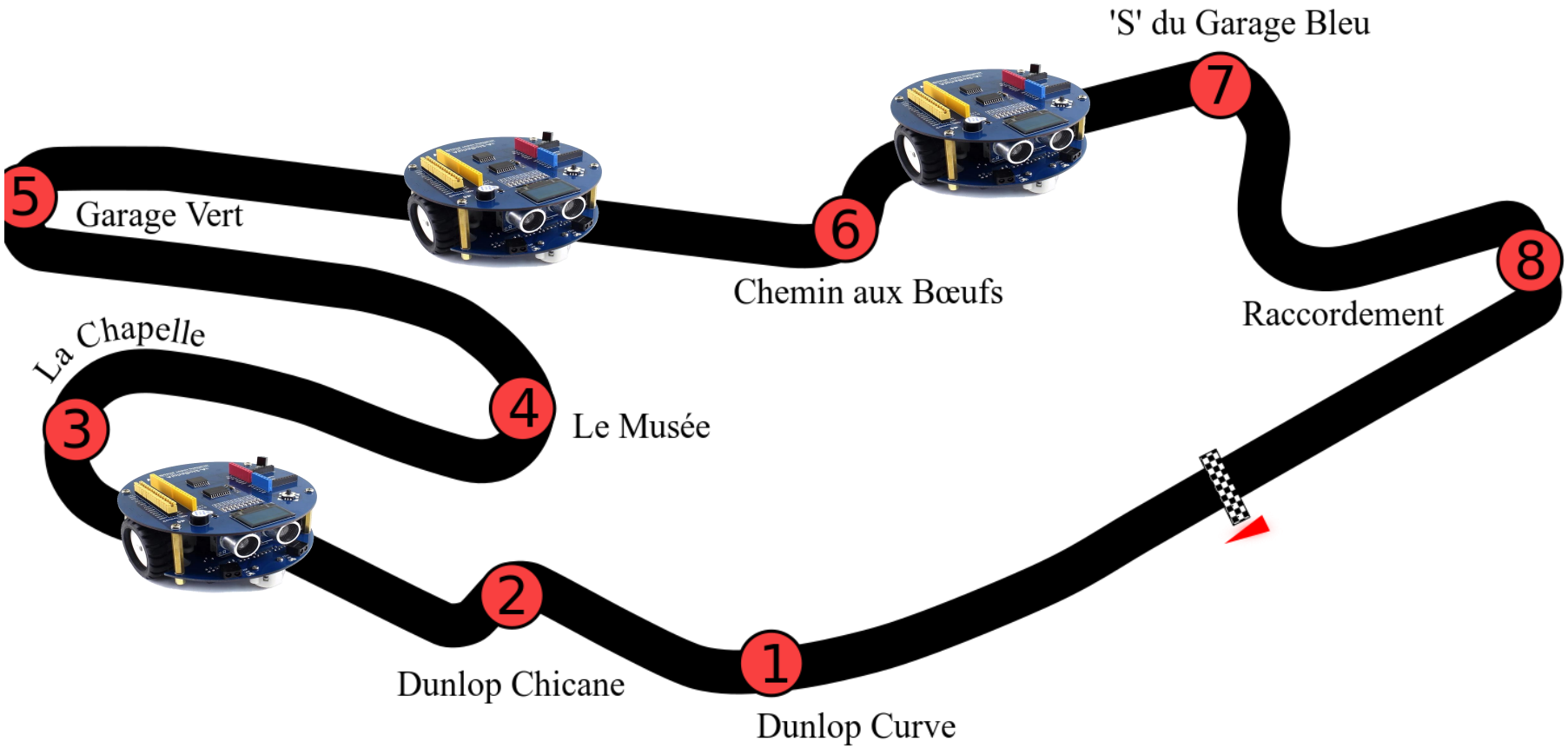
Environnement de la course

Course d'Hyper'Bots

- Course d'Hyper'Bots (AlphaBots) type « FIA WEC » sur une piste imprimée proche du circuit Bugatti des 24h du Mans.
- Comme une « vraie » compétition: Après une séance d'essais/qualifications, les Hyper'Bots participeront à l'épreuve finale lors d'une course à durée déterminée (~24 * 100 dixièmes de secondes).
- Caméra au dessus de la piste pour récupérer puis communiquer des informations aux participants (position sur le circuit, classement …)
- Des pénalités seront appliquées selon le règlement du jeu
- Mode « autonome » (bonus).



life.augmented


'S' du Garage Bleu
7
5
Garage Vert
6
Chemin aux Bœufs
Raccordement
8
La Chapelle
3
4
Le Musée
2
Dunlop Chicane
1
Dunlop Curve

Règlement 1/6

- Les Hyper’Bots sont sur la base d’un AlphaBot. Ils seront customisés pour donner l’apparence d’une voiture.
- La piste sera suffisamment large pour permettre des dépassements.
- La Direction de Course (ST) peut donner des directives et indications:
  - Start
  - Stop
  - Full Course Yellow
  - Classement des robots à chaque tour

life.augmented

Règlement 2/6

- En plus des capteurs présents sur l'alphaBot, une camera placée à la verticale du circuit permettra d'envoyer un flux vidéo qui sera:
  - Traité par la Direction de Course pour déterminer la position (x, y) de chaque robot, cette position sera ensuite envoyée à l'ensemble des robots
  - Reçu par les « Central Station » qui devront en déterminer leur position (x, y)

- Le classement des Robots sera affiché par la Direction de Course et communiqué aux robots.

Reglement 3/6

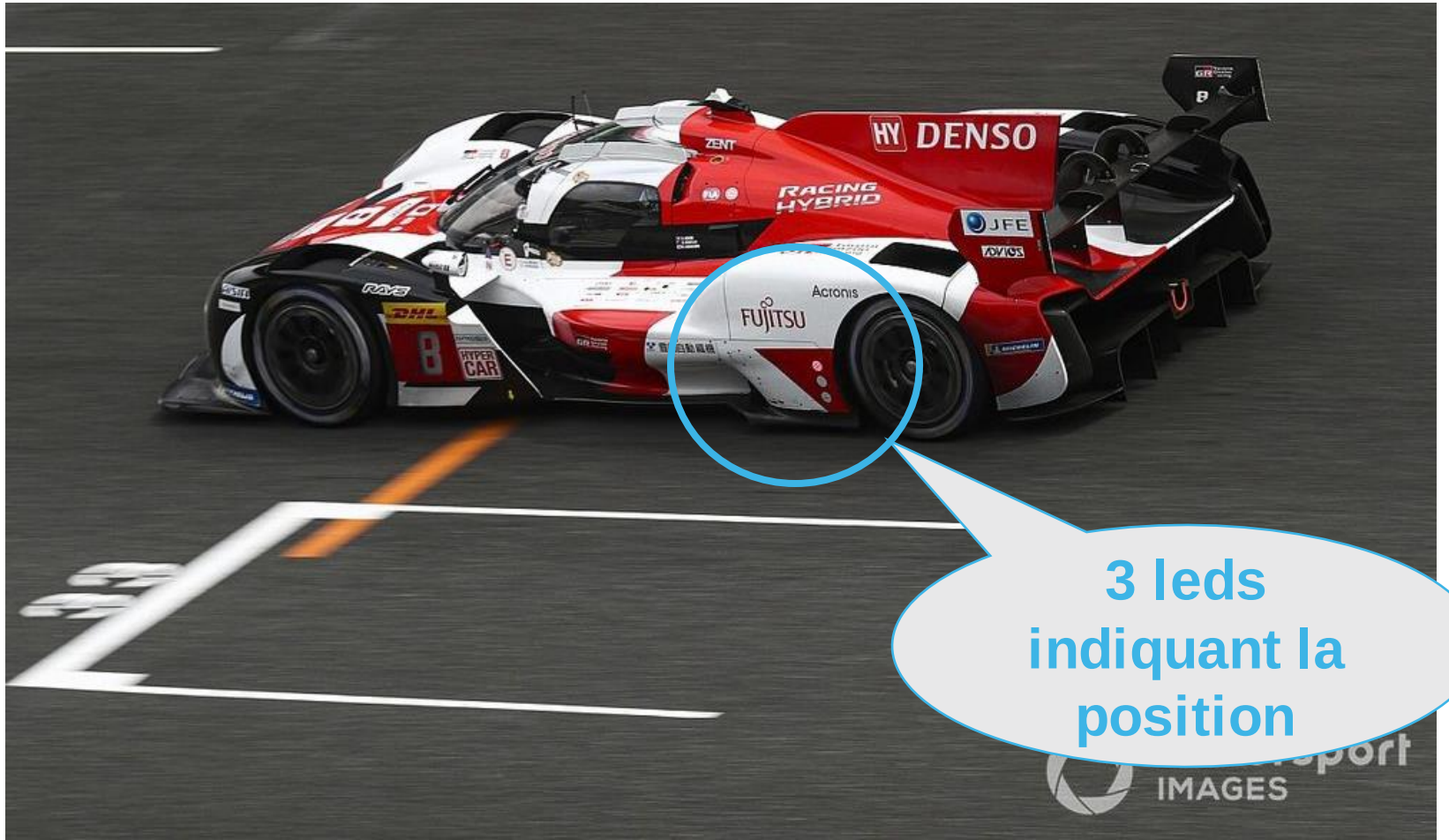
- Le classement des robots sera visible via le système de lumières positionné sur les cotés de la carrosserie.

  1er: une lumiere

  2eme: deux lumieres

  3eme: trois lumieres

  couleur : rouge



life.augmented

Règlement 4/6

- La “Vérification Technique” permettra de valider des points techniques comme :
  - Bon fonctionnement des lumières
  - Prise en compte des directions
- La Direction de Course vérifiera que l’environnement de course est pris en compte
  - Slow (Vitesse « Full Course Yellow »)
  - Start
  - Stop

Règlement 5/6

- Nombre de robots par course en fonction du nombre de participants et des essais/qualification (max 4 robots par course).
- Taille du circuit : 4m x 2m

Règlement 6/6

- Mode télécommande autorisé uniquement pour l’épreuve 2

life.augmented

Epreuve 1 – Démarrer votre équipement

life.augmented

Installation du matériel et de son logiciel pour le piloter

- Le **robot** est livré « monté ». Vous pouvez le connecter à votre PC grâce au **câble USB fourni**.
- Les **3 cartes STM32WB55 de votre kit** (robot, dongle USB, carte supplémentaire) contiennent:
  - Un firmware pour l'interpréteur MicroPython: celui-ci est déjà présent, vous n'avez pas besoin d'y toucher.
  - Un firmware pour la stack BLE: celui-ci est déjà présent, vous n'avez pas besoin d'y toucher.
  - Un système de fichiers pour accueillir les scripts MicroPython: **c'est ici votre zone de jeu :-)**
    - Note: le **dongle USB** contient déjà les fichiers MicroPython pour discuter avec le PC et avec le robot en Bluetooth BLE ➔ Vous n'avez donc pas besoin de modifier ce code contenu dans le dongle USB.
- En cas de problème, **n'hésitez pas à contacter l'équipe ST** qui pourra remettre à jour le contenu du dongle USB et/ou effacer de manière propre le contenu du système de fichiers de vos cartes…
- Du code source MicroPython d'exemples ainsi que les bibliothèques pour gérer **l'AlphaBotv2** (le robot) et ses capteurs (**fichiers stm32_*.py**) se trouvent sur la clé USB ou sur le git.
- Pour les amateurs d'électronique: plus d'informations, dont les schémas sur le site WaveShare.

L'environnement logiciel de développement

- **Système**: Windows, Linux ou macOS, à vous de choisir :-)
- **Editeur Python**: Votre éditeur de code favoris (Visual Studio Code, GVim, Emacs, NotePad++, Sublime Text…). Pour l'éducation, nous utilisons souvent l'éditeur très simple Thonny…
- **Terminal de communication**: votre terminal de communication préféré (minicom, hyperterminal, PuTTy, TeraTerm, pyterm, gtkterm…) avec la configuration **115200-8-N-1**: Windows (COMx) , Linux (/dev/ttyACMx).
- **Tips**: Dans la console,
  - **CTRL+D** pour redémarrer la carte
  - **CTRL+C** pour arrêter le programme MicroPython en cours.
- N'hésitez pas à consulter le site https://stm32python.gitlab.io/fr/ qui contient quelques informations complémentaires…

Quelques réponses à vos questions…

- **Comment mettre mon code MicroPython sur le robot ou sur la carte supplémentaire?**
  - Suivez la procédure décrite à partir de la **page 28** du Guide d'utilisation - Robot martien version STM32 vous n'avez besoin que du mode « **5V VIN** » pour le robot et « **USB_MCU** » pour la carte supplémentaire ou plus simplement, **ne touchez pas aux cavaliers de configuration car ils sont déjà à la bonne place ;-)**
  - Dans votre explorateur de fichiers, un nouveau lecteur au nom de « **PYBFLASH** » apparait, copiez vos fichiers MicroPython dedans, **main.py** sera le 1er fichier de votre application. Quelques commandes d'exemples:
    - Linux: `$ cp main.py /media/$USER/PYBFLASH; sync`
    - macOS: `$ cp main.py /Volumes/PYBFLASH; sync`
  - **IMPORTANT pour éviter le risque de corruption du système de fichiers!**
    - Lors de la mise à jour des fichiers, une LED rouge (LED3) s'allume; attendez bien qu'elle s'éteigne avant de faire le reset ou de retirer le câble USB…
    - Si vous avez besoin de retirer le câble USB (notamment pour laisser le robot autonome), n'oubliez pas avant de retirer le câble USB de "démonter" le lecteur « PYBFLASH ».
- **Comment recharger les piles du robot?** Suivez les informations **page 29** du Guide d'utilisation - Robot martien version STM32

life.augmented

Epreuve 2 – Verification technique

life.augmented

Quelques réponses à vos questions…

- **Pourquoi avons-nous besoin d'un dongle USB pour le Bluetooth BLE alors que mon PC possède nativement du Bluetooth BLE?** Car cela vous évite d'installer une "stack Bluetooth de développement" sur votre PC sachant que celle-ci serait différente sur Windows, Linux et macOS :-)
- **Quel est le contenu du dongle USB ?** Il contient la librairie aioble et le fichier mainDongle.py (renommé en main.py), ces fichiers se trouvent dans le répertoire "ble". Vous n'avez pas à toucher au contenu du dongle USB :-)
- **Où se trouve le code d'exemple pour communiquer avec le robot?** Ce sont les fichiers ComWithDongle.py et mainPcTestBLE.py qui se trouvent dans le répertoire « ble », lisez bien le fichier ble/README.md pour plus d'informations… et explorer le code :-)

Quelques réponses à vos questions…

- **Qu'est-ce que la librairie "aioble"?** C'est une librairie fournie avec micropython (on peut la trouver ici https://github.com/micropython/micropython-lib/tree/master/micropython/bluetooth/aioble) qui permet de gérer le BLE à plus haut niveau que la librairie "bluetooth". Cette librairie est déjà installée sur le dongle et le robot, vous n'avez pas besoin d'y toucher.

Quelques réponses à vos questions…

- **Comment identifier mon robot et ma carte supplémentaire?** Utiliser le nom de votre équipe. Ex: Pilote (pour le robot) & Pilote+ (pour la carte supplémentaire). Voir dans le fichier "ble/README.md" pour plus de détails sur comment renseigner le nom de votre équipe dans le code
- **Comment utiliser les bandeaux LEDs (sur le coté du robot) ?**
  - Bandeau leds NeoPixel (3 par bandeau)
  - Pins : D0 & D1
  - Vittascience est votre ami…

Epreuve 3 – Environnement de course
Epreuve 8 – Classement du robot

life.augmented

Environnement de Course

- **Race data streaming**
  - Officielle (course) : 224.1.1.1:5007
  - Le code source d'exemple utilise par défaut cette adresse. Si celle-ci venait à changer pendant la course, la Team ST vous communiquera la nouvelle adresse à mettre à jour via une simple variable d'environnement:

```
# Example for the env variable:
# For Linux:
# $> export MCAST="224.1.1.1:5007"
# For Windows:
# $> set MCAST=224.1.1.1:5007
# IMPORTANT For Windows user only:
# In UDP multicast, Windows requires an extra information: the network interface IP of your own laptop
# Set this network interface IP with an env variable (you can get it with ipconfig), for example:
# $> set NETWORK_INTERFACE=10.201.21.74
```

Validation des épreuves nécessitant une simulation : 224.1.1.2:5007

Environnement de Course

- Ajouter port 5007 dans firewall Windows
  - How To Open Firewall Ports In Windows 10 https://www.tomshardware.com/news/how-to-open-firewall-ports-in-windows-10,36451.html
  - How to Open UDP Port in Windows 10/11 Firewall https://www.youtube.com/watch?v=t0raIizX1aU

Environnement de Course

- **Video streaming**
- Le server ST diffuse un flux video en udp multicast / rtsp & webrtc aux adresses suivantes:

```
IP_ADDRESS=192.168.0.122
PORT=8554
MAPPING=mystream
rtsp://${IP_ADDRESS}:${PORT}/${MAPPING}
web rtc http://${IP_ADDRESS}:9997
```

Environnement de Course

- **Video streaming**
- Pour récupérer un flux video en rtsp:

Exemple avec GStreamer pour afficher le flux en rtsp udp multicast:

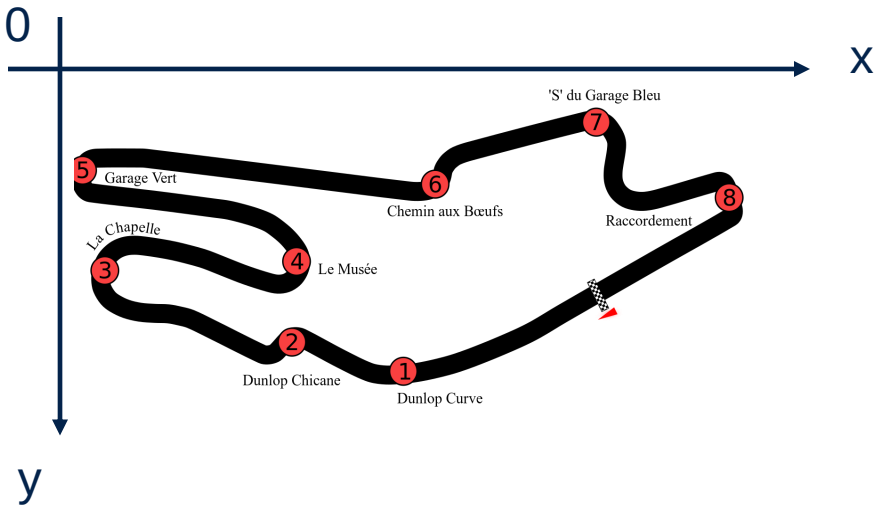
```
IP_ADDRESS=192.168.0.122
PORT=8554
MAPPING=mystream
export MYSINK="fpsdisplaysink text-overlay=false sync=false video-sink=\"xvimagesink\" -v"
export MYRTSP="rtspsrc location=rtsp://$IP_ADDRESS:$PORT/$MAPPING"

gst-launch-1.0 $MYRTSP latency=10 protocols=udp-mcast ! queue ! rtph264depay ! h264parse ! avdec_h264
! videoconvert ! video/x-raw,width=1280,height=720 ! ${MYSINK} # udp multicast, low latency
```

- Il est également possible de mettre ce flux vidéo dans un programme openCV

Quelques réponses à vos questions…

- **1 trame Race Data** sera envoyé continuellement contenant les informations de course.
  - racedata : gestion des données de course pour convertir les bytes stream réseau en données utiles de courses
- Une image de référence comme ci-dessous vous sera fournie :



life.augmented

Epreuve finale

life.augmented

Quelques réponses à vos questions…

- **Que doit-on faire à l'épreuve finale?**

Les objectifs sont décrits dans la présentation « Epreuves ». De plus, n'hésitez pas à vous reporter à la présentation faite le samedi matin.

Bonne chance à tous :-)

life.augmented