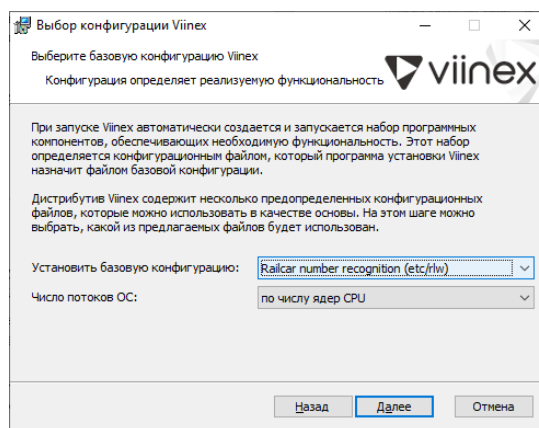


# Распознавание номеров грузовых железнодорожных вагонов в Viinex 2.0

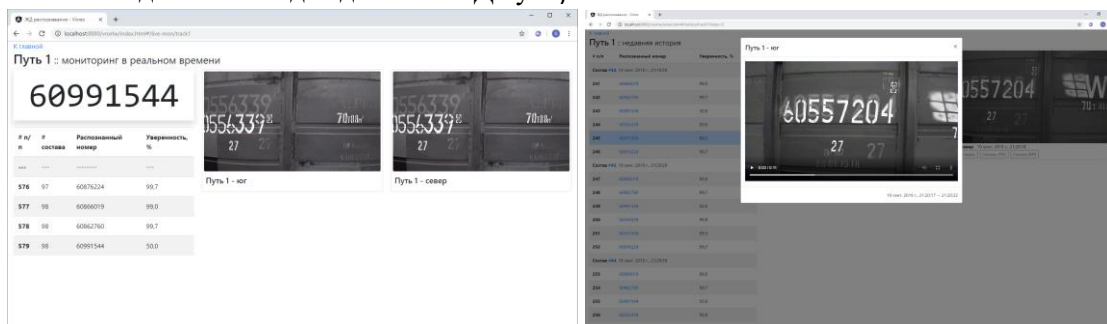
## Первая установка и запуск демонстрационного ролика

Для первой установки и запуска системы с демо-роликом предлагается выполнить следующие шаги:

- Скачать дистрибутив по ссылке <ftp://ftp.viinex.com/viinex-2.0/Viinex-2.0.0.324-ru.msi>
- Установить дистрибутив. При установке выбрать в выпадающем меню выбора начальной конфигурации предпоследний пункт (Railcar number recognition, папка etc/rw). (Это рекомендуется делать только при ознакомительной установке. Для разработки и реальной эксплуатации Viinex 2.0 рекомендуется выбирать опцию «Конфигурация в папке etc/conf.d»).

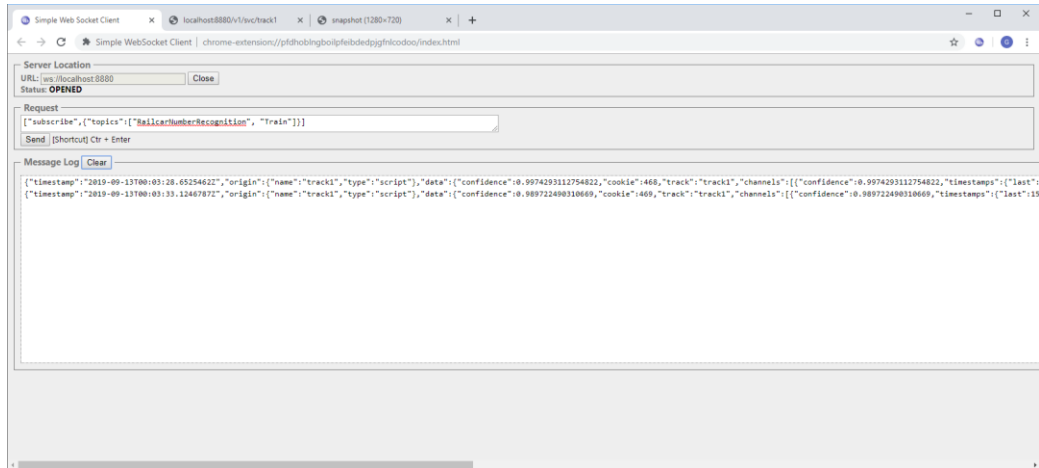


- Этот и следующий шаг нужен только для демонстрации: Скачать демонстрационный ролик по ссылке <ftp://ftp.viinex.com/demo-video/rlpr/rlwuz.mp4>
- Скопировать файл rlwuz.mp4 в папку etc/rw.
- Установить драйвер SenseLock [ftp://ftp.viinex.com/viinex-2.0/prerequisites/senselock\\_windows\\_3.1.0.0.zip](ftp://ftp.viinex.com/viinex-2.0/prerequisites/senselock_windows_3.1.0.0.zip), если он еще не был установлен. Вставить в компьютер комплект USB ключей Viinex 2.0.
- Запустить службу Viinex 2.0.
- В браузере Chrome открыть демонстрационный веб-интерфейс системы распознавания номеров ЖД вагонов, доступный по ссылке <http://localhost:8880/vnrlw/index.html>. В веб-приложении по этой ссылке должны быть доступны все сконфигурированные ЖД пути (после установки по умолчанию это 1 путь), для каждого из которых будет доступен режим мониторинга в реальном времени (отображение живого видео от всех видеоканалов для данного ЖД пути, и получение в реальном времени событий о результатах распознавания номеров вагонов), а также режим просмотра последней истории (просмотр информации о вагонах в последних зарегистрированных составах: время прохождения состава, результаты распознавания вагонов, а также просмотр и экспорт архивного видео и изображений вагонов по всем видеоканалам для данного ЖД пути).

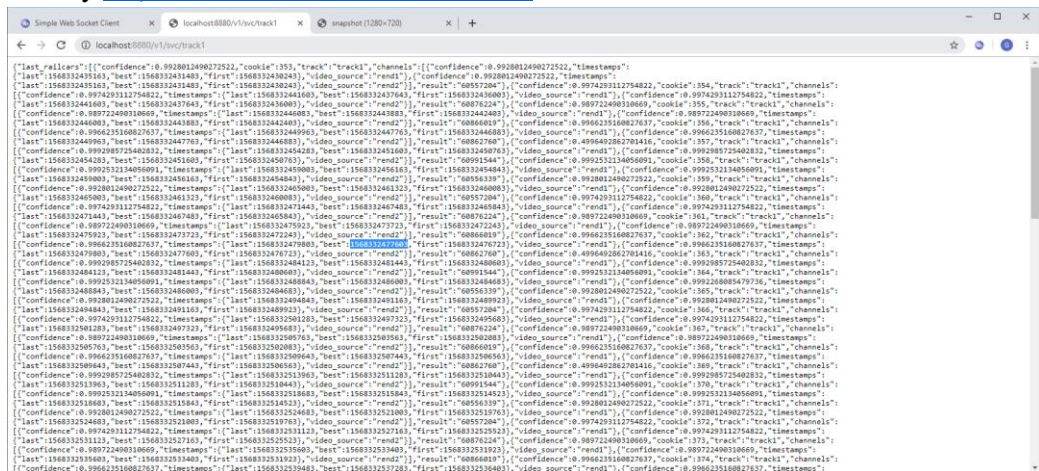


- Открыть webrtc видеопоток от эмулируемой камеры 1 или камеры 2 (<http://localhost:8880/#/webrtc-video/webrtc0/cam1>).

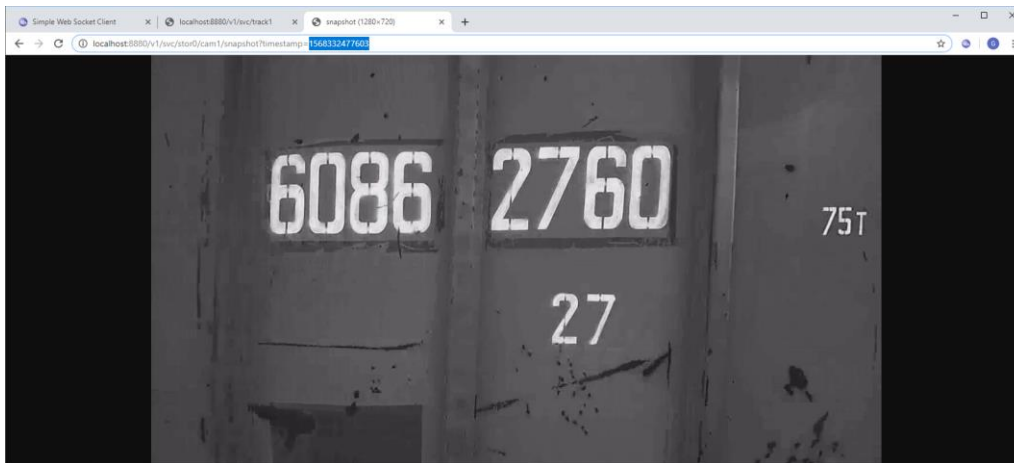
- Этот и следующий шаг нужен только для тестирования функциональности получения событий. Установить в браузере Chrome расширение Simple WebSocket Client <https://chrome.google.com/webstore/detail/simple-websocket-client/pfdhoblngboilpfeibdedpjgfnlcodoo>.
- Запустить расширение Simple WebSocket Client. Подключиться к Viinex (в поле Server Location URL ввести ws://localhost:8880 и нажать Open). Подписаться на события о распознавании номеров ЖД вагонов (в поле Request ввести ["subscribe", {"topics": ["RailcarNumberRecognition", "Train"]}]). Нажать кнопку Send. В нижнем окне "Message log" должны начать приходить события о распознанных номерах.



- Получить последние распознанные номера с помощью HTTP запроса: в браузере открыть ссылку <http://localhost:8880/v1/svc/track1>



- Получить наилучший кадр из видеоархива по временной метке: выбрать в результатах предыдущего запроса значение метки времени timestamps.best для одного из вагонов (выделено на изображении), и используя это значение, открыть ссылку <http://localhost:8880/v1/svc/stor0/cam1/snapshot?timestamp=1568332477603> (в параметре timestamp= должно быть подставлено актуальное значение времени).



## Компоненты и внутреннее устройство системы

Система распознавания номеров грузовых железнодорожных вагонов и цистерн на базе Viinex 2.0 использует оригинальное ядро Viinex без каких-либо модификаций, и стандартные компоненты, описанные в документации к последнему <https://www.viinex.com/Viinex20Guide.pdf>. Реализация специфичной видеоаналитики (собственно распознавание номеров вагонов по видео) и логики (управление процессом распознавания, сбор результатов, управление записью видео) выполнены как стандартные объекты Viinex 2.0 – внешний процесс («process», раздел 2.1.15 документации), и скрипт («script», раздел 2.1.14 документации).

### Видеоаналитика

Видеоаналитика, как внешний процесс, реализована в исполняемом файле v20rlwrec.exe, запускается службой Viinex 2.0 автоматически, и обслуживает все ЖД пути и все видеоканалы распознавания для данного сервера. При этом данный процесс содержит минимальную необходимую логику. Будучи отдельным по отношению к Viinex процессом, данный модуль обменивается с Viinex сообщениями, которые служат для передачи сигнала о необходимости начать или закончить распознавание по тому или иному пути, а также для передачи результатов распознавания в Viinex. Кроме того, данный процесс получает декомпрессированное видео для анализа. Для этого используется механизм «локального транспорта», коротко описанный в гл.5 документации. Источниками декомпрессированного видео являются объекты «renderer», которые осуществляют декодирование видеопотока от видеоканалов и публикацию такого потока в области разделяемой памяти. В свою очередь, процесс v20rlwrec.exe подключается к этой же области разделяемой памяти, и после получения команды для начала распознавания – начинает анализировать видеоданные, попадающие в нее. После получения команды для окончания распознавания, процесс v20rlwrec формирует результат распознавания и передает его в Viinex 2.0, где этот результат направляется в управляющий скрипт, о котором идет речь в следующем подразделе.

### Управление распознаванием и логика работы

Процесс v20rlwrec.exe реализует лишь минимальную необходимую функциональность для распознавания номеров ЖД вагонов, он получает на вход простые команды, выдает результат распознавания, не запоминая его, и практически не имеет внутреннего состояния. Этот процесс следует рассматривать как внутренний компонент системы распознавания номеров ЖД вагонов. Взаимодействие с другими частями системы и более сложную логику реализует управляющий скрипт. Механизм встроенных в Viinex 2.0 скриптов, описанный в главе 4 документации. Особенностью такого способа построения логики системы распознавания номеров вагонов в Viinex 2.0 является то, что компания Viinex предоставляет клиентам достаточную для типовых решений реализацию управляющего скрипта, однако эта реализация является открытой, поставляется в виде исходного кода, и при необходимости доступна для модификации.

Управляющий скрипт создается как объект Viinex типа «script»; исходный код типовой реализации управляющего скрипта устанавливается в папку Program Files (x86)\Viinex\etc\rlw в файл rlwlogic.js.

Экземпляр объекта «script», загружающий исходный код из данного файла, должен быть создан для каждого ЖД пути, для которого требуется распознавание номеров вагонов. В соответствии с принятой в Viinex логикой, соответствующий объект script должен быть связан с другими объектами Viinex, с которыми он должен взаимодействовать, в разделе links. Это всегда будет процесс, v20rlwrec, который осуществляет распознавание, а также, как правило, объект Recording Controller (для управления записью видео в архив), иногда объект типа “modbus” или другой источник событий DigitalInput (для получения сигналов от оптического датчика), а также веб-сервер (для публикации событий с результатами распознавания, а также для обработки управляющих API-запросов). В демонстрационных конфигурациях вместо объекта modbus может фигурировать объект h264sourceplugin, который является одновременно источником и видео, и субтитров, которые служат для имитации сигналов от оптического датчика, синхронизированных с видеопотоком.

Типовая реализация управляющего скрипта в ходе работы принимает события от оптического датчика (либо субтитры, в случае имитации сигналов от датчика), интерпретирует эти события и принимает решение о включении либо выключении распознавания по своему ЖД пути. Если в конфигурации скрипта выбрано управление распознаванием через API, а не автоматически по событиям, то скрипт обрабатывает управляющие HTTP API запросы (см. раздел 3.16.2 документации), и в качестве реакции на них включает или выключает распознавание. Для управления распознаванием скрипт взаимодействует с процессом v20rlwrec.exe посредством понятных последнему команд. Результаты распознавания, полученные от процесса v20rlwrec.exe, управляющий скрипт аккумулирует, публикует в веб-сервере (см. раздел 3.16.1 документации), и отправляет в качестве событий через интерфейс WebSocket. Скрипт также осуществляет управление записью видео в архив, включая запись в момент появления состава на ЖД пути, и выключая ее после того, как состав проехал.

## Настройка системы распознавания вагонов

### Режим работы

Режим работы системы распознавания номеров вагонов в тестовой конфигурации управляется файлом rlw.json.

*Для того чтобы не потерять изменения конфигурации, рекомендуется при установке Viinex 2.0 для разработки ПО и внедрения – выбирать размещение конфигурации в папке “etc/conf.d”.*

*Установочная программа не изменяет и не удаляет файлы, которые находятся в etc/conf.d, но удаляет и заменяет файлы, находящиеся в папке etc и etc/rlw при удалении/переустановке Viinex 2.0.*

Наиболее важной настройкой системы распознавания номеров вагонов является режим управления распознаванием – по датчику, по API вызову, либо по субтитрам, имитирующим срабатывание датчика. Режим задается значением свойства “init.mode” объекта script, соответствующего ЖД пути. Это свойство может быть установлено в значение “ray”, “api”, “subtitles” для перечисленных случаев. Дополнительно к этому, при выборе вариантов “ray” или “subtitles”, в конфигурации должен присутствовать соответствующий одноименный раздел, указывающий, какие именно события будут инициировать включение и выключение распознавания номера очередного вагона. В частности, в разделе “ray” для управляющего скрипта, работающего по сигналу от оптических датчиков, устанавливается название источника событий о срабатывании датчиков в конфигурации Viinex (как правило, это объект для связи с контроллером Modbus, но может быть и другой источник событий типа DigitalInput), номер дискретного входа на данном контроллере, а также флаг invert, позволяющий инвертировать логику включения/выключения распознавания при замыкании/размыкании контакта.



В случае режима работы “api”, дополнительные настройки не требуются. В этом случае управляющему скрипту (“track1” в тестовой конфигурации) можно послать HTTP запрос POST, содержащий JSON объект с булевым полем “recognize” (true для начала распознавания, и false для окончания), и опциональным значением cookie, в котором может находиться произвольное 64-разрядное целое значение. Это значение служит для того, чтобы впоследствии можно было идентифицировать результат распознавания, соответствующей конкретной команде.

Важной частью конфигурации управляющего скрипта является раздел “substitute”. Его назначение состоит в том, чтобы в результатах распознавания, получаемых от движка, заменить названия источников видео с названия производных служебных объектов video renderer (с которыми может работать движок) на названия исходных камер.



```
    ],
  },
  {
    "type": "script",
    "name": "track1",
    "meta": {
      "type": "RailwayTrack",
      "name": "Путь 1",
      "desc": "Описание пути",
      "channels": ["cam1","cam2"],
      "api": false
    },
    "load": ["etc/rlw/rlwlogic.js"],
    "init": {
      "track": "track1",
      "rem mode": "api|ray|subtitles",
      "mode": "subtitles",
      "ray": {
        "origin": "moxa0",
        "pin": 0,
        "invert": false
      },
      "subtitles": {
        "origin": "cam1"
      },
      "substitute": {
        "rend1": "cam1",
        "rend2": "cam2"
      },
      "store_railcars": 300,
      "train_timeout": 5
    }
  },
  "links":
  [
    ["web0", "track1"],
    ["track1", ["rlwrecengine0", "cam1", "recct11"]]
```

Кроме того, в конфигурации управляющего скрипта может присутствовать раздел meta. Как и для любых других объектов Viinex, этот раздел не передается самому экземпляру объекта и не влияет на его работу, но виден через HTTP API Viinex, и, в частности, служит для настройки работы демонстрационного пользовательского интерфейса системы распознавания номеров ЖД вагонов, который доступен на веб-сервере Viinex по ссылке <http://localhost:8880/vnxrlw/index.html>. В разделе meta для управляющего скрипта должны присутствовать значения: meta.type, равное “RailwayTrack” – тег, по которому веб-приложение среди других объектов Viinex идентифицирует скрипт, связанный с ЖД путем; meta.name и meta.desc – строковые значения, задающие название ЖД пути и его описание, выводимое в пользовательском интерфейсе; meta.channels – список видеоканалов, связанных с данным ЖД путем, который используется для отображения видеопотоков в интерфейсе; наконец, булевское значение meta.api позволяет указать, следует ли в пользовательском интерфейсе отображать кнопку для управления распознаванием, если скрипт настроен на работу в управляемом режиме (“mode”: “api”).

### Настройки движка распознавания

Настройки движка распознавания содержатся в секции objects/type=process,executable=v20rlwrec.exe в подразделе init. В нем должны быть сконфигурированы все ЖД пути, и для каждого пути – все видеоканалы, по которым должно вестись распознавание. Каждый канал настраивается пятью параметрами: video\_source – ссылка на объект renderer, который раскодирует видеопоток в shared memory для распознавателя; поля roi\_top, roi\_bottom, ограничивающие область распознавания в кадре сверху и снизу, а также поля plate\_width и plate\_height, указывающие типичный размер номера. Все размеры указываются в относительных единицах, -- долях от ширины или высоты изображения.

Кроме этого, для работы движка в конфигурации его объекта process должен быть указан параметр licenses, в котором явно указано общее количество видеоканалов, по которым должно происходить распознавание (RailcarRecognizer), и количество путей (RailcarConsolidator):

```
"licenses": [{"RailcarRecognizer",  
2}, {"RailcarConsolidator", 1}]
```

Эти значения должны быть согласованы с количеством путей (RailcarConsolidator) и видеоканалов, по которым ведется распознавание (RailcarRecognizer), в секции init.

Параметр init.name является меткой, идентифицирующей путь (на изображении – значение “track1”). Эта метка должна соответствовать значению параметра init.track в конфигурации управляющего скрипта, -- для каждого ЖД пути. Причиной является то, что каждый экземпляр управляющего скрипта обслуживает один ЖД путь, в то время как процесс v20rlwrec выполняет распознавание для всех ЖД путей на данном сервере.

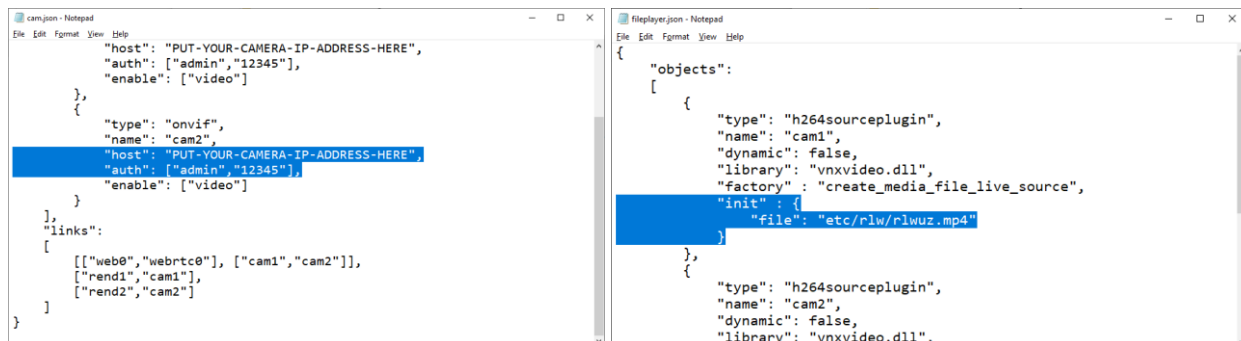
### Настройка других объектов Viinex 2.0

В целом, настройка других объектов Viinex 2.0 для работы системы распознавания номеров ЖД вагонов не отличается от настройки Viinex для других задач, и описана в документации <https://www.viinex.com/Viinex20Guide.pdf>. Здесь приведен короткий список настроек, которые необходимо менять при развертывании системы на любом объекте.

Демонстрационная конфигурация Viinex содержит настройку для распознавания номеров по двум камерам, и разбита на относительно независимые части, как описано в разделе 2.5 документации.

### Источники видео

В демонстрационной конфигурации настройка источников видео описана в файлах cam.json и/или fileplayer.json. Это взаимно исключающие файлы: они используют одинаковые идентификаторы cam1, cam2 для источников видео. В файле cam.json определена настройка для подключения системы к двум ONVIF камерам. При использовании этого варианта, в данном файле необходимо изменить значения “host” (установить в нем реальные IP адреса видеокamer в локальной сети), а также, вероятно, значения “auth” (установить в нем логин и пароль для соответствующей камеры). Подробно настройка подключения к ONVIF камерам описана в разделе 2.1.2 документации.



При использовании демонстрационного видеоролика вместо камер, может быть использован файл fileplayer.json. В нем может потребоваться изменение пути к файлу, содержащему демонстрационный ролик. В демонстрационной конфигурации предполагается, что ролик будет помещен в папку etc/rlw, то есть рядом с файлами демонстрационной конфигурации.

Файлы cam.json и fileplayer.json содержат взаимно исключающие элементы конфигурации, поэтому при использовании одного из этих файлов, второй должен быть исключен из конфигурации: удален из



папки, содержащей конфигурационные файлы, либо его расширение должно быть изменено с .json на какое-либо иное (например, .json\_, с подчеркиванием на конце).

Для каждого источника видео, по которому необходимо осуществлять распознавание номеров вагонов, должен быть создан объект `rendered`, который будет декодировать H264 видеопоток и публиковать его в разделяемой области памяти. В демонстрационной конфигурации такие объекты настроены в файле `rend.json`. В этом файле может потребоваться изменение при выборе разрешения камер, отличного от FullHD. Также, в этом файле нужно добавить дополнительные экземпляры объектов `rendered` для настройки распознавания по четырем видеоканалам.

### Источники событий

При использовании оптического датчика для разделения ЖД состава на вагоны, интеграция такого датчика с Viinex возможна через подключение специализированного Ethernet-контроллера (стандартного Modbus TCP, или реализующего другой поддерживаемый в Viinex протокол VK Module Socket 1). Подробности настройки подключения к такому контроллеру описаны в разделе 2.1.11 документации.

В демонстрационной конфигурации подключение Modbus контроллера описано в файле `ray.json`. В реальной установке в данном файле необходимо установить значение IP адреса контроллера, а также, возможно, количество сухих контактов, которые нужно считывать.

Конкретный номер GPIO контакта, который будет использован для управления распознаванием номеров вагонов на каждом отдельном ЖД пути, указывается в настройках управляющего скрипта в разделе `init.ray` (свойство `origin` для указания идентификатора объекта Modbus контроллера, и свойство `pin` для указания номера сухого контакта). Там же может быть указано свойство `invert`, инвертирующего логику интерпретации сигнала от оптического датчика. Может потребоваться изменение данного свойства в зависимости от установки датчика, нормально замкнут либо нормально разомкнут.

В случае имитации сигнала от датчиков посредством дорожки с субтитрами в демонстрационном видеоролике, в настройках управляющего скрипта выбирается `mode`: `subtitles`, и в разделе `init.subtitles` в поле `origin` должен быть указан идентификатор объекта `h264sourceplugin`, который является также источником событий-субтитров. Формат субтитров для имитации сигнала от луча прост: в субтитрах должно быть текстовое значение «0» (одна цифра ноль) для имитации сигнала о замыкании оптического датчика (т.е. вагона нет), и текстовое значение «1» (либо 2, 3, и т.д.), -- числовое значение, отличное от нуля, записанное в десятичной системе счисления без дополнительных символов, пробелов и т.п., -- для имитации сигнала о размыкании оптического датчика (т.е. вагон есть).



```
ray.json - Notepad
File Edit Format View Help
{
  "objects":
  [
    {
      "type": "modbus",
      "name": "moxa0",
      "host": "192.168.0.115",
      "inputs": 4
    }
  ],
  "links": [
    ["moxa0", ["web0", "track1"]]
  ]
}
```



```
ray.json - Notepad
File Edit Format View Help
},
"load": ["etc/rlw/rlwlogic.js"],
"init": {
  "track": "track1",
  "rem mode": "api|ray|subtitles",
  "mode": "ray",
  "ray": {
    "origin": "moxa0",
    "pin": 0,
    "invert": false
  },
  "subtitles": {
    "origin": "cam1"
  },
  "substitute": {
    "rend1": "cam1",
  }
}
```

## Видеоархив и запись

Настройки видеоархива и записи содержатся в демонстрационной конфигурации в файле `rec.json`.

Это типовые настройки полностью описанные в разделах 2.1.6 и 2.1.7 документации. В реальной установке в файле `rec.json` может потребоваться изменить путь к папке с видеоархивом `Viinex` (свойство `folder` у элемента с типом `storage`), а также

предельные значения для хранимого объема видео, в абсолютных цифрах – свойство `max_size_gb`, -- размер на диске, в гигабайтах, отведенный под видеоархив, -- либо в относительных цифрах, -- свойство `keep_free_percents`, -- проценты свободного места на диске, которые следует держать свободными при заполнении архива. Подробно смысл этих настроек описан в разделе 2.1.6 документации.

Объект `recctl` управляет записью группой камер в видеоархив. Отдельный экземпляр этого объекта нужно создать для каждого ЖД пути. В настройках объекта `recctl` может потребоваться изменение значения `prerecord`, задающего, сколько секунд видео должно быть записано ДО получения управляющим скриптом информации о начале ЖД состава.

Кроме того, при добавлении камер, в том числе обзорных, эти новые камеры должны быть связаны в разделе `links` конфигурации `Viinex` с объектом `recctl` для соответствующего ЖД пути (в примере выше – строка `"links": [..., ["recctl1", ["cam1", "cam2"]]`), -- список связанных камер может быть расширен.



```
{
  "type": "storage",
  "name": "stor0",
  "folder": "C:/viinexvideorlw",
  "filesize": 16,
  "limits": {
    "keep_free_percents": 20,
    "max_size_gb": 2
  }
},
```



```
},
{
  "type": "recctl",
  "name": "recctl1",
  "prerecord": 2,
  "postrecord": 1
}
],
"links":
[
  ["web0", "stor0"],
  ["stor0", "recctl1"],
  ["recctl1", ["cam1", "cam2"]]
]
}
```

## Программный интерфейс

Взаимодействие с системой распознавания номеров вагонов в `Viinex 2.0` осуществляется по протоколу HTTP и WebSocket. Данное взаимодействие, специфичное для задачи распознавания номеров вагонов, осуществляет управляющий скрипт. Исходный код данного скрипта поставляется в открытом виде и доступен для изменения клиентами, поэтому разработчики могут расширить API системы при необходимости.

### Получение результатов по запросу

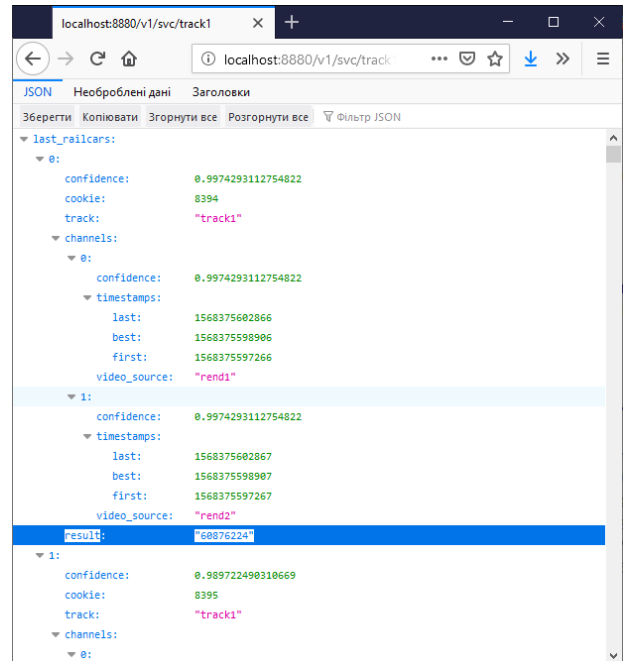
Для получения результатов распознавания по каждому отдельному ЖД пути можно использовать HTTP запрос GET к управляющему скрипту, обслуживающему данный ЖД путь. Результаты распознавания возвращаются в форме JSON объекта, содержащего самый последний результат распознавания (в поле `last_result`), а также массив из некоторого числа последних результатов распознавания (в поле `last_railcars`). Структура объектов, описывающих результат распознавания по каждому вагону, в данных полях одинакова.

Результат распознавания по каждому вагону содержит поля:

- `track` – идентификатор ЖД пути, на котором был получен результат
- `result` – собственно текстовое поле, содержащее распознанный номер



- **confidence** – числовое значение в диапазоне от 0 до 1, характеризующее уверенность движка распознавания в результате, в порядковой шкале, т.е. смысл имеет не само значение данного числа, а отношения порядка между этими числами для различных результатов распознавания: чем больше значение **confidence**, тем больше уверенность движка в результате
- **cookie** – целое 64-разрядное число, идентифицирующее акт распознавания. При работе в автоматическом режиме, когда управляющий скрипт инициирует начало и окончание распознавания при получении событий от оптического датчика, значение **cookie** выбирается управляющим скриптом (в типовой реализации это просто целочисленный счетчик). При работе в управляемом режиме, когда распознавание инициируется через API, значение **cookie** может быть задано внешним ПО, и, таким образом, может служить для идентификации результата распознавания конкретного вагона
- **channels** – массив, содержащий для каждого вагона информацию по деталям распознавания по каждому из видеоканалов. А именно, по каждому из видеоканалов указывается:
  - **video\_source** – идентификатор объекта **renderer**, на видеопотоке от которого был получен результат
  - **confidence** – уверенность движка в результате по видео с данного видеоканала
  - **timestamps** – объект, содержащий три временные метки, **first**, **last** и **best**, указывающие время первого кадра, обработанного для данного вагона по данному видеоканалу, время последнего кадра для данного вагона, а также временную метку кадра, на котором был получен наилучший результат. Эти временные метки могут быть использованы для получения снимков и проигрывания видеороликов, записанных в видеоархиве.



### Получение результатов распознавания в реальном времени

Для считывания результатов распознавания по мере того, как эти результаты возникают в ходе работы системы, можно использовать протокол **WebSocket** и соответствующий программный интерфейс, описанный в разделе 3.17 документации. После подключения по протоколу **WebSocket** к веб-серверу **Viinex**, приложение должно подписаться на интересующие его события, используя, например, команду

```
[ "subscribe", { "topics": [ "RailcarNumberRecognition", "Train" ] } ]
```

После этого приложение начнет получать в свой **WebSocket** канал события о результатах распознавания, а также о начале и окончании прохождения состава, генерируемые управляющим скриптом системы распознавания номеров ЖД вагонов. Каждое событие о результатах распознавания является отдельным результатом распознавания по отдельному вагону. Содержащиеся в нем поля описаны в предыдущей секции настоящего документа.

### Получение снимков и видео из видеоархива

После окончания записи видео в архив, из последнего могут быть получены отдельные кадры, а также проиграны либо экспортированы видеоролики, на которых зарегистрировано прохождение отдельных вагонов либо ЖД состава в целом.

Программный интерфейс для получения кадров и роликов из видеоархива описан в разделах 3.7.1, 3.5.5 и 3.5.6 документации Viinex 2.0. В частности, для получения кадра из видеоархива следует использовать HTTP GET запрос к URL вида

<http://localhost:8880/v1/svc/stor0/cam1/snapshot?timestamp=1568332477603>

где параметр `timestamp` должен содержать временную метку кадра, полученную, например, из свойства `timestamps.best` результата распознавания вагона, как описано в параграфе «Получение результатов по запросу» настоящего раздела. Идентификаторы `stor0` и `cam1` в данном запросе также должны соответствовать конкретному видеоархиву и видеокамере в конфигурации Viinex.

Для проигрывания и экспорта видеороликов из архива Viinex можно использовать, соответственно, URL вида

<http://localhost:8880/v1/svc/stor0/cam1/stream.m3u8?begin=1568377513426&end=1568377517106>

<http://localhost:8880/v1/svc/stor0/cam1/export?format=isom&begin=1568377513426&end=1568377517106>

где параметры `begin` и `end` должны содержать временную метку начала и конца запрашиваемого видеофрагмента. Первая ссылка (`stream.m3u8`) выдаст HLS поток и может быть использована для проигрывания видео в браузере с помощью, например, скрипта `hls.js`. Вторая ссылка позволяет скачивать (экспортировать) MP4 файл, генерируемый непосредственно в момент скачивания, на лету.

## Управление распознаванием через API

Если, вместо использования сигналов от оптических датчиков для управления распознаванием в автоматическом режиме, у внешнего ПО есть необходимость управлять началом и окончанием распознавания, -- управляющий скрипт должен в конфигурации иметь свойство, задающее режим работы `“mode”`, установленное в значение `“api”`.

В результате такой настройки, управляющий скрипт может принимать HTTP запросы POST в свой адрес (<http://localhost:8880/v1/svc/track1> в демонстрационной конфигурации). В теле такого запроса скрипт будет ожидать JSON объект, содержащий обязательное булевское поле `“recognize”`, принимающее значение `true` для начала распознавания, и `false` для окончания распознавания вагона. Кроме того, данный объект может содержать поле `“cookie”`, заполненное произвольным 64-разрядным целым числом. Это число будет впоследствии указано в результате распознавания соответствующего вагона. Поскольку результаты распознавания возвращаются асинхронно (как события, либо по запросу, но все равно вне связи с запросом, инициировавшим распознавание), -- этот механизм позволяет связать команды, которые были даны для распознавания номера конкретного вагона, с результатами распознавания этого вагона.