**Протокол информационного взаимодействия**

**Системы управления камерой и**

**Системой аналитики**

**Версия 1.04**

Обеспечиваемые функции:

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

Используемый порт для двухстороннего обмена – 56000

Каждое переданное сообщение должно представлять из себя корректную json-строку

Любое сообщение, у которого не указана возвращаемая структура, возвращает подтверждение выполнения запроса:

{

"type": string = RequestType + "Response"

}

Например ответ на SetMoveSpeed будет SetMoveSpeedResponse

Используемые структуры

Данные камеры для подключения

CameraName: {

"ip": string,

"port": int | string

}

Координаты положения камеры (включая фокусировку)

Position: {

"x": float,

"y": float,

"zoom": float

}

Координаты скоростей изменения величин вектора скоростей (включая фокусировку). Его длина не может быть нулевой.

Speed: {

"x\_speed": float,

"y\_speed": float,

"zoom\_speed": float

}

Запросы

Каждый запрос обязан содержать имя: поле "type"  
Если запрос предполагает какую то информацию кроме своего имени, она должна храниться в поле "block".

Если запрос направлен на конкретную камеру – то она должна быть заранее подключена к пользователю через ConnectionRequest, а запрос должен содержать внутри "block" ее имя CameraName. Такие условия позволяют избежать лишних проверок при получении/отправлении запросов.

1. Установка соединения с камерой.

input: {

"type": "ConnectionRequest",

"block": {

"camera\_name": CameraName,

"user": string,

"password": string

}

}

output: {

"type": "ConnectionResponse",

"block": {

"camera\_name": CameraName,

"is\_connected": bool

}

}

2. Установить значения скоростей для движения по умолчанию.

{

"type": "SetMoveSpeed",

"block": {

“camera\_name”: CameraName,

"speed": Speed

}

}

3. Относительное движение на определенный промежуток времени с заданными скоростями. Для параметра speed при пропуске будет задано значение переданное в SetMoveVelocity.

{

"type": "ContiniousMove",

"block": {

"camera\_name": CameraName,

"duration": float, (>=0)

"speed”: Speed [Optional]

}

}

4. Получить текущую позицию.

input: {

"type": "GetPosition"

"block": {

"camera\_name": CameraName

}

}

output: {

"type": "Position",

"block": {

"position": Position

}

}

5. Установить текущую позицию домашней.

{

"type": "SetHomePosition",

"block": {

“camera\_name”: CameraName

"position": Position

}

}

6. Движение на позицию установленную как домашняя. Для параметра speed при пропуске будет задано максимально возможное значение.

{

"type": "GotoHomePosition"

“block”: {

“camera\_name”: CameraName,

“speed”: Speed [Optional]

}

}

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

input: {

"type": "CloseConnection",

“block”: {

“camera\_name”: CameraName

}

}

8. Получить путь к RTSP потоку

input: {

"type": "GetRTSP",

“block”: {

“camera\_name”: CameraName

}

}

output: {

"type": "RTSP",

"block": {

"path": string

}

}

9. Получить доступные камеры.

request: {

"type": "GetAvailableCameras"

}

response: {

"type": "AvailableCameras",

"block" {

"cameras": list[CameraName]

}

}

10. Остановки движения.

{

"type": "Stop"

“block”: {

“camera\_name”: CameraName,

“stop\_x\_y”: bool,

“stop\_zoom”: bool

}

}

11. Абсолютное движение. Для параметра speed при пропуске будет задано значение переданное в SetMoveVelocity, при отсутствии там – максимально возможное.

{

"type": "AbsoluteMove",

"block": {

“camera\_name”: CameraName,

"position": Position,

"speed": Speed [Optional]

}

}

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

request: {

"type": "GetLimits",

“block”: {

“camera\_name”: CameraName

}

}

response {

"type": "Limits",

"block": {

“position\_min”: Position,

“position\_max”: Position

}

}

13. Относительное движение. Для параметра speed при пропуске будет задано значение переданное в SetMoveVelocity, при отсутствии там - максимально возможное.

{

"type": "RelativeMove",

"block": {

“camera\_name”: CameraName,

"relative\_position": Position,

"speed": Speed [Optional]

}

}

Возвращаемые ошибки

В случае некорректного запроса, сервер должен указывать клиенту, где его запрос неверен.

В случае, если было получено некорректная json-строка, будет отправлены указания на ошибку от JSONDecoder:

{

"Invalid syntax": string

}

Ответы на ошибки, связанные с содержанием переданного словаря возвращаются в формате словаря, содержащего массив пояснений к каждому ошибочному полю:

{

Field1 name: list[string]

Field2 name: list[string]

...

}