**Описание процессов Node-RED**

Система управления проездами «Q-YMS»

Версия 1.0 (КАТ)

**Содержание**

Содержание

[1. Общая информация о системе 3](#_Toc79691048)

[2. Концептуальная архитектура системы 4](#_Toc79691049)

[2.1 Стойка проезда 5](#_Toc79691050)

[2.2 Зона проезда 5](#_Toc79691051)

[3. Настройка Q-YMS 6](#_Toc79691052)

[3.1 Добавление зоны проезда 6](#_Toc79691053)

[3.2 Добавление стойки проезда 7](#_Toc79691054)

[3.3 Добавление доступов 8](#_Toc79691055)

[3.4 Основные настройки 9](#_Toc79691056)

[3.5 Настройка модуля Курск-Агро 10](#_Toc79691057)

# Общая информация

Для гибкой настройки логики работы без изменения исходного кода системы, а также интеграции с внешними системами используется сервер потокового программирования Node-RED (<https://nodered.org>). Node-RED предоставляется по лицензии Apache 2.0. Лицензия разрешает коммерческое использование.

В составе системы Q-YMS Node-RED поставляется в составе образа docker. Основное взаимодействие осуществляется через HTTP API (RESTfull API).

Процессы сгруппированы по разным потокам, в зависимости от их назначения.

# Описание процессов

# Поток ДетекторНомераRFID

В данном потоке реализованы HTTP-обработчики для получения данных от системы распознавания номеров Trassir и сервиса RFID Q-LTK. Узлы служат для отладки сообщений на определённом сегменте потока

используется для отправки кода ответа при запросах к HTTP-обработчикам.

 позволяют отправлять произвольные данные в поток. Используются для отладки.

# RFID Detector

HTTP Endpoint **/qyms/rfid** принимающий POST запросы в формате JSON, содержащие данные от системы Q-LTK.

# PlateNumber Detector

HTTP Endpoint **/** **qqyms/plate-number** принимающий POST запросы в формате JSON, содержащие данные ГРЗН от системы Trassir.

# RFID\_Normalizer

Выполняет нормализацию данных полученных от Q-LTK в структуру необходимую для отправки на проверку в Q-YMS.

# ALPR\_Normalizer

Выполняет нормализацию данных полученных от Trassir в структуру необходимую для отправки на проверку в Q-YMS. В данном узле осуществляется сопоставление UID канала и стойки (по атрибуту camera\_channel).

# Поток ОсновнойПроцессСтойки

Данный поток является основным процессом, детектирующим транспортное средство и интерпретирующий сигналы от датчиков стоек. Также в данном потоке осуществляется настройка параметров контроллера и взаимодействие с системой Q-YMS.

Основным блоком реагирующим на события является SNMP Trap Server, который получает сообщения от контроллеров MOXA



Блок ниже вызывает задержку выполнения потока.



# Параметры

Основной блок параметров. В данном блоке задаются основные параметры сервера, сопоставление входов и выходов контроллера, параметры привязки данных к стойкам контроллера.

# Запрос данных по стойкам

В данном блоке выполняется запрос на получение списка стоек, их параметров и атрибутов.

# Преобразование данных в JSON

В данном блоке выполняется преобразование строки JSON в объект JS.

# СопоставлениеСтоек\_PLC\_по\_ID

В данном блоке выполняется сопоставление параметров стоек с параметрами потока.

2.2.5 Преобразование в CSV  
В данном блоке выполняется преобразование сообщений контроллеров в CSV объект.

2.2.6 Установка состояний датчиков стойки (1)   
В данном блоке выполняется обработка состояний датчиков для первой стойки управляемой контроллером. Устанавливаются значения для стойки параметрам потока.

2.2.7 Установка состояний датчиков стойки (2)   
В данном блоке выполняется обработка состояний датчиков для второй стойки управляемой контроллером. Устанавливаются значения для стойки параметрам потока.

2.2.8 Вычисляем hash состояния

В данном блоке выполняется вычисление значение hash текущего состояния стойки. Используется при необходимости исключения отправки данных с одинаковым значением.

2.2.8 Вычисляем hash состояния

В данном блоке выполняется вычисление значение hash текущего состояния стойки. Используется при необходимости исключения отправки данных с одинаковым значением.

2.2.9 Проверяем смену состояний датчиков

В данном блоке выполняется проверка текущего и предыдущего значения датчиков состояния проезда. Если состояние изменилось выполняется алгоритм при смене состояний.

2.2.10 setSensorSource\_IN

В данном блоке выполняется установка источника смены состояний для въездного датчика.

2.2.11 setSensorSource\_OUT

В данном блоке выполняется установка источника смены состояний для выездного датчика.

2.2.12 sensor\_IN\_state

В данном блоке выполняется определение значения состояния и перенаправление потока в зависимости от значения для въездного датчика стойки.

2.2.13 sensor\_OUT\_state

В данном блоке выполняется определение значения состояния и перенаправление потока в зависимости от значения для выездного датчика стойки.

2.2.14 Блок проверки последовательности проезда

В данном блоке выполняется определение последовательности сработки датчиков проезда. Если датчики проезда перешли в состояние свободно выполняется перенаправление на команду закрытия шлагбаума.

2.2.15 Установка метки времени датчиков

В данном блоке выполняется установка метки времени при сработке датчиков проезда. На основе данным меток вычисляется последовательность проезда (вперед или назад).

2.2.16 Определяем команду на закрытие

В данном блоке выполняется определение команды на закрытие стрелы шлагбаума.

2.2.17 Проверка положения стрелы

В данном блоке выполняется определение положение стрелы и датчиков проезда. Если хотя бы один из датчиков, на момент проверки, перейдет в состояние «Занято» команда на закрытие будет отменена.

2.2.18 Эмулятор закрытия стрелы шлагбаума

В данном блоке выполняется эмуляция сигнала закрытия при установке параметра «fake\_barrier\_closed».

2.2.19 Проверка состояния датчиков

В данном блоке выполняется определение состояния датчиков проезда. Если хотя бы один из датчиков, на момент проверки, перейдет в состояние «Занято» команда эмуляции закрытия будет прервана.

2.2.20 Подготовка данных состояний для отправки

В данном блоке выполняется подготовка данных состояний контроллеров для привязки состояний к конкретной стойке.

2.2.21 Отправка данных с состояниями

В данном блоке выполняется отправка данных в систему Q-YMS через HTTP.

2.2.22 Подготовка данных состояний для отправки

В данном блоке выполняется подготовка и привязка данных детекторов RFID и ГРНЗ для привязки данных к конкретной стойке.

2.2.23 Подготовка данных состояний для отправки

В данном блоке выполняется проверка данных датчиков и последовательность их сработки при закрытии шлагбаума.

2.2.24 Признак корректности проезда

В данном блоке выполняется перенаправление потока в зависимости от признака корректности проезда.

2.2.25 Сброс данных детектора

В данном блоке подготавливаются данные для сброса значений детектора перед отправкой Q-YMS.

2.2.26 Подготовка данных для сессии

В данном блоке подготавливаются данные для передачи данных в обработчик сессий проезда в Q-YMS.

2.2.27 Подтверждение проезда

В данном блоке осуществляется отправка данных для обработки сессии в систему Q-YMS.

2.2.28 Отправка подтверждения в модуль КУРСК

В данном блоке осуществляется отправка данных для подтверждения проезда в модуль КУРСК.

# Поток Q-YMS\_Handler

В данном потоке осуществляется обработка данным между потоками и событиями от системы Q-YMS.

2.3.1 Обработчик команд Q-YMS

В данном блоке осуществляется получение данных от системы Q-YMS.

2.3.2 Выбор события для обработки

Осуществляется выбор потока обработки события полученных от Q-YMS:

**TransportEntryRequestMessage –** при получении сообщения от RMQ с типом TransportEntryRequestMessage

**WeighResultsMessage –** привязка результатов взвешивания к сессии проезда для последующей отправки в RMQ.

**LaboratorySamplingMessage -** привязка результатов отбора проб к сессии проезда для последующей отправки в RMQ.

**Barrier –** команда на управление шлагбаумом.

**Vehicle\_Gone –** уведомление о завершении работы детекторов RFID и ГРЗН.

2.3.3 Нормализация данных

В данном блоке осуществляется приведение данных к формату необходимому для управления шлагбаумом в ручном режиме.

2.3.4 Подготовка данных для проверки

В данном блоке осуществляется подготовка данных для запроса возможности проезда по данным детекторов RFID/ГРНЗ.

2.3.5 Проверка свободна ли зона

В данном блоке осуществляется отправка запроса на проверку свободна ли зона. Для зон имеющих тип КПП (Checkpoint) может присутствовать только одно ТС в течении сессии.

2.3.6 Проверка данных в модуле Курс-Агро

В данном блоке осуществляется запрос данных проезда в модуле КУРСК-АГРО.

2.3.7 Привязка GUID к новой сессии

В данном блоке осуществляется привязка данных GUID документа модуля КУРСК-АГРО к сессии проезда для дальнейшей обработки процесса.

2.3.8 Подготовка данных для запроса

В данном блоке осуществляется подготовка данных для проверки атрибутов сессии по данным передаваемым в сообщении **LaboratorySamplingMessage.**

2.3.9 Запрос на проверку данных сессии.

В данном блоке осуществляется отправка запроса на проверку данных сессии соответствующих сообщению **LaboratorySamplingMessage.** Если данные присутствуют в сессии осуществляется открытие шлагбаума для выезда автомобиля.

# Поток MoxaTriggers

Данный поток обеспечивает взаимодействие между исполнительными контроллерами и стойками проезда. Задается соответствие между стойками и выходами контроллеров управления.

2.4.1 Инициализация параметров (Красный)

В данном блоке осуществляется определение UID стойки для последующего сопоставления с необходимым контроллером. UID принудительно преобразовывается в необходимый формат через regexp.

2.4.2 Выбор стойки по UID

В данном блоке осуществляется определение необходимой стойки по UID для отправки команды управления контроллеру. В функциональных блоках на выходе осуществляется установка необходимых номеров выходов контроллера.

2.4.3 Отправка команды в PLC

В данном блоке осуществляется отправка запроса с командой для контроллера управления исполнительными устройствами.

2.4.4 Подтверждение выполнения команды

В данном блоке выполняется пост обработка выполнения команды контроллера

2.4.5 Принудительное управление шлагбаумом

В данном блоке выполняется принудительное открытие шлагбаума даже в режиме блокировки.