# MQTT JSON

---

## ROUTE

### Transmetteur -> Receveur

Description

```
{

}
```

# /map

## OpenCV -> GameEngine, Joueur

Informations de la map (longueur, largeur), informations obstacles  (angle orientation, longueur, largeur):

A RENVOYER A CHAQUE X SECONDES ?

```
{
  "mapWidth": int,
  "mapHeight": int,
  "checkpoints: [
    {
      "id": int,
      "x": int,
      "y": int,
    },
    …
  ],
  "obstacles": [
    {
      "id": int
      "angle": float,
      "x": int
      "y": int
    },
    …
  ]
}
```

# /player/register

## Joueur -> GameEngine

Informations de base du joueur qui se connecte à la partie (salon, attente des joueurs)

```
{
  "uuid": str, // utiliser QUuid de Qt
  "pseudo":str,
  "controller": str, // ia, keyboard, controller, vr, phone // pour le graphique
  "vehicle": str,
  "team": int // null automatiquement si partie libre
}
```

# /player/control

## Joueur -> GameEngine

```
{
  "uuid": str,
  "angle": float, // [-90°;90°]
  "power": int, // [-100%;100%]
  "buttons": { // état des boutons
    "banana": bool,
    "bomb": bool,
    "rocket": bool
  }
}
```

# /game

## GameEngine -> Joueur

Informations de la partie en temps réel (coordonnées des joueurs, leur véhicule, leur vitesse… )
Informations des items  placés sur la map en temps réel (coordonnées).

```
{
  "players": [
    {
      "uuid": str,
      "pseudo": str,
      "color": str,
      "team": int
      "x": int,
      "y": int,
      "angle": float,
      "speed": int,
      "vehicle": str, // type de vehicle : bike, car, truck
      "items": { // nombre d'objet que le joueur possède
        "banana": int,
        "bomb": int,
        "rocket": int
      },
      "lastCheckpoint": int, // dernier point de passage passé
      "currentLap": int,
      "status": str, // driving, accident
      "controller": str // ia, keyboard, controller, vr, phone // pour
    },                      de l'affichage seulement
    …
  ],
  "items": [
    {
      "x": int,
      "y": int,
      "angle": float,
      "status": str // banana : placed; bomb : flying, waiting,
exploding; rocket : flying, exploding
      "type": str // banana, bomb, rocket
    },
    …
  ],
  "elapsedTime": int,
  "infoMessage": str, // ex : "la partie commence dans 10 secondes"
  "status": str // waiting, progress, ended, paused
}
```

# /game/properties

## GameEngine -> Joueur

Initialisation de la partie (propriété de la partie)
Taille de la carte
Nombre d'items

A RENVOYER A CHAQUE X SECONDES ?

```
{
  "lapsNb": int, // nombre de tour à faire
  "teamNb": int, // nombre d'équipe dans la partie

  "circleRadius": float,
  "rectangleWidth": float,
  "rectangleHeight": float,
  "checkpointRadius": float,

  "bananaNb": int, // -1 = infinie
  "bombNb": int, // -1 = infinie
  "rocketNb": int, // -1 = infinie
  "bananaCd": int, // temps entre deux utilisations
  "bombCd": int, // temps entre deux utilisations
  "rocketCd": int, // temps entre deux utilisations
  "rocketSpeed": float, // vitesse d'une roquette
  "bananaTtl": int, // temps avant que la banane disparaisse
  "bombTtl": int, // temps avant explosion

  "vehicleOptions" {
    "<vehicle>": { // vehicle : bike, car, truck
      "maxSpeed": int,
      "acceleration": float,
      "weight": int,
      "steeringAngle": float, // angle de braquage
    }
  }
}
```

Angle : en radian
UUID : V4 obligatoirement généré par une fonction sur le client - RFC4122