Требования к протоколу:

1. Отображение состояния соединения (запрос на соединение – ответ готовность).

2. Управление перемещением камеры (вверх-вниз-влево-вправо)

3. Задание скорости перемещения камеры (можно 1 раз + необязательное поле при управлении)

4. Управление приближением-отдалением

5. Выдача текущего положения

6. Задание положения “по умолчанию”

7. Команда на возврат к положению по умолчанию

3. Протокол должен содержать несколько разных структур (по функционалу)

4. Управление поворотом и приближением-отдалением должно работать одновременно (чтобы можно было скомандовать камере одновременно повернуться и приблизить или отдалить)

5. Команда завершения работы (чтобы камера случайно не вращалась без остановки) – перевод в положение по умолчанию

6. Передача пути для RTSP потока по запросу.

7. \*запрос и выбор доступных камер (это пока опционально предусмотрим)

Сообщения на входе и на выходе удобнее всего будет читать из формата JSON

Используемые структуры

Данные камеры для подключения

CameraConf: {

"ip": string,

"port": int

}

Координаты положения камеры (включая фокусировку)

Position: {

"x": float,

"y": float,

"zoom": float

}

Запросы

Каждый запрос обязан содержать имя: поле "type"  
Если запрос предполагает какую то информацию кроме своего имени, она должна храниться в поле "block". Такие условия позволяют избежать лишних проверок при получении/отправлении запросов.

1. Проверка соединения

input: {

"type": "ConnectionRequest"

}

output: {

"type": "ConnectionResponse"

}

2. Установить значения скоростей для движения по умолчанию

{

"type": "SetMoveVelocity",

"block": {

"x\_velocity": float,

"y\_velocity": float,

"zoom": float

}

}

3. Относительное движение на определенный промежуток времени с заданными скоростями

{

"type": "ContiniousMove",

"block": {

"duration": float, (>=0)

"default"velocities": bool (Если устанавливаем True, можем не отправлять скорости, используются скорости по умолчанию)

"x\_velocity": float,

"y\_velocity": float,

"zoom": float

}

}

4. Получить текущую позицию

input: {

"type": "GetPosition"

}

output: {

"type": "Position",

"block": {

"position": Position

}

}

5. Установить домашнюю позицию

{

"type": "SetHomePosition",

"block": {

"position": Position

}

}

6. Движение на позицию установленную как домашняя

{

"type": "MoveToHomePosition"

}

7. Безопасно закрыть соединение (остановливает камеру перед закрытием соединения)

input: {

"type": "CloseConnection"

}

output: {

"type": "ConnectionClosed"

}

8. Получить путь к RTSP потоку

input: {

"type": "GetRTSP"

}

output: {

"type": "RTSP",

"block": {

"path": string

}

}

9. Получить доступные камеры

request: {

"type": "GetAvailableCameras"

}

response: {

"type": "AvailableCameras"

"block" {

"cameras": list[CameraConf]

}

}

10. Остановки движения

{

"type": "StopMoving"

}

Future Release:

11. Абсолютное движение

{

"type": "AbsoluteMove"

"block": {

"position": Position

"velocity": float [Optional]

}

}

12. Получить ограничения камеры по координатам и по скоростям (x, y, zoom)

{

"type": "GetLimits"

"block": {

...

}

}