**Общее руководство пользователя системы «РоадАР Аналитика - Остановка ТС»**

**Содержание**

[**Введение 3**](#_je8m6wcagh1x)

[**1 Назначение и условия применения системы “РоадАР Аналитика - Остановка ТС” 4**](#_3dy6vkm)

[1.1 Назначение системы 4](#_1t3h5sf)

[1.2 Базовый функционал 4](#_4d34og8)

[1.3 Функциональные характеристики 4](#_czhijeca4yrq)

[1.4 Программное обеспечение рабочего места 4](#_2s8eyo1)

[**1.5 Виды пользователей 5**](#_17dp8vu)

[1.6 Описание системы 5](#_3rdcrjn)

[**2 Инструкция по установке системы 7**](#_8hnbzf3klg5a)

[2.1 Установка Docker 7](#_b98xe46k97cc)

[2.2 Скачивание и сбор частей архив, загрузка образа и его запуск 8](#_z83rduyufh6j)

[**3 Контактная информация производителя программного продукта 10**](#_ecomzkltjoke)

[3.1 Юридическая информация 10](#_3rdcrjn)

[3.2 Контактная информация службы технической поддержки 10](#_2tow4ctptd8)

# 

# Введение

**“РоадАР Аналитика - Остановка ТС”** – система, разработанная компанией ООО “РоадАР” для распознавания и фиксации преднамеренного прекращения движения транспортного средства (далее - ТС) на время до 5 минут, а также на большее, если это необходимо для посадки или высадки пассажиров, либо загрузки или разгрузки ТС, прекращение движения на проезжей части, посреди дороги, на автострадах, в других местах, где остановка и стоянка запрещены.

**Система “РоадАР Аналитика - Остановка ТС”** поддерживает механизмы взаимодействия со сторонними информационными системами. Посредством вызова API поставляемой библиотеки. А также текущая версия системы предполагает выбор и реализацию конкретного механизма взаимодействия с внешними системами по согласованию с заказчиками.

# 1 Назначение и условия применения системы “РоадАР Аналитика - Остановка ТС”

## 1.1 Назначение системы

Обработка изображений для выявления и фиксации преднамеренного прекращения движения ТС на время до 5 минут, а также на большее, если это необходимо для посадки или высадки пассажиров, либо загрузки или разгрузки ТС, прекращение движения на проезжей части, посреди дороги, на автострадах, в других местах, где остановка и стоянка запрещены.

## 1.2 Базовый функционал

* обработка изображений;
* выявление и фиксация преднамеренного прекращения движения ТС на время до 5 минут, а также на большее, если это необходимо для посадки или высадки пассажиров, либо загрузки или разгрузки ТС, прекращение движения на проезжей части, посреди дороги, на автострадах, в других местах, где остановка и стоянка запрещены;
* информационный обмен с внешними системами.

## 1.3 Функциональные характеристики

Текущая версия системыпозволяет решать задачи по распознаванию преднамеренного прекращения движения ТС на время до 5 минут, а также на большее, если это необходимо для посадки или высадки пассажиров, либо загрузки или разгрузки ТС, прекращение движения на проезжей части, посреди дороги, на автострадах, в других местах, где остановка и стоянка запрещены.

## 1.4 Программное обеспечение рабочего места

Библиотека может быть встроена в программное обеспечение, установленное на рабочем месте пользователя. В этом случае клиентская часть **системы “РоадАР Аналитика - Остановка ТС”** может использоваться на любом рабочем месте, имеющем подключение к сети Internet (или сети передачи данных предприятия). Разрешающая способность видеосистемы и монитора – не ниже 1280х1024. Рекомендуется широкоформатный монитор.

| Вид ПО | Программный продукт |
| --- | --- |
| ОС (приведены варианты) | Windows 10  Linux (Ubuntu, Debian, Альт, ROSA, UBLinux, ICLinux). |

*Таблица 1. Системные требования*

Данные требования могут меняться в зависимости от особенностей программного обеспечения, которое использует библиотеку системы **“РоадАР Аналитика - Остановка ТС”**.

## 1.5 Виды пользователей

Пользователем системы является пользователь программного обеспечения, в которое встраивается библиотека **системы “РоадАР Аналитика - Остановка ТС”**. Описание доступных возможностей API выполняется для пользователя, обладающего максимально возможными правами по доступу к программе.

## 1.6 Описание системы

API Системы написано на C++.

***~ src/core/modules/analytic\_specific\_params.hpp:***

…

namespace an::core {

/\*!

\* \brief Общая структура для хранения оющих парметров для дорожных аналитик

\*/

struct RoadAnalyticParams {

RoadAnalyticParams() = default;

/\*\*

\* \brief Конструктор с параметрами

\* \param [in] grnd указатель на класс для проекции объектов на дорогу

\* \param [in] roadLanes множество полос, поданных на вход аналитике

\* выпаших грузов

\*/

RoadAnalyticParams(std::shared\_ptr<GroundCalibration> &grnd,

std::vector<RoadLane> lanes)

: ground(grnd), roadLanes(std::move(lanes)) {

spdlog::info("Road lanes count for road analytic: {}", roadLanes.size());

}

/// Conversion API -> Core

explicit RoadAnalyticParams(const an::RoadAnalyticParams &params) {

std::transform(params.roadLanes.begin(),

params.roadLanes.end(),

std::back\_inserter(roadLanes),

[](const an::RoadLane &rl) { return RoadLane(rl); });

}

bool operator==(const RoadAnalyticParams &other) const {

bool eqGround = ground == other.ground

|| (ground && other.ground && \*ground == \*other.ground);

return eqGround && roadLanes == other.roadLanes;

}

/// указатель для проецирования объектов на дорогу (через калибровочную

/// матрицу)

std::shared\_ptr<GroundCalibration> ground;

/// множество полос дороги

std::vector<RoadLane> roadLanes;

};

/\*!

\* \brief Структура для хранения параметров аналитики "Внезапная остановка

\* транспортного средства"

\*/

struct VehicleStopAnalyticParams : RoadAnalyticParams {

/\*\*

\* \brief Конструктор с параметрами

\* \param [in] roadParams общие параметры для дорожной аналитики

\*/

explicit VehicleStopAnalyticParams(RoadAnalyticParams roadParams)

: RoadAnalyticParams(std::move(roadParams)) {}

/// Conversion API -> Core

explicit VehicleStopAnalyticParams(

const an::VehicleStopAnalyticParams &params)

: RoadAnalyticParams(params),

maxHistorySize(params.maxHistorySize),

maxFramesToCount(params.maxFramesToCount),

upperStopSpeedBound(params.upperStopSpeedBound),

upperMoveSpeedBound(params.upperMoveSpeedBound),

maxBoxWidth(params.maxBoxWidth),

maxBoxHeight(params.maxBoxHeight),

framesToSkip(params.framesToSkip),

secondsToSkip(params.secondsToSkip) {}

bool operator==(const VehicleStopAnalyticParams &other) const {

return RoadAnalyticParams::operator==(other)

&& maxHistorySize == other.maxHistorySize

&& maxFramesToCount == other.maxFramesToCount

&& upperStopSpeedBound == other.upperStopSpeedBound

&& upperMoveSpeedBound == other.upperMoveSpeedBound

&& maxBoxHeight == other.maxBoxHeight

&& maxBoxWidth == other.maxBoxWidth

&& framesToSkip == other.framesToSkip

&& secondsToSkip == other.secondsToSkip;

}

int maxHistorySize{30};

// how many frames to check before stop event

int maxFramesToCount{9};

// from 0 to n km/h

double upperStopSpeedBound{2};

double upperMoveSpeedBound{5};

double maxBoxWidth{6};

double maxBoxHeight{6};

// secondsToSkip is preferred over framesToSkip (in case both were non-zero)

uint64\_t secondsToSkip{0};

int framesToSkip{500};

};

}

***~ src/core/modules/vehicle\_stop/vehicle\_stop\_analytic.hpp***

#pragma once

#include "modules/analytic.hpp"

#include "modules/analytic\_specific\_params.hpp"

#include "modules/tracker/vehicle/track\_analyzer/track\_analyzer.hpp"

#include "nn/openvino/classifier.hpp"

namespace an { namespace core {

struct CarStopInfo {

uint64\_t trackId = 0;

std::string uuid;

cv::Rect2f bbox;

cv::Point2f bot;

int stopDetectCount = 0;

double time = 0;

int64\_t skipUntilFrameNum = -1;

double skipUntilTimePoint = 0;

std::vector<CustomPolygonIdPoints> lanes;

CarStopInfo(uint64\_t trackId, std::string uuid)

: trackId(trackId), uuid(std::move(uuid)), bbox(cv::Rect2f()) {}

CarStopInfo() = default;

};

std::vector<CustomPolygonIdPoints> getLanesPolyForPointFromLaneStatus(

const cv::Point2f &p, const std::vector<RoadLaneStatus> &roadLanes);

class VehicleStopAnalytic : public Analytic {

public:

explicit VehicleStopAnalytic(const std::string &analyticId,

const AnalyticEventQueue &analyticEventQueue,

bool needSavePreview,

const VehicleStopAnalyticParams &roadParams,

std::shared\_ptr<InferenceEngine::Core> &engine,

std::string &xmlPath);

void processFrame(const pFrame &) override;

private:

static bool isCarInJamLane(const pFrame &,

Track &,

const std::vector<RoadLaneStatus> &);

AnalyticEvents produceEvents(const pFrame &);

/// проверка были ли в ближайшее время детекты остановки авто в этой области

/// @return true - недавно в этой области была остановка

bool checkRecentStopEvents(const CarStopInfo &);

void updateRecentStopEvents(const CarStopInfo &);

void removeRecentStopEvents(const CarStopInfo &);

MatCore drawCarStopPreview(const pFrame &, Track &, bool /\*stopped\*/);

void doDebugDraw(const pFrame &, std::vector<RoadLaneStatus> &);

std::unique\_ptr<TrackAnalyzer> trackAnalyzer\_;

std::unordered\_map<uint64\_t, CarStopInfo> carStopInfoMap;

std::shared\_ptr<Classifier> carBgClassificator\_;

int maxHistorySize{30};

/// сколько времени храним

double keepRecentCarStops\_{300.0};

// how many frames to check before stop event

int maxFramesToCount{9};

// from 0 to n km/h

double upperStopSpeedBound{2};

double upperMoveSpeedBound{5};

double maxBoxWidth{6};

double maxBoxHeight{6};

std::vector<CarStopInfo> recentCarStops;

uint64\_t secondsToSkip{0};

[[deprecated("Use seconds to skip instead")]] int framesToSkip{500};

};

}} // namespace an::core

***~ src/core/modules/vehicle\_stop/vehicle\_stop\_analytic\_controller.hpp***

#pragma once

#include "modules/analytic\_controller.hpp"

namespace an { namespace core {

class VehicleStopAnalyticController : public AnalyticController {

public:

VehicleStopAnalyticController(const FrameQueue &inputQueue,

const FrameQueue &outputQueue,

const AnalyticEventQueue &eventQueue,

std::shared\_ptr<InferenceEngine::Core> &ie,

std::string xmlPath);

~VehicleStopAnalyticController() override;

AnalyticType getControllerAnalyticType() const override;

protected:

std::unique\_ptr<Analytic> makeSpecificAnalytic(

const AnalyticSpecificParams &params) override;

private:

/// объект инференса для классификатора

std::shared\_ptr<InferenceEngine::Core> ie\_;

/// путь до нейросети классификатора заднего фона

std::string netPath\_;

};

}} // namespace an::core

# 2 [Инструкция по установке системы](#_26in1rg)

## 2.1 Установка Docker

В инструкции подразумевается, что пользователь использует ОС Linux, Ubuntu 20.04 (для других дистрибутивов, инструкции могут отличаться)

Обязательным предусловием для установки серверной платформы MDT является установка следующих пакетов:

* ***docker 18.06.1-ce+***;

На Ubuntu их можно установить следующими командами:

| $ sudo apt-get update  $ sudo apt-get install \  apt-transport-https \  ca-certificates \  curl \  gnupg-agent \  software-properties-common  $ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -  $ sudo add-apt-repository \  "deb [arch=amd64] https://download.docker.com/linux/ubuntu \  $(lsb\_release -cs) \  stable"  $ sudo apt-get update  $ sudo apt-get install docker-ce docker-ce-cli containerd.io |
| --- |

Детали для установки Докера и добавления вашего пользователя в группу Докер можно найти по ссылкам:

1. <https://docs.docker.com/compose/install>
2. [https://docs.docker.com/install/linux/docker-ce/ubuntu](https://docs.docker.com/install/linux/docker-ce/ubuntu/).

## 2.2 [Скачивание и сбор частей архив, загрузка образа и его запуск](https://docs.google.com/document/d/1D2HKTmWfVFK2WDGv-CmaTSiSUEOrdlzCtZN6pbzHwXc/edit#heading=h.z83rduyufh6j)

1. Необходимо установить архиватор. В терминале системы Linux команда:

sudo apt install unzip

1. Скачайте папку Документы РоадАР Аналитики <https://www.roadar.info/file-share/5ad2fb14-dd07-427d-afb5-a0feb1f99cbe>Наименование файла: \_\_\_\_\_\_\_\_\_\_\_

Для загрузки файла необходимо авторизоваться. Логин и пароль передаются при покупке системы.

Загруженный файл представляет собой архив с компонентами системы. Пароль от него так же передаётся при покупке системы.

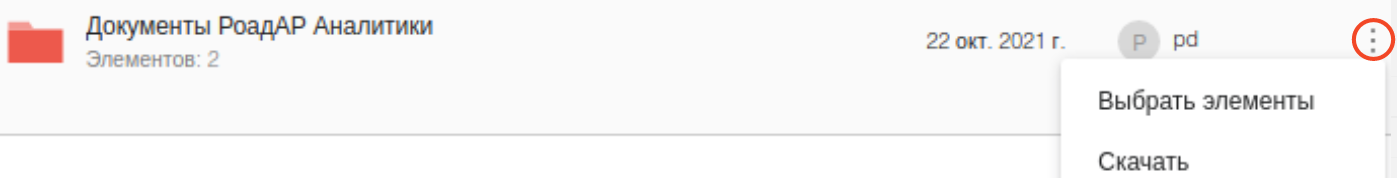
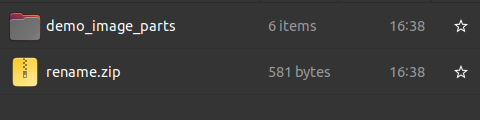


Рисунок 1. Скачивание папки с демо.

1. С помощью терминала зайдите в папку Документы РоадАР Аналитика - Остановка ТС. В текущей рабочей директории вы должны увидеть папку demo\_image\_parts и архив rename.zip



1. Разархивируйте архив rename.zip (пароль не нужен).

| unzip rename.zip |
| --- |

1. Выполните команду (она уберет .zip с конца названия файла):

python3 rename.py

1. Собираем один архив из его частей:

| cat demo\_image\_parts/analytics\_demo.part-\* > analytics\_image.zip |
| --- |

1. Далее разархивируем полученный файл (потребуется ввести пароль):

| unzip analytics\_demo\_img.zip |
| --- |

1. Затем загружаем полученный докер-образ в список образов докера

| docker load -i analytic\_demo |
| --- |

1. Далее запускаем образ:

| docker run --entrypoint=/app/entrypoint.sh analytics:rospotent |
| --- |

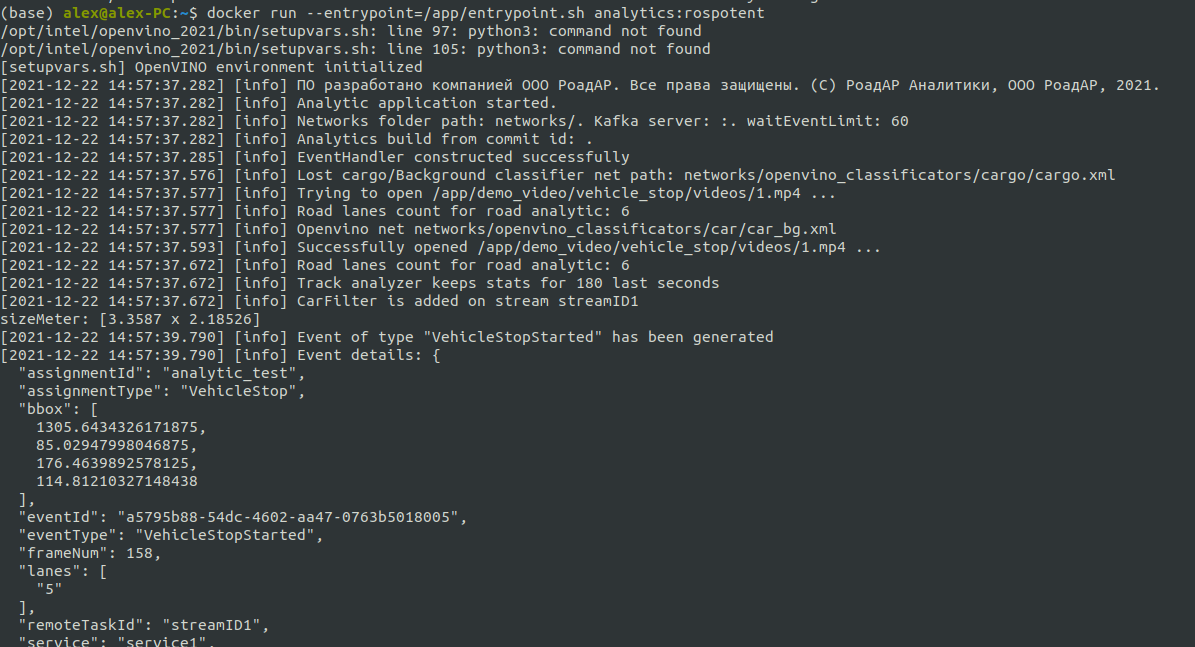


Рисунок 2. Пример вывода в консоль удачного запуска (дождитесь вывода статуса).

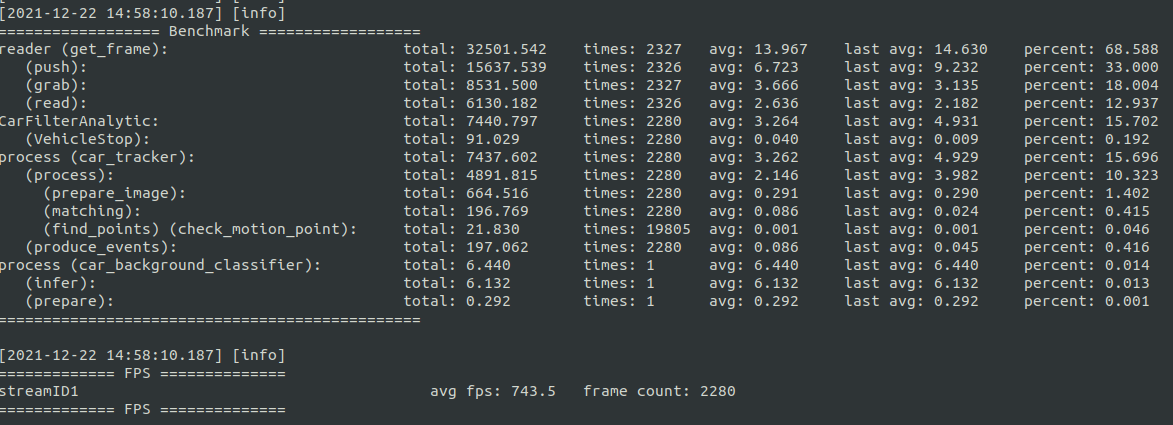


Рисунок 3. Пример вывода текущего статуса (обновляется каждые 2 минуты).

# 

# 3 Контактная информация производителя программного продукта

## 3.1 Юридическая информация

* **Название компании:** ООО «РоадАР».
* **ИНН 1615013172**
* **ОГРН 1161690183665**
* **Юр. адрес:** 420500, г Иннополис, ул Университетская, д 7, офис 332

## 3.2 Контактная информация службы технической поддержки

* **Сайт:** roadar.info
* **Email:** [info@roadar.info](mailto:info@roadar.info)
* **Тел.:** +7-903-307-16-75

Фактический адрес размещения инфраструктуры разработки:

420500, г. Иннополис, ул. Университетская, дом 7, офис 715

Фактический адрес размещения разработчиков:

420500, г. Иннополис, ул. Университетская, дом 7, офис 715

Фактический адрес размещения службы поддержки:

420500, г. Иннополис, ул. Университетская, дом 7, офис 715