**

Колледж космического машиностроения и технологии

КУРСОВОЙ ПРОЕКТ

По дисциплине «Прикладное программирование»

Тема: «**Электронное КПП»**

Выполнил студент

Грингоф Олег Олегович

Группа П2-15

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_(Дата сдачи работы)

Проверил преподаватель:

Гусятинер Леонид Борисович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(оценка)

Королев, 2018

Содержание

[Задание на выполнение курсового проекта. 3](#_Toc518367303)

[Введение 4](#_Toc518367304)

[1. Теоретическая часть 6](#_Toc518367305)

[1.1. Описание библиотеки OpenCV2 6](#_Toc518367306)

[1.2. Каскад Хаара 8](#_Toc518367307)

[1.3. Оптическое распознавание символов. 11](#_Toc518367308)

[1.4. MNIST 13](#_Toc518367309)

[1.5. Описание библиотеки PyQT5 14](#_Toc518367310)

[1.6. Описание библиотеки multiprocessing 16](#_Toc518367311)

[1.7. Global Interpreter Lock (GIL) 17](#_Toc518367312)

[1.8. Процесс и поток 18](#_Toc518367313)

[1.9. Описание библиотеки re 19](#_Toc518367314)

[1.10. Описание библиотеки Collections 20](#_Toc518367315)

[2. Проектная часть 21](#_Toc518367316)

[2.1. Основные требования к программе 21](#_Toc518367317)

[2.2. Описание инструмента разработки программы 22](#_Toc518367318)

[2.3. Описание разработки приложения 24](#_Toc518367319)

[2.4. Описание разработанного приложения 25](#_Toc518367320)

[2.5. Описание функций файла Main.py 28](#_Toc518367321)

[2.6. Описание функций файла filter.py 28](#_Toc518367322)

[2.7. Структура программы 29](#_Toc518367323)

[3. Организационная часть 32](#_Toc518367324)

[3.1. Установка Tesseract - OCR 32](#_Toc518367325)

[3.2. Установка MySQL server 38](#_Toc518367326)

[3.3. Руководство пользователя 53](#_Toc518367327)

[3.4. Рекомендуемые минимальные системные требования. 57](#_Toc518367328)

[Заключение 58](#_Toc518367329)

[Список используемой литературы 59](#_Toc518367330)

[Приложение 60](#_Toc518367331)

[Листинг 1 60](#_Toc518367332)

[Листинг 2 63](#_Toc518367333)

[Листинг 3 64](#_Toc518367334)

# **Задание на выполнение курсового проекта.**

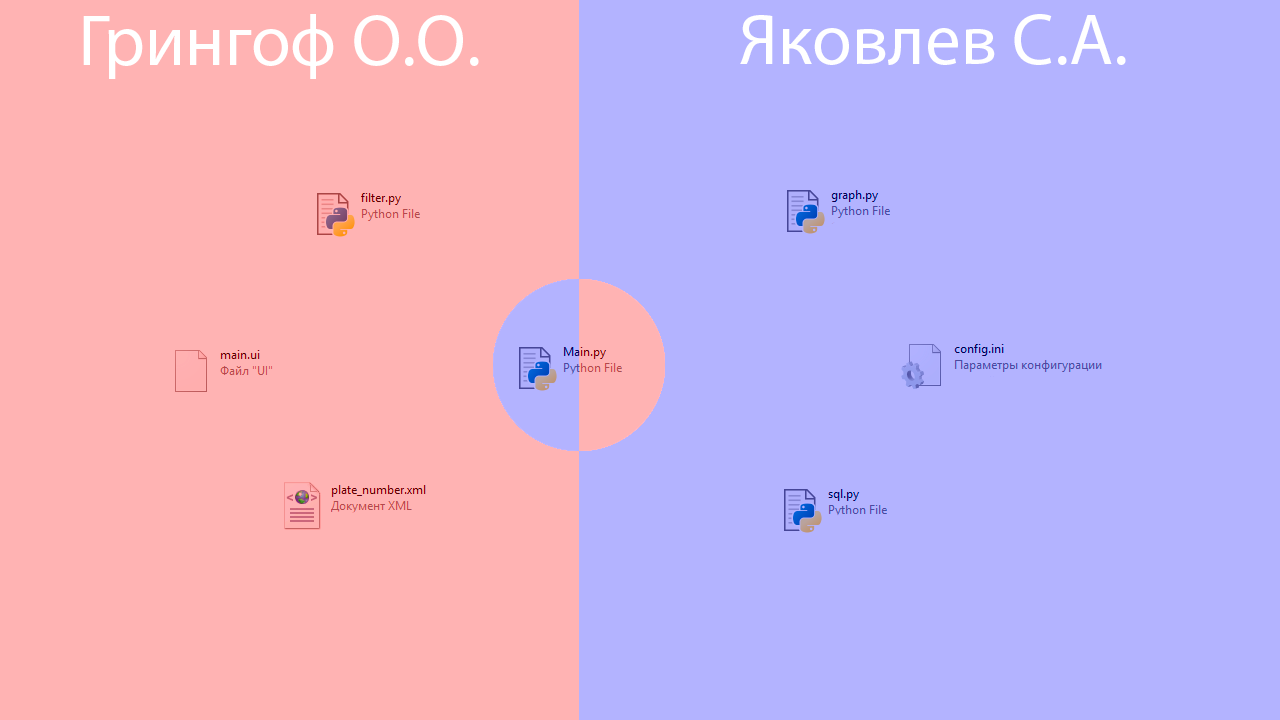
Разработка программы автоматизации КПП с возможностью распознавания государственных номеров.

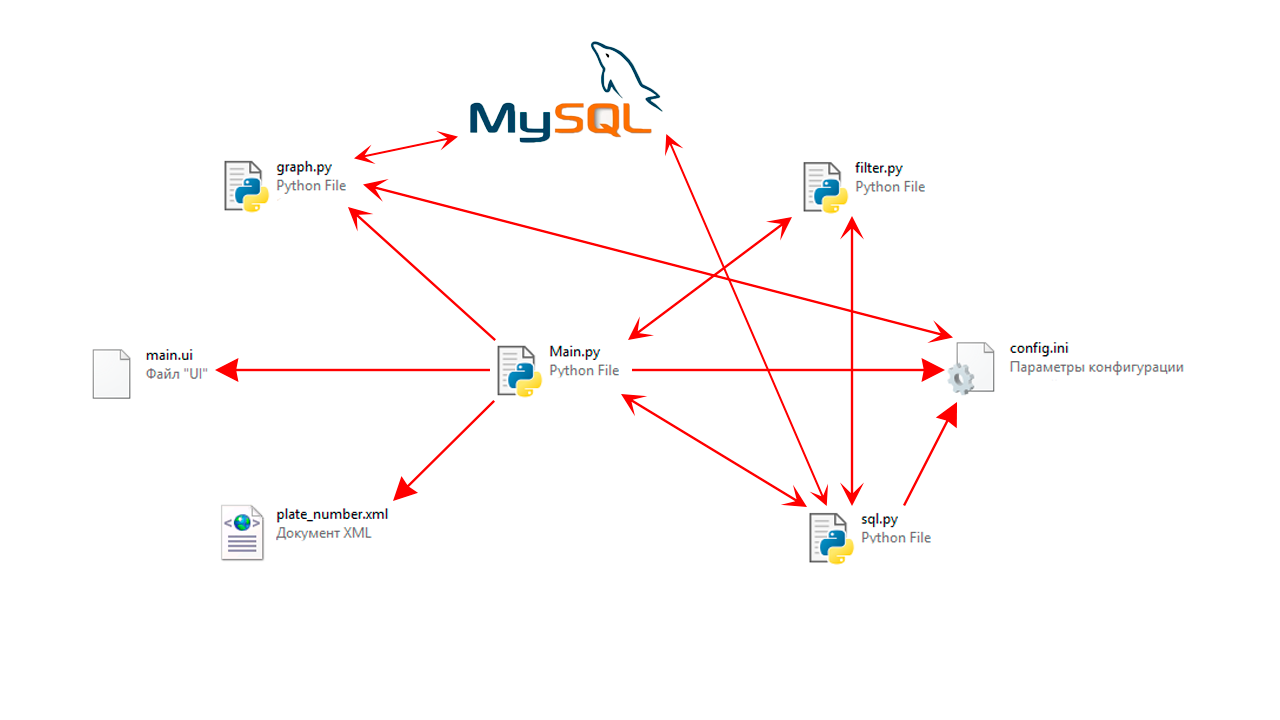
# **Введение**

Появилась идея автоматизации КПП. После поисков в интернете были найдены аналоги, цены на которые оказались весьма завышенные. Было принято решение совместно с Грингоф Олегом создать собственное «Автоматизированное КПП» и выбрали в качестве темы на курсовой проект.

Курсовой проект был разбит на две части:

1. Разрабатывал Грингоф О. О. включает в себя, «Main.py» «filter.py» «main.ui», «plate\_number.xml».
2. Разрабатывал Яковлев С. А. включает в себя, «sql.py» «graph.py» «config.ini».

Рисунок 1 - Структура проекта

Рисунок 2 - Связи в проекте

В процессе используется интегрированная среда разработки программного обеспечения PyCharm, так же свободная компьютерная программа для распознавания текстов Tesseract OCR. Разработка программы будет происходить на языке программирования Python, база данных MySQL.

Для осуществления обозначенной цели служат следующие задачи:

* 1. Описание языка Python, SQL.
  2. Изучение литературы по библиотеке PyQT5, Opencv2, collections, pytesseract, multiprocessing, re.

# **1. Теоретическая часть**

* 1. **Описание библиотеки OpenCV2**

[OpenCV](http://ru.wikipedia.org/wiki/OpenCV) (Open Source Computer Vision) — библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом. Содержит алгоритмы для: интерпретации изображений, калибровки камеры по эталону, устранение оптических искажений, определение сходства, анализ перемещения объекта, определение формы объекта и слежение за объектом, 3D-реконструкция, сегментация объекта, распознавание жестов и т.д.

Основные модули библиотеки:

cxcore — ядро

* базовые операции над многомерными числовыми массивами
* матричная алгебра, генераторы случайных чисел
* Запись/восстановление структур данных в/из XML
* базовые функции 2D графики

CV — модуль обработки изображений и компьютерного зрения

* — операции над изображениями (фильтрация, геометрические преобразования, преобразование цветовых пространств и т. д.)
* анализ изображений (выбор отличительных признаков, морфология, поиск контуров, гистограммы)
* анализ движения, слежение за объектами
* обнаружение объектов, в частности лиц
* калибровка камер, элементы восстановления пространственной структуры

Highgui — модуль для ввода/вывода изображений и видео, создания пользовательского интерфейса

* захват видео с камер и из видео файлов, чтение/запись статических изображений.
* функции для организации простого UI
* Cvaux — экспериментальные и устаревшие функции.
* пространств. зрение: стерео калибрация, само калибрация.
* поиск стерео-соответствия, клики в графах.
* нахождение и описание черт лица.
  1. **Каскад Хаара**

Признаки Хаара - признаки цифрового изображения, используемые в распознавании образов. Своим названием они обязаны интуитивным сходством с вейвлетами Хаара. Признаки Хаара использовались в первом детекторе лиц, работающем в реальном времени.

Исторически сложилось так, что алгоритмы, работающие только с интенсивностью изображения (например, значение RGB в каждом пикселе), имеют большую вычислительную сложность.

Для примера рассмотрим базу данных с человеческими лицами. Общим для всех изображений является то, что область в районе глаз темнее, чем область в районе щек. Следовательно, общим признаком Хаара для лиц является 2 смежных прямоугольных региона, лежащих на глазах и щеках.

На этапе обнаружения в методе Виолы – Джонса окно установленного размера движется по изображению, и для каждой области изображения, над которой проходит окно, рассчитывается признак Хаара. Наличие или отсутствие предмета в окне определяется разницей между значением признака и обучаемым порогом. Поскольку признаки Хаара мало подходят для обучения или классификации (качество немного выше чем у случайной нормально распределенной величины), для описания объекта с достаточной точностью необходимо большее число признаков. Поэтому в методе Виолы — Джонса признаки Хаара организованы в каскадный классификатор.

Ключевой особенностью признаков Хаара является наибольшая, по сравнению с остальными признаками, скорость. При использовании интегрального представления изображения, признаки Хаара могут вычисляться за постоянное время (примерно 60 процессорных инструкций на признак из двух областей).

**Прямоугольные признаки Хаара**

Простейший прямоугольный признак Хаара можно определить как разность сумм пикселей двух смежный областей внутри прямоугольника, который может занимать различные положения и масштабы на изображении. Такой вид признаков называется 2-прямоугольным. Виола и Джонс также определили 3-прямоугольные и 4-прямоугольные признаки. Каждый признак может показать наличие (или отсутствие) какой-либо конкретной характеристики изображения, такой как границы или изменение текстур. Например, 2-прямоугольный признак может показать, где находится граница между темным и светлым регионами.

**Наклонные признаки Хаара**

Линхарт и Майд представили идею наклоненных (45 градусов) признаков Хаара. Это было сделано для увеличения размерности пространства признаков. Способ оказался удачным и некоторые наклонные признаки были способны лучше описывать объект. Например, 2-прямоугольный наклонный признак Хаара может показать наличие края, наклоненного на 45 градусов.

Мессом и Барзак дополнили концепцию наклонных признаков Хаара. Хоть идея и является математически верной, на практике при использовании признаков под разными углами возникают проблемы. Для ускорения вычислений, детектор использует изображения низкого разрешения, что приводит к ошибке округления. Исходя из этого, наклонные признаки Хаара обычно не используются.

Эта техника удобна тем, что позволяет использовать предварительно рассчитанное описание характеристик искомого изображения многократно и является одним из самых быстрых способов распознавания объектов на изображении (при условии использования предварительно рассчитанного каскада).

Принцип работы алгоритма заключается в том, что по всему изображению прогоняется прямоугольное сканирующее окно. Во всех тестируемых положениях рассчитывается яркость зон, описанных в примитивах поискового каскада. Если карта яркости фрагмента изображения в окне похожа на искомый примитив - информация об этом сохраняется. По итогам сканирования изображения найденные совпадения анализируются, группируются и проверяются на соответствие условиям взаимного расположения элементов в каскаде. Если расположение найденных элементов совпадает с эталонным, зоны, содержащие требуемые признаки, подаются на выход алгоритма.

Помимо высокой скорости распознавания преимуществом использования данного алгоритма является то, что каскады для большинства основных фигур и предметов (включая части человеческого тела) уже неоднократно были рассчитаны с достаточной точностью, что позволяет не проводить обучение и подбор соответствующих выборок самостоятельно, а просто взять готовый классификатор и использовать в программе.

Помимо выделения силуэтов на изображении, выполняется отделение движущихся объектов от фона с помощью алгоритма, описанного P. Kadew и R. Bowden в их работе «An improved adaptive background mixture model for real-time tracking with shadow detection»

* 1. **Оптическое распознавание символов.**

Оптическое распознавание символов (optical character recognition - OCR) — механический или электронный перевод изображений рукописного, машинописного или печатного текста в текстовые данные, использующиеся для представления символов в компьютере (например, в текстовом редакторе). Распознавание широко применяется для преобразования книг и документов в электронный вид, для автоматизации систем учёта в бизнесе или для публикации текста на веб-странице. Оптическое распознавание символов позволяет редактировать текст, осуществлять поиск слов или фраз, хранить его в более компактной форме, демонстрировать или распечатывать материал, не теряя качества, анализировать информацию, а также применять к тексту электронный перевод, форматирование или преобразование в речь. Оптическое распознавание текста является исследуемой проблемой в областях распознавания образов, искусственного интеллекта и компьютерного зрения.

Системы оптического распознавания текста требуют калибровки для работы с конкретным шрифтом; в ранних версиях для программирования было необходимо изображение каждого символа, программа одновременно могла работать только с одним шрифтом. В настоящее время больше всего распространены так называемые «интеллектуальные» системы, с высокой степенью точности распознающие большинство шрифтов. Некоторые системы оптического распознавания текста способны восстанавливать исходное форматирование текста, включая изображения, колонки и другие нетекстовые компоненты.

Три основные технологии распознавания символов:

1. Шаблонная. Во входном изображении выделяются растровые изображения отдельных символов, сравниваются со всеми шаблонами, имеющимися в базе, выбирается шаблон с наименьшим количеством точек, отличных от входного изображения. Шаблонные системы проще в реализации, устойчивы к дефектам изображения, имеют высокую скорость обработки входных данных, но надежно распознают только те шрифты, шаблоны которых им известны.
2. Структурная. Объект описывается как граф, узлами которого являются элементы входного объекта, а дугами - пространственные отношения между ними. Структурные системы высоко чувствительны к графическим дефектам изображения, нарушающим составляющие элементы. Для этих систем, в отличие от шаблонных и признаковых, до сих пор не созданы эффективные автоматизированные процедуры обучения.
3. Фонтанное преобразование. Совмещает в себе достоинства шаблонной и структурной систем. Любой воспринимаемый объект рассматривается как целое, состоящее из частей, связанных между собой определенными отношениями.
   1. **MNIST**

Для калибрации систем распознавания текста создана стандартная база данных «MNIST», состоящая из изображений рукописных цифр.

База данных «MNIST» — объёмная база данных образцов рукописного написания цифр. База данных является стандартом, предложенным «Национальным институтом стандартов и технологий США» с целью калибрации и сопоставления методов распознавания изображений с помощью машинного обучения в первую очередь на основе нейронных сетей. Данные состоят из заранее подготовленных примеров изображений, на основе которых проводится обучение и тестирование систем. База данных была создана после переработки оригинального набора чёрно-белых образцов размером 20x20 пикселей NIST. Создатели базы данных NIST, в свою очередь, использовали набор образцов из Бюро переписи населения, к которому были добавлены ещё тестовые образцы, написанные студентами университетов. Образцы из набора MNIST были нормализированы, прошли сглаживание и приведены к серому полутоновому изображению размером 28x28 пикселей.

База данных MNIST содержит 60000 изображений для обучения и 10000 изображений для тестирования.

* 1. **Описание библиотеки PyQT5**

PyQt5 - это набор Python библиотек для создания графического интерфейса на базе платформы Qt5 от компании Digia.

Он доступен для Python 2.x и 3.x. Библиотека Qt является одной из самых мощных библиотек GUI (графического интерфейса пользователя).

PyQt5 реализован в виде набора python-модулей. Эта библиотека имеет более 620 классов и 6000 функций и методов.

Это мульти платформенная библиотека, которая работает на всех основных операционных системах, в том числе Unix, Windows и Mac OS.

Модуль QtCore содержит ядро с неграфической функциональностью. Этот модуль используется для работы с временем, файлами, папками, различными типами файлов, потоками, адресами URL, MIME-типами и процессами.

QtGui содержит классы для интеграции систем окон, обработки событий, 2D-графики, базовой обработки изображений, шрифтов и текста.

Модуль QtWidgets содержит классы, которые обеспечивают набор UI-элементов для создания классических пользовательских интерфейсов.

QtMultimedia содержит классы для управления мультимедиа-содержимым, а также API для доступа к камере или функциональности радио.

Модуль QtBluetooth содержит классы для поиска устройств, соединения и взаимодействия с ними.

Модуль QtNetwork содержит классы для сетевого программирования. Эти классы облегчают работу с клиентами и серверами TCP/IP и UDP, делая сетевое программирование более легким и адаптивным.

QtPositioning содержит классы для определения местоположения с использованием всевозможных источников, таких как спутники, Wi-Fi или текстовые файлы.

Модуль Enginio реализует клиентскую библиотеку для организации доступа к облачным сервисам Qt – Manaded Application Runtime.

Модуль QtWebSockets содержит классы, которые реализуют протокол WebSocket.

QtWebKit содержит классы для реализации веб-браузера, основанного на библиотеке WebKit2.

Модуль QtWebKitWidgets содержит классы для WebKit1, основанные на реализации веб-браузера, для использования в приложениях, базирующихся на QtWidgets.

QtXml содержит классы для работы с XML файлами. Этот модуль предоставляет реализацию сразу для двух API: SAX и DOM.

Модуль QtSvg содержит классы для отображения содержимого SVG-файлов. Scalable Vector Graphics (SVG) – это язык для описания двумерной графики и графических приложений в XML.

Модуль QtSql предоставляет классы для работы с базами данных.

* 1. **Описание библиотеки multiprocessing**

Модуль «multiprocessing» позволяет создавать несколько процессов, что дает возможность распараллеливания программ. Суть в том, что, создание процесса позволяет обойти GIL (Global Interpreter Lock) и воспользоваться возможностью использования нескольких процессоров на компьютере. Пакет multiprocessing также включает ряд API, которых нет в модуле threading.

* 1. **Global Interpreter Lock (GIL)**

Global Interpreter Lock (GIL) — это способ синхронизации потоков, который используется в некоторых интерпретируемых языках программирования, например, в Python и Ruby. GIL является самым простым способом избежать конфликтов при одновременном обращении разных потоков к одним и тем же участкам памяти. Когда один поток захватывает его, GIL, работая по принципу мьютекса (объект, ограничивающий количество потоков, которые могут войти в заданный участок кода), блокирует остальные. Нет параллельных потоков — нет конфликтов при обращении к разделяемым объектам. Очерёдность выполнения потоков определяет интерпретатор в зависимости от реализации, переключение между потоками может происходить: когда активный поток пытается осуществить ввод-вывод, по исчерпании лимита выполненных инструкций или по таймеру.

* 1. **Процесс и поток**

Процесс – это исполняемая копия приложения. Поток – отдельное исполняемое задание внутри процесса. Процесс может содержать множество исполняемых потоков. После запуска приложения исполняется главный поток, который далее может порождать другие потоки. Каждый процесс обладает собственной памятью. Потоки же, которые запущены внутри процесса, разделяют память между собой. Процесс внутри операционной системы обладает собственным идентификатором. Потоки существуют внутри процесса и обладают идентификатором внутри работающего приложения. Каждый из потоков имеет свой собственный стек (он не делит его с другими потоками и другие потоки не могут в него залезть) и собственный набор регистров (поток не изменит значения регистра другого потока во время работы).

* 1. **Описание библиотеки re**

Регулярное выражение это cтрока, задающая шаблон поиска подстрок в тексте, используется для анализа текстов на предмет соответствия находящейся в них текстовой информации некоему шаблону.

Регулярные выражения имеют два основных направления применения:

* анализ и поиск в текстовых массивах
* проверка данных на соответствие шаблону

В Python для работы с регулярными выражениями используется модуль re, который входит в стандартную библиотеку Python.

re.match(pattern, string) - метод поиска по заданному шаблону в начале строки.

Для составления регулярных выражений используются данные символы:

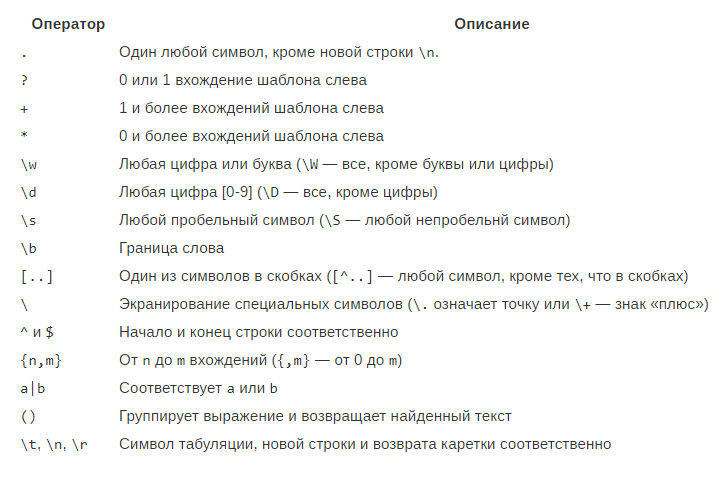


Рисунок 3 - Символы

* 1. **Описание библиотеки Collections**

Модуль collections - предоставляет специализированные типы данных, на основе словарей, кортежей, множеств, списков. Есть несколько специальных методов:

collections.Counter - вид словаря, который позволяет нам считать количество неизменяемых объектов (в большинстве случаев, строк)

elements() - возвращает список элементов в лексикографическом порядке.

most\_common([n]) - возвращает n наиболее часто встречающихся элементов, в порядке убывания встречаемости. Если n не указано, возвращаются все элементы.

Наиболее часто употребляемые шаблоны для работы с Counter():

* sum(c.values()) - общее количество.
* c.clear() - очистить счётчик.
* list(c) - список уникальных элементов.
* set(c) - преобразовать в множество.
* dict(c) - преобразовать в словарь.
* c.most\_common() - n наименее часто встречающихся элементов.
* c += Counter() - удалить элементы, встречающиеся менее одного раза.

# **Проектная часть**

## **Основные требования к программе**

Программа должна корректно работать, обладать быстродействием, исключать ошибочное распознания.

Основания для разработки:

Проанализировав множество программ и узнав цены, было принято решение разработать собственную программу по автоматизации КПП.

Стадии и этапы разработки:

1. Разработка технического задания.
2. Разработка функций.
3. Разработка интерфейса.
4. Написание кода, исправление ошибок, тесты.

## **Описание инструмента разработки программы**

PyCharm — интегрированная среда разработки для языка программирования Python. Предоставляет средства для анализа кода, графический отладчик, инструмент для запуска юнит-тестов и поддерживает веб-разработку на Django. PyCharm разработана компанией JetBrains на основе IntelliJ IDEA.

PyCharm работает под операционными системами Windows, Mac OS X и Linux.

Возможности:

* Статический анализ кода, подсветка синтаксиса и ошибок.
* Навигация по проекту и исходному коду: отображение файловой структуры проекта, быстрый переход между файлами, классами, методами и использованиями методов.
* Рефакторинг: переименование, извлечение метода, введение переменной, введение константы, подъём и спуск метода и т. д.
* Инструменты для веб-разработки с использованием фреймворка Django
* Встроенный отладчик для Python
* Встроенные инструменты для юнит-тестирования
* Разработка с использованием Google App Engine
* Поддержка систем контроля версий: общий пользовательский интерфейс для Mercurial, Git, Subversion, Perforce и CVS с поддержкой списков изменений и слияния

PyCharm Professional Edition имеет несколько вариантов лицензий, которые отличаются функциональностью, стоимостью и условиями использования.

PyCharm Professional Edition является бесплатным для образовательных учреждений и проектов с открытым исходным кодом.

Существует также бесплатная версия Community Edition, обладающая усеченным набором возможностей. PyCharm предоставляет умную проверку кода, быстрое выявление ошибок и оперативное исправление, вкупе с автоматическим рефакторингом кода, и богатыми возможностями в навигации. Редактор кода PyCharm предоставляет первоклассную поддержку для Python, JavaScript, CoffeeScript, TypeScript, CSS, и других популярных языков.

* 1. **Описание разработки приложения**
     1. **Идея.**

Узнав о библиотеки «OpenCV2», начал её изучения, рассмотрев некоторые примеры, зародилась идея разработки собственного проекта на базе данной библиотеке.

* + 1. **Цели на проект.**

Изучение материалов, поиск подходящих библиотек, изучения языка программирования Python.

* + 1. **Изучение материала.**

Библиотека «OpenCV2», цифровая обработка изображений, каскад Хаара, MySQL, работа с файлами конфигурации.

* + 1. **Разбиение проекта на две части.**

Было принято решение взять данный проект на курсовую работу за 3 курс. Проект разделили на две части между: Грингоф Олегом и Яковлевым Сергеем. Деление проекта можно увидеть на [Рисунке 1.](#_Введение)

* + 1. **Разработка функций.**

В связи с тем, что проект разрабатывался несколькими людьми, он должен был быть модульным для дальнейшего совмещения разработанных модулей вместе.

* + 1. **Разработка базы данных.**

База данных MySQL состоящая из двух таблиц «pd» «pdtime». База данных выполняет два запроса «"select number, Fname, Iname, Oname, post from pd where number = %s", {number}», данный запрос проверяет есть ли такой человек в базе и «"insert into platedata.pdtime (number, time) values (%s, %s)", {str(num), str(ti)}», записывает в базу данных время и дату въезда.

* + 1. **Написание кода, исправление ошибок, тесты**

Было проведено множество тестов, исправлено много ошибок, на данный момент остаются ошибки, которые не исправили.

## **Описание разработанного приложения**

Данная программа является системой контроля КПП.

Когда к КПП подъезжает автомобиль, программа считывает государственный номер автомобиля, проверяет его наличие в Базе Данных разрешённых на въезд автомобилей, если номер есть в системе, пропускает авто на территорию и выводит данные владельца автомобиля (ФИО, государственный номер, дату и время въезда на территорию). Так же эта информация записывается в базу данных, на базе которой в дальнейшем возможно построение графика загруженности КПП по часам.



Рисунок 4 – Схема работы

Направления дальнейшей разработки

Дальнейшая разработка приложения может вестись в нескольких направлениях:

* Работа совместно с человеком.
* Замена человека в периоды низкой загруженности.
* Полная замена человека.

Работа совместно с человеком

В местах повышенной загруженности КПП компаний, риск человеческого фактора вырастает, что может повлечь как замедление работы пропускного пункта, так и ошибки в пропуске.

Это можно минимизировать совместной работой человека и компьютера. Вариантов совместной работы может быть несколько:

* Работник может сверяться с данными приложения
* Работник может часть работы доверить полностью программе
* Работник может полностью доверить работу программе, при этом контролируя ее и окончательный решение о пропуске остается на работнике

Замена человека в периоды низкой загруженности

При работе любого КПП есть время максимальной и минимальной загруженности. В то время, когда при большой нагрузке требуется максимальная производительность, ответственность и внимательность, при низкой загруженности эти требования гораздо меньше, и работу можно полностью возложить на программу. На пример, если ночью загруженность минимальная, то удерживать сотрудников на рабочем месте не имеет смысла и можно полностью передать контроль программе. Это позволит экономить компании на заработной плате сотрудникам за ночной период, ведь из всего штата можно оставить лишь пару человек на случай какого-либо происшествия или сбоя в системе.

Полная замена человека

И наконец можно полностью возложить всю работу на программу, тем самым автоматизировать весь процесс работы КПП, тем самым компания сможет экономить на заработной плате, ведь для обслуживания понадобится только несколько человек, которые могут работать как находясь рядом, так и удаленно.

Задачи, решаемые с помощью распознавания номеров Технология оптического распознавания номеров обеспечивает идентификацию автомобилей, без необходимости установки специальных меток или устройств на автомобили. Основные группы решаемых задач:

• Контроль проезда и мониторинг трафика Цель - получение информации о проехавших ТС + разного рода отчеты и статистика + оповещение в реальном времени (о VIP-клиентах, проверяющих органах, нежелательных авто, разыскиваемых автомобилях)

• Управление доступом на территорию. Цель - пускать на территорию только тех, кому въезд разрешен.

* 1. **Описание функций файла Main.py**

def \_\_init\_\_ () – Конструктор класса «MainWindow () »

def ImgConv () - Конвертирует полученное изображение взывает фильтр поиска номерных знаков и удаляет лишние символы, добавляет значение в базу данных.

def camRead () -Получает изображение с камеры, преобразует в градации серого, меняет палитру цветов из RGB в BGR, поиск государственного номера на изображении, находит границы, обрезает и выводит в отдельном окне, передает полученное изображение в «ImgConv».

def status () - Вызывается в случае обнаружение камеры.

def Run () - Запуск основного цикла программы.

* 1. **Описание функций файла filter.py**

def filter\_re() – фильтрует и вырезает из полученного текста государственный номер автомобиля

* 1. **Структура программы**

Алгоритм работы программы:

* 1. Подъезжает автомобиль к территории предприятия.
  2. Камера фотографирует номер.
  3. С помощью каскада Хаара «plate\_number.xml» opencv распознает номер автомобиля.
  4. Tesseract OCR ищет символы в номере, посылает найденные символы в «filter.py», фильтруются найденные символы с помощь регулярных выражений по алгоритму (1 символ буква, 2,3,4 символ цифра, 5,6 символ буква).
  5. Найденный вариант номера отправляется в базу данных, для проверки данного на совпадение в базе данных.
  6. Если номер присутствует в базе данных, то выводится (ФИО, номер автомобиля, должность, время въезда)



Рисунок 5 – Алгоритм работы программы

Экранная форма.

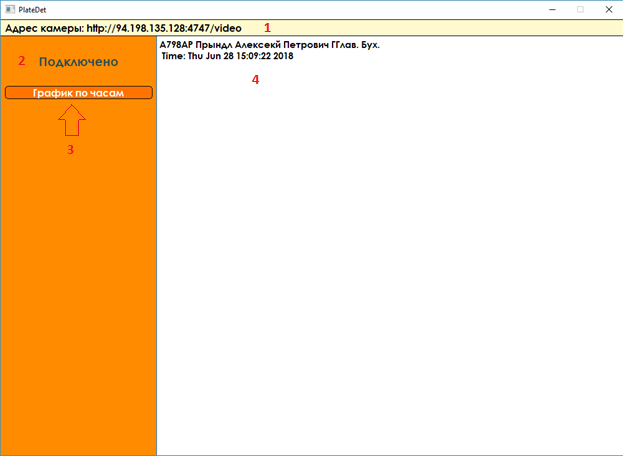


Рисунок 6 – Экранная форма

* 1. Адрес камеры, указывается в файле config.ini, под заголовком [CamSettings] в поле «ip».
  2. Статус подключения, если соединение с камерой установлено выведет «Подключено», иначе выведет «Нет подключение».
  3. Кнопка «График по часам» строит графика загруженности КПП, по часам в течение суток. Функция для построения графика:

def GraphHours():

time, \_ = zapros()

X = [time[str(i)] for i in range(24)]

Y = np.arange(len(X))

L = [i for i in range(0, 24)]

plt.bar(Y, X, align='center')

plt.xticks(Y, L)

plt.show()

* 1. Поле вывода информации о владельце автомобиля из базы данных. Строка вывода информации о владельце автомобиля.

self.textBrowser.append(self.text + '\n Time: ' + ADTime + '\n')

# **3. Организационная часть**

Для работы программы требуется установка дополнительного программного обеспечения:

* 1. Установка Tesseract - OCR.
  2. MySQL Community Server

## **Установка Tesseract - OCR**

1. Запускаем установщик (tesseract-ocr-setup-3.05.01.exe).

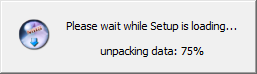


Рисунок 7 - Запуск установщика

1. Выбираем русский язык и нажимаем кнопку «ОК».

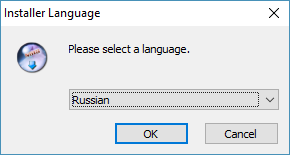


Рисунок 8 - Выбор языка

1. Рекомендации к установке, нажимаем кнопку «Далее».

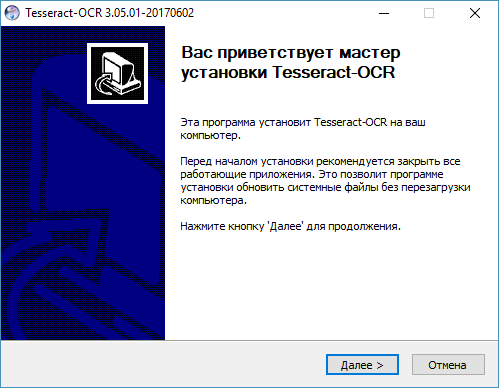


Рисунок 10 - Окно установки

1. Лицензионное соглашение, внимательно прочитываем и соглашаемся, поставив галочку, нажимаем кнопку «Далее».

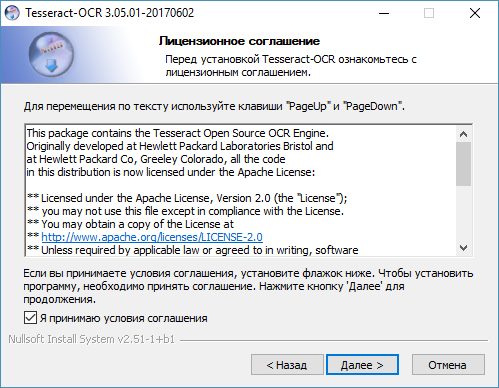


Рисунок 11 - Лицензионное соглашение

1. Выбор пользователей персонального компьютера, которым будет доступен Tesseract-OCR, нажимаем кнопку «Далее».

«Install for anyone using computer» - установка для всех пользователей.

«Install just for me» - установка для текущего пользователя.

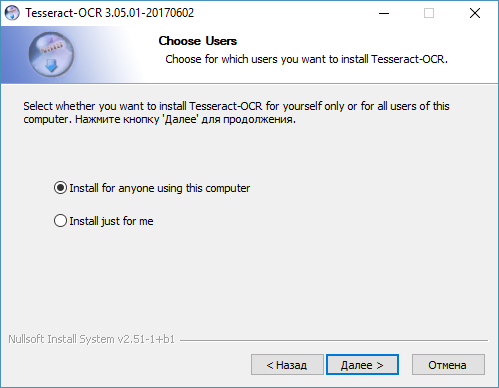


Рисунок 12 - Выбор пользователей

1. Выбор компонентов для установки. В пункте «Additional language data (download)» выбрать пункты «Russian» «English – Middle (1100 - 1500)», нажимаем кнопку «Далее».

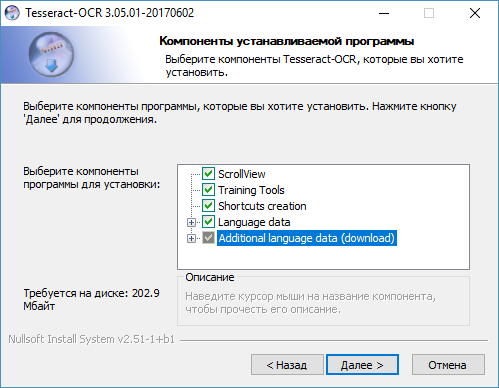


Рисунок 13 - Компоненты установки

1. Выбираем путь установки, нажимаем кнопку «Далее».

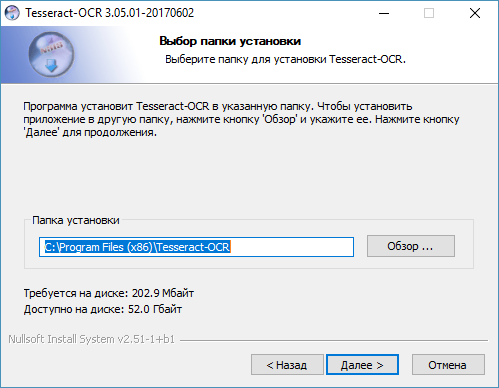


Рисунок 14 - Путь установки

1. Выбираем название папки в меню «Пуска», нажимаем кнопку «Установить».

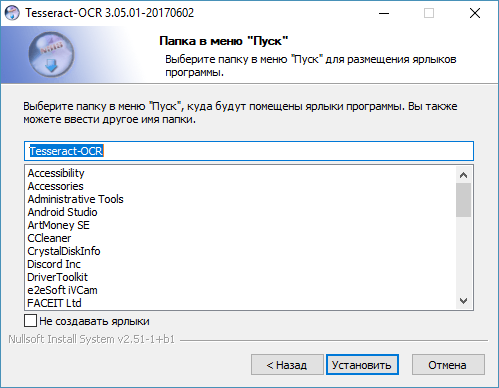


Рисунок 15 - Название папки в меню пуска

1. Ожидание завершение процесса установки.

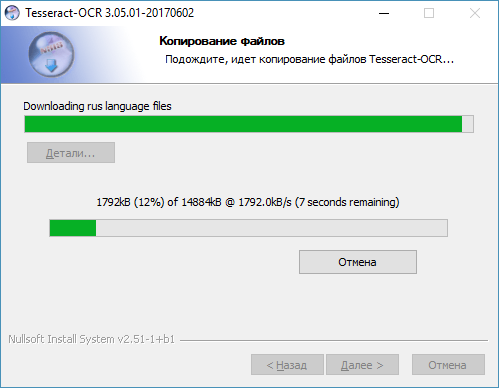


Рисунок 16 - Процесс установки

1. После завершение установки, нажимаем кнопку «Далее».

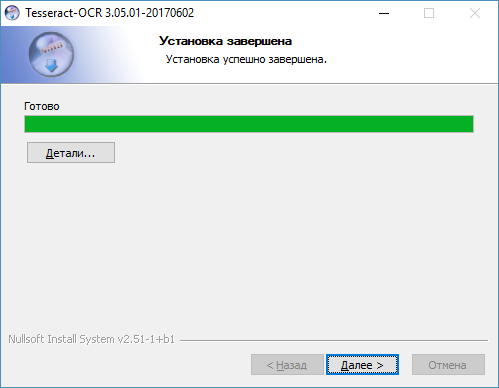


Рисунок 17 - Установка завершена

1. Завершаем работу мастера установки нажатием кнопки «Готово»

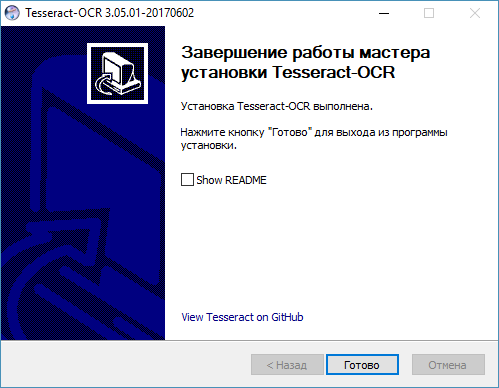


Рисунок 18 - Завершение работы мастера установки

## **Установка MySQL server**

* + 1. Запускаем установщик (mysql-installer-community-5.7.22.1.msi).

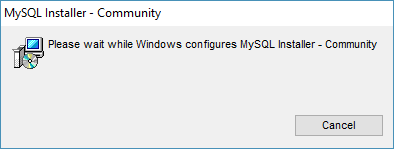


Рисунок 19 - Начало установки

* + 1. Лицензионное соглашение, внимательно прочитываем и соглашаемся, поставив галочку, нажимаем кнопку «Next».

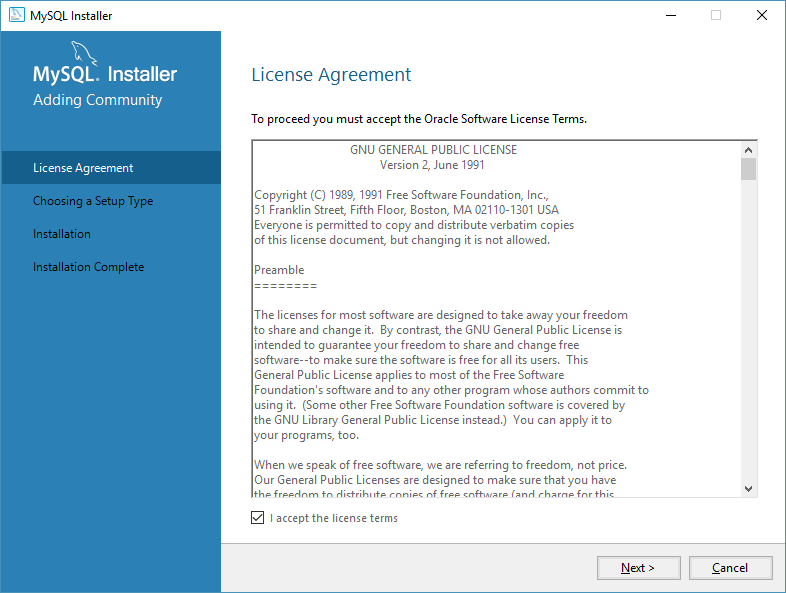


Рис. 20 - Лицензионное соглашение

* + 1. Выбираем тип установки «Developer Default», нажимаем «Next»

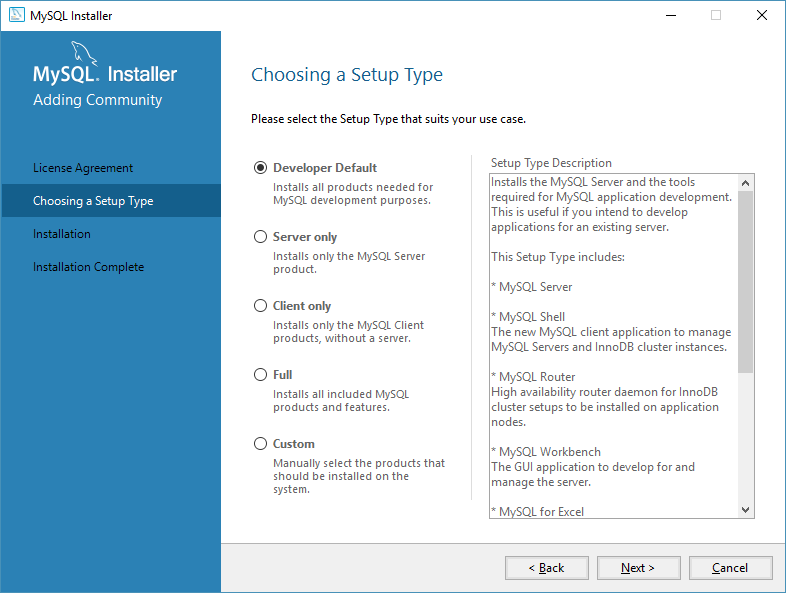


Рисунок 21 - Тип установки

* + 1. Нажимаем «Next» в паявшемся окне нажимаем «Yes»

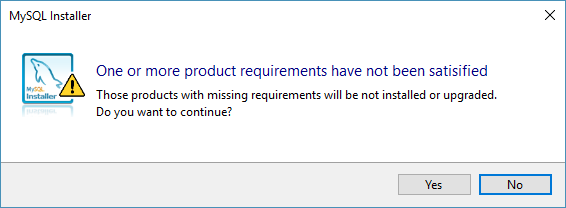


Рисунок 22 - Окно с предупреждением

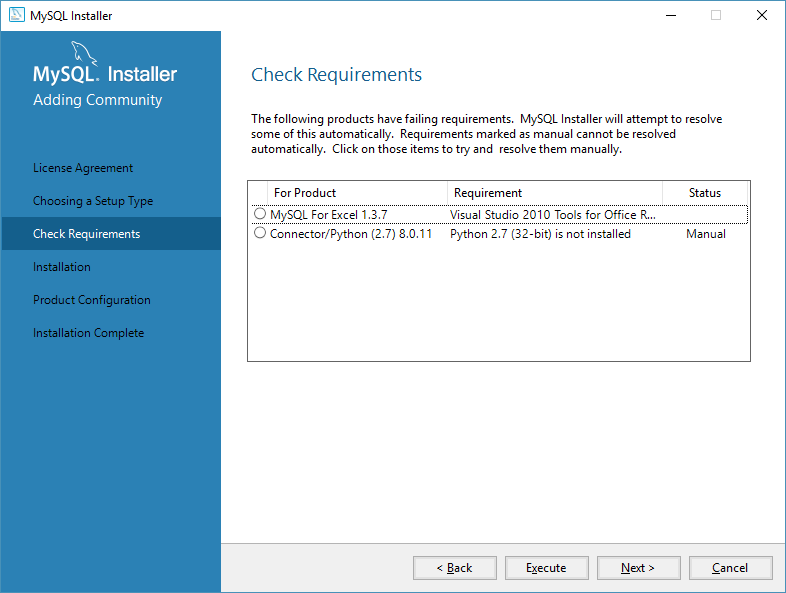


Рисунок 23 - Дополнительные модули

* + 1. Пакеты, которые будут установлены, нажимаем кнопку «Execute».

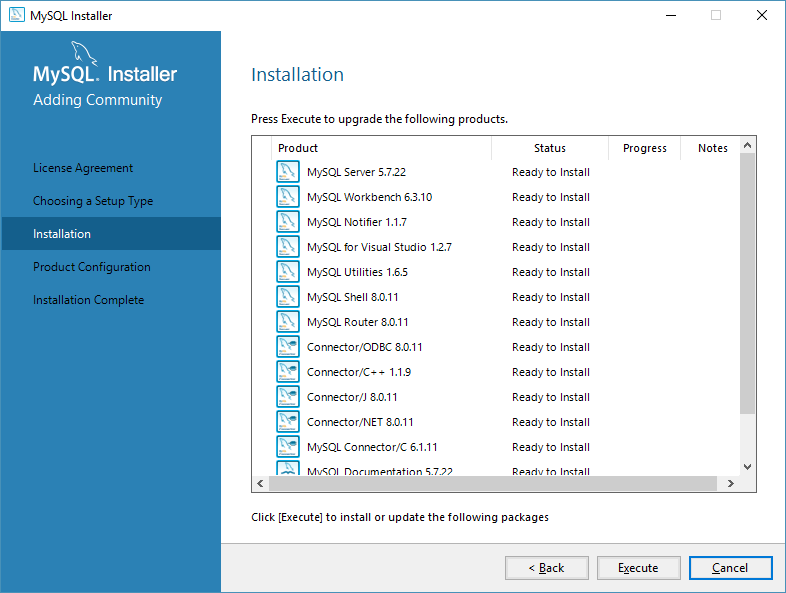


Рисунок 20 - Пакеты

* + 1. Ожидаем завершение процесса установки. После завершения установки всех пакетов, нажимаем «Next»

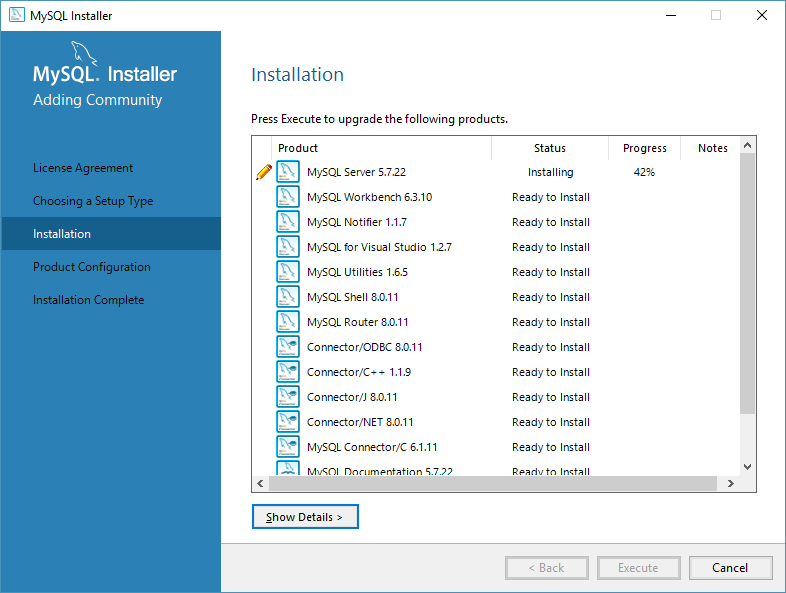


Рисунок 21 - Процесс установки

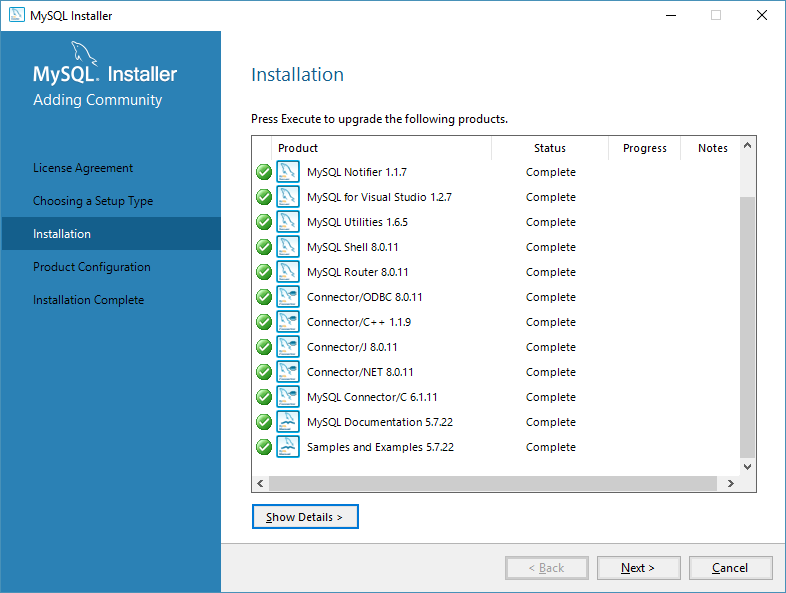


Рисунок 22 - Завершение установки пакетов

* + 1. Можем увидеть конфигурацию сервера.

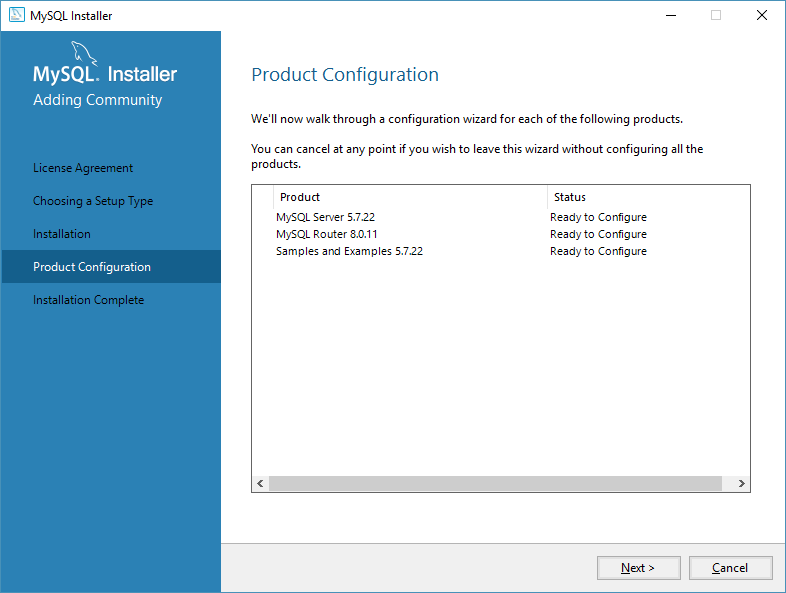


Рисунок 23 - Конфигурация продукта

* + 1. Метод построение сервера. Выбираем «Standalone MySQL Server / Classic MySQL Replication» и нажимаем «Next»

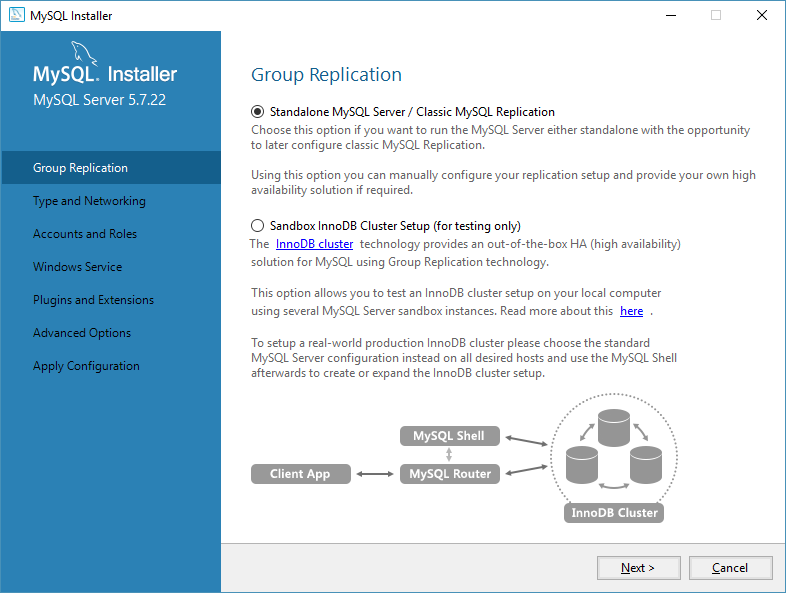


Рисунок 24 - Построение сервера

* + 1. Галочки оставляем по умолчанию и нажимаем кнопку «Next».

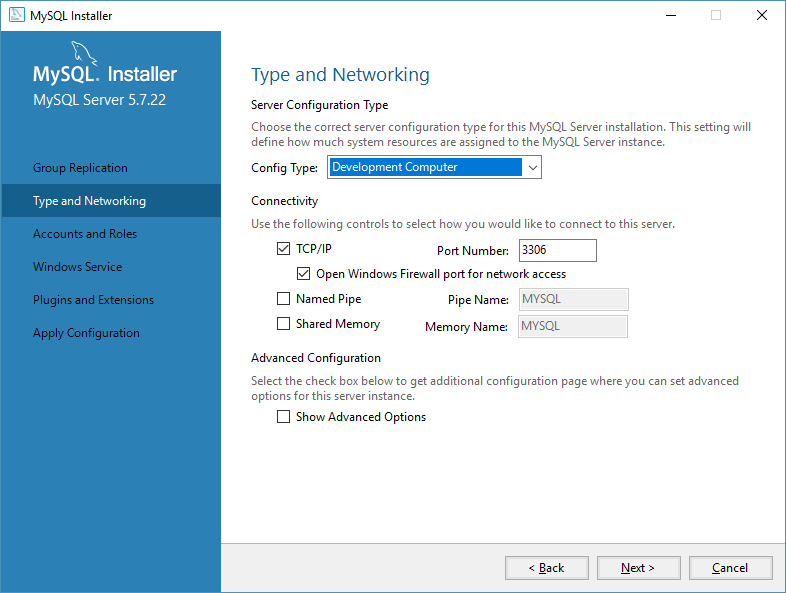


Рисунок 25 - Тип и соединение

* + 1. Нажимаем «Add User». В поле «Username» - вводим имя пользователя. В поле «Host» - выбираем «<All Host (%)>». В поле «Role» - выбираем «DB Admin». В полях «Password» и «Confirm Password» - вводим пароль для базы данных. Нажимаем кнопку «ОК». В полях «MySQL Root Password» и «Repeat Password» придумываем пароль главного пользователя и нажимаем кнопку «Next».

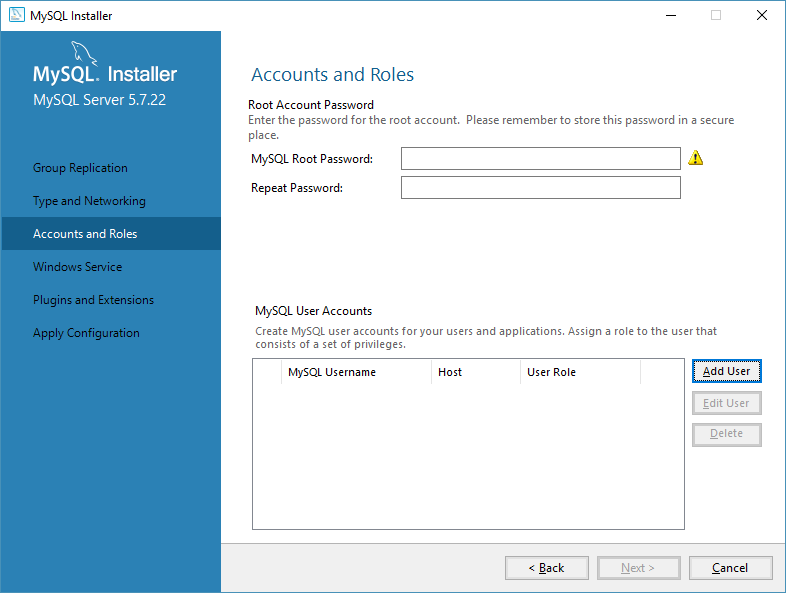


Рисунок 26 - Аккаунты и роли

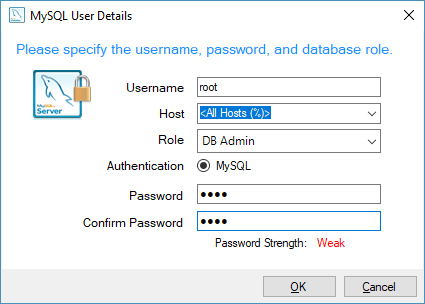


Рисунок. 27. Данные пользователя

* + 1. Ставим галочку «Configure MySQL Server as a Windows Service», придумываем имя серверу в поле «Windows Service Name», далее выбираем «Standard System Account» и нажимаем кнопку «Next».

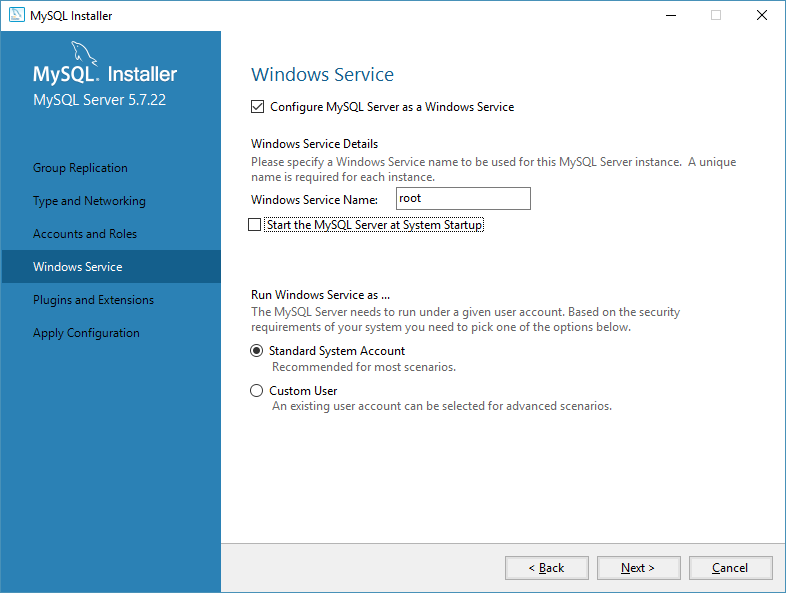


Рисунок 28 - Настройка сервисов

* + 1. Галочки оставляем по умолчанию и нажимаем кнопку «Next».

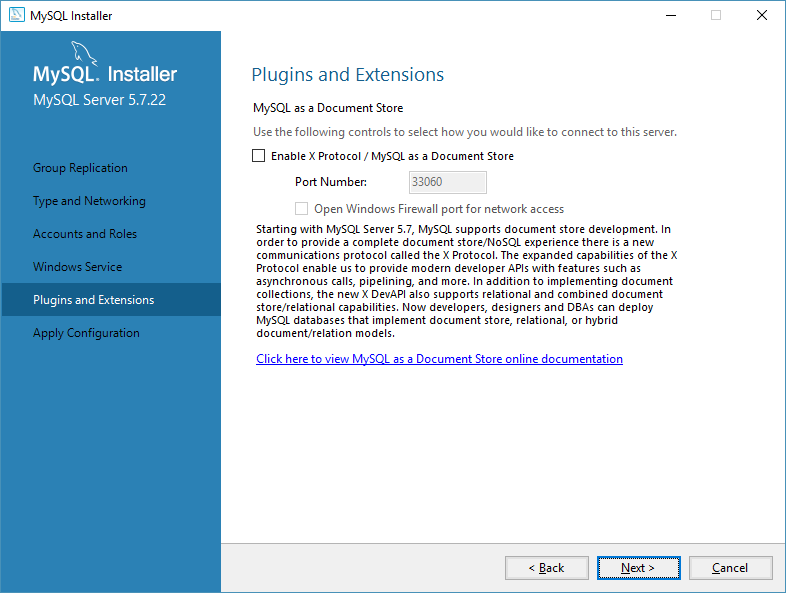


Рисунок 29 - Плагины

* + 1. Принимаем конфигурацию и нажимаем кнопку «Execute».

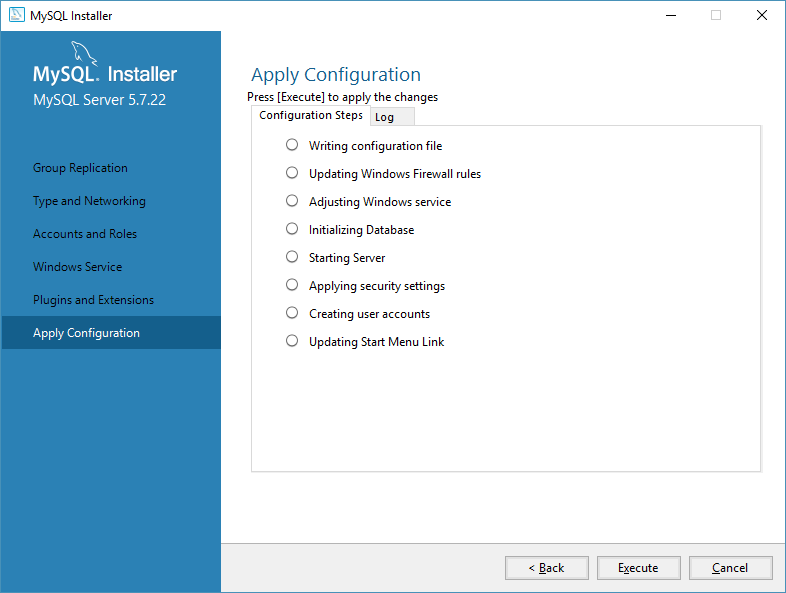


Рисунок 30 - Конфигурация

* + 1. Ожиданием завершения процесса установки.

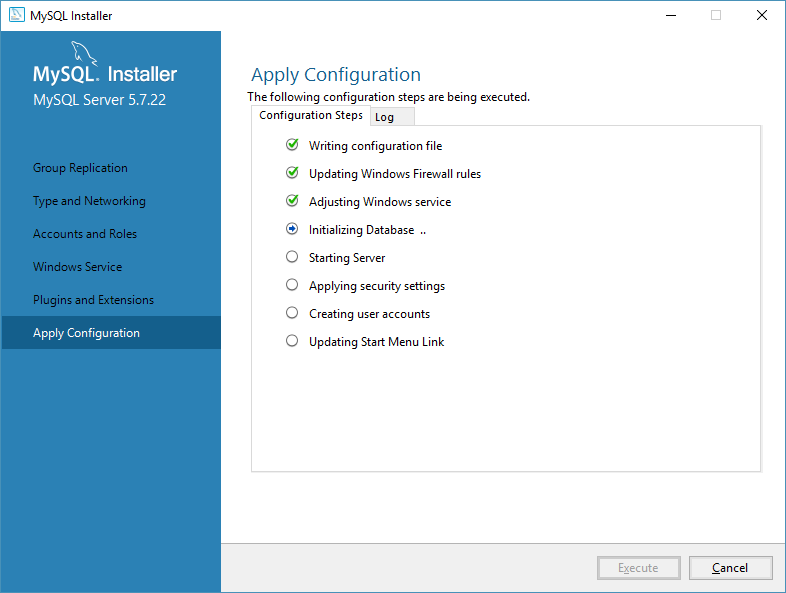


Рисунок 31 - Процесс установки

* + 1. После завершения процесса установки нажимаем кнопку «Finish»

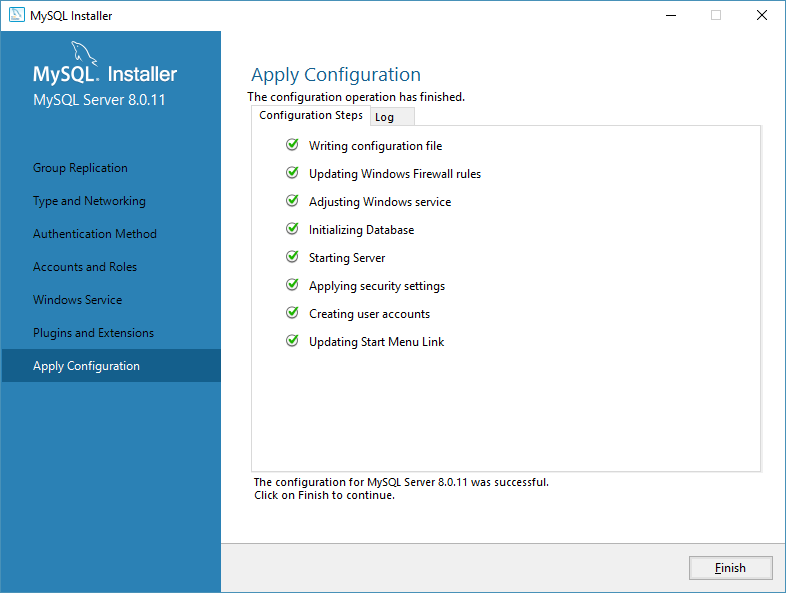


Рисунок 32 - Завершение процесса установки

## **Руководство пользователя**

1. Настройка программы.

Файл config.ini

Заголовок [CamSetting]

В параметр «ip» указывается адрес камеры.

В параметр «tesseract\_path» указывается путь к tesseract.exe.

В параметр «cascade» указывается путь к каскаду Хаара, содержащий примеры государственных номеров.

Заголовок [NetworkSQLSettings]

В параметр «ip» указывается адрес сервера базы данных.

В параметр «login» указывается логин пользователя базы данных.

В параметр «password» указывается пароль пользователя базы данных.

В параметр «charset» указывается кодировка базы данных.

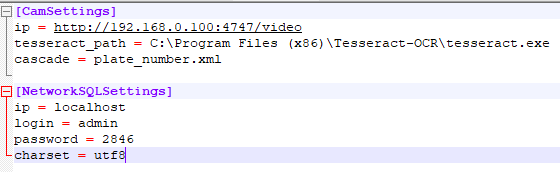


Рисунок 32 - config.ini

1. Работа с программой.

Для запуска программы запустите файл «UI.py». У вас появится несколько окон:

1. Основное окно программы. В нем:
   1. Адрес камеры.
   2. Статус подключения.
   3. Кнопка построение графика загруженности КПП, по часам.
   4. Поле вывода информации о владельце автомобиля из базы данных.

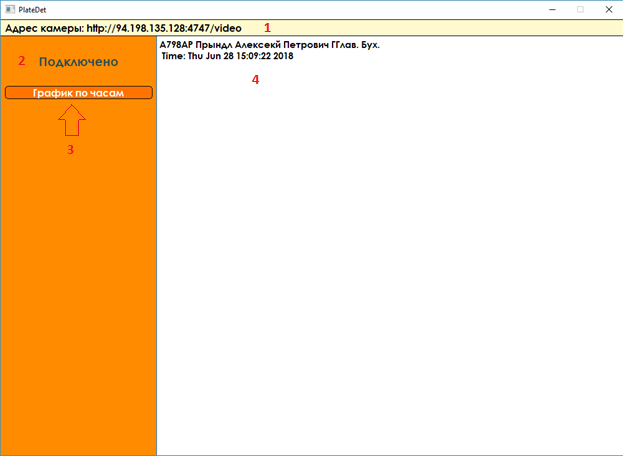


Рисунок 33 - Основное окно

1. Окно вывода изображения с камеры. В нем:
2. Выводится текущие изображение с камеры.

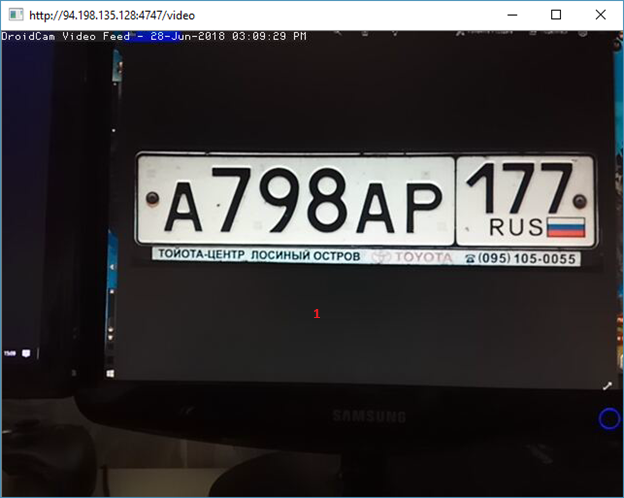


Рисунок 34 - Изображение с камеры

1. Окно с обнаруженным номером.



Рисунок 35 – Найденный номер

1. Окно с графиком, открывающиеся после нажатия кнопки в основном окне «График по часам».

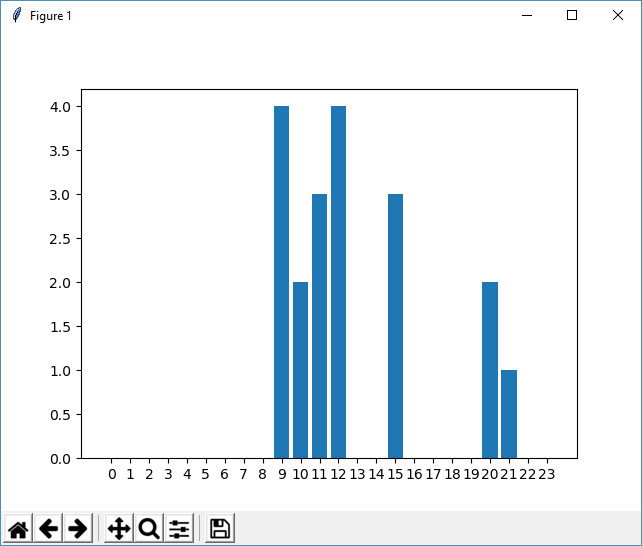


Рисунок 36 - График

## **Рекомендуемые минимальные системные требования.**

ОС: Windows 10 32/64 Bit, Windows 8.1 32/64 Bit, Windows 8 32/64 Bit, Windows 7 32/64 Bit.

Процессор: Intel Core i3 8100 @ 3.6 ГГц (четырехъядерный) / AMD FX-4300 @ 3.8 ГГц (четырехъядерный)

Оперативная память: 4 GB

Место на диске: 2 GB

Дополнительно:

Камера с разрешением не ниже HD(1280х720)

# **Заключение**

В заключение хочется сказать, что разработка данной программы принесла большое количество, как положительных, так и отрицательных эмоций. Достаточно хорошо укрепились знания работы с библиотеками: Opencv2, MySQLbd, Configparses, Matplotlib, Multithrading, PyQT5, так же закрепились знания MySQL.

Были рассмотрены:

* 1. Описание языка Python.
  2. Процессы.
  3. База данных MySQL.
  4. Работа с классами и функциями.
  5. Написание кода программы.
  6. Описание руководства пользователя.

# **Список используемой литературы**

* 1. http://forum.webmasterschool.ru/tmp/sql.pdf
  2. Николай Прохоренок, Владимир Дронов - Python 3 и PyQt 5. Разработка приложений.
  3. https://pythonworld.ru/gui/pyqt5-firstprograms.html
  4. http://qaru.site/questions/124067/very-basic-python-client-socket-example
  5. https://docs.python.org/2/library/multiprocessing.html
  6. http://python-3.ru/page/multiprocessing
  7. https://pythonspot.com/pyqt5/
  8. https://www.w3schools.com/sql/sql\_autoincrement.asp
  9. https://opencv.org/links.html
  10. https://matplotlib.org/tutorials/index.html
  11. https://docs.python.org/3/library/configparser.html
  12. https://python-scripts.com/configparser-python-example
  13. http://mysql-python.sourceforge.net/MySQLdb.html

# **Приложение**

## **Листинг 1**

"""---Интерфейс---"""

from PyQt5.QtCore import \*

from PyQt5.QtWidgets import QMainWindow, QWidget

from PyQt5.uic import loadUi

from PyQt5.QtWidgets import QApplication

"""------Обработка------"""

from pytesseract import image\_to\_string, pytesseract

import cv2

import re

"""---------------------"""

import configparser

import sys

import time

from multiprocessing import Process

"""---Файлы проекта---"""

import filter

import sql

import graph

class MainWindow(QMainWindow, QWidget):

def \_\_init\_\_(self):

super(MainWindow, self).\_\_init\_\_()

loadUi('main.ui', self)

self.star\_num = ''

self.setWindowTitle('PlateDet')

'''----Конфиг. файл открытие----'''

config = configparser.ConfigParser()

config.read("config.ini")

'''----Конфиг. файл чтение----'''

pytesseract.tesseract\_cmd = config.get("CamSettings", "tesseract\_path") # Путь к файлу tesseract.exe

self.img = cv2.VideoCapture(config.get("CamSettings", "IP")) # Адрес камеры

self.plate\_cascade = cv2.CascadeClassifier(config.get("CamSettings", "cascade")) # Путь к каскаду гос. номера

self.hourButton.clicked.connect(graph.GraphHours)

self.camera = config.get("CamSettings", "IP")

self.adress.setText(' Адрес камеры: ' + self.camera)

def ImgConv(self, image):

a = (255, 255, 255)

b = (110, 110, 110)

image = cv2.inRange(image, b, a)#фильтрация номера от фона

#поиск текста на изображении

text = image\_to\_string(image, lang='rus', config='--psm 6')

text = text.upper()

#

text2 = re.findall(r'[АВСЕНКМОРТХУ0123456789]', text)

text2 = ''.join(text2)

cv2.imshow('Detected number', image)

self.text, self.star\_num = filter.filter\_re(text2, self.star\_num)

if self.text != None:

print('SQL ->', self.text)

ADTime = time.asctime(time.localtime(time.time()))

self.textBrowser.append(self.text + '\n Time: ' + ADTime + '\n')

sql.zTime(self.text.split()[0], ADTime)

print(self.text.split()[0], ADTime)

else:

print('Number is not in the database!')

def camRead(self):

f = self.img.read()[1]

gray = cv2.cvtColor(f, cv2.COLOR\_BGR2GRAY)

plate = self.plate\_cascade.detectMultiScale(gray, 1.3, 5)

#обрезка изображения по размерам найденного номера

x, y, w, h = plate[0]

cv2.rectangle(f, (x, y), (x + w, y + h), (72, 255, 0), 2)

cropped = f[y:y + h, x:x + w]

final\_wide = 600

r = float(final\_wide) / cropped.shape[1]

dim = (final\_wide, int(cropped.shape[0] \* r))

resized = cv2.resize(cropped, dim, interpolation=cv2.INTER\_AREA)

#вывод обрезанного изображения

cv2.imshow(self.camera, f)

self.ImgConv(resized)

def status(self):

self.connect.setStyleSheet("color:#2F4F4F; background: rgba(0,0,0, .0); border:0px solid;")

self.connect.setText('Подключено')

if \_\_name\_\_ == '\_\_main\_\_':

app = QApplication(sys.argv)

MainWindow = MainWindow()

MainWindow.show()

def Run():

try:

while True:

try:

MainWindow.camRead()

MainWindow.status()

except:

f = MainWindow.img.read()[1]

cv2.imshow(MainWindow.camera, f)

MainWindow.status()

if cv2.waitKey(25) == 27:

break

except:

pass

prcRun = Process(target=Run())

prcRun.start()

prcRun.join()

sys.exit(app.exec\_())

*Исходник находится в папке/Курсовая работа/main.py*

## **Листинг 2**

import re

import sql

def filter\_re(text, star\_num):

for i in range(len(text) - 5):

if len(text) >= 6:

rt = text[i:i + 6]

if re.match(

r'["А","В","С","Е","Н","К","М","О","Р","Т","Х","У"]{1}[0-9]{3}["А","В","С","Е","Н","К","М","О","Р","Т","Х","У"]{2}',

rt):

if rt != star\_num:

star\_num = rt

return sql.zapros(rt), star\_num

*Исходник находится в папке/Курсовая работа/filter.py*

## **Листинг 3**

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>MainWindow</class>

<widget class="QMainWindow" name="MainWindow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>1000</width>

<height>700</height>

</rect>

</property>

<property name="minimumSize">

<size>

<width>1000</width>

<height>700</height>

</size>

</property>

<property name="maximumSize">

<size>

<width>1000</width>

<height>700</height>

</size>

</property>

<property name="windowTitle">

<string>MainWindow</string>

</property>

<widget class="QWidget" name="centralwidget">

<property name="styleSheet">

<string notr="true">QWidget {

background: #FF8C00;

}

QTextEdit {

background: white;

font: Bold 15px Century Gothic;

}

QLabel {

background: rgba(0,0,0, .4);

color: white;

font: Bold 20px Century Gothic;

border:1px solid;

border-radius: 10px;

}

QPushButton {

font: 12pt Century Gothic;

background: #FF7400;

color: white;

font-weight: 700;

text-decoration: none;

outline: none;

border: 1px solid;

border-radius: 5px;

}

QPushButton:hover {

color: rgb(224,63,35);

}

QPushButton:pressed {

background: rgba(254,239,182,.9);

}

QToolTip {

color: #FFFFFF; background-color: #808080; border: none;

}</string>

</property>

<widget class="QTextBrowser" name="textBrowser">

<property name="geometry">

<rect>

<x>250</x>

<y>26</y>

<width>751</width>

<height>675</height>

</rect>

</property>

<property name="html">

<string>&lt;!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML 4.0//EN&quot; &quot;http://www.w3.org/TR/REC-html40/strict.dtd&quot;&gt;

&lt;html&gt;&lt;head&gt;&lt;meta name=&quot;qrichtext&quot; content=&quot;1&quot; /&gt;&lt;style type=&quot;text/css&quot;&gt;

p, li { white-space: pre-wrap; }

&lt;/style&gt;&lt;/head&gt;&lt;body style=&quot; font-family:'Century Gothic'; font-size:15px; font-weight:600; font-style:normal;&quot;&gt;

&lt;p style=&quot;-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;&quot;&gt;&lt;br /&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>

</property>

</widget>

<widget class="QLabel" name="adress">

<property name="geometry">

<rect>

<x>-2</x>

<y>0</y>

<width>1003</width>

<height>27</height>

</rect>

</property>

<property name="styleSheet">

<string notr="true">QLabel {

background:#FFFACD;

color: Black;

font: Bold 16px Century Gothic;

border:1px solid;

border-radius: 0px;

}</string>

</property>

<property name="text">

<string/>

</property>

</widget>

<widget class="QWidget" name="">

<property name="geometry">

<rect>

<x>7</x>

<y>33</y>

<width>237</width>

<height>95</height>

</rect>

</property>

<layout class="QVBoxLayout" name="verticalLayout">

<item>

<widget class="QLabel" name="connect">

<property name="styleSheet">

<string notr="true">QLabel {

background: rgba(0,0,0, .0);

color: 2F4F4F;

font: Bold 16px Century Gothic;

border:0px solid;

border-radius: 10px;

}</string>

</property>

<property name="text">

<string>Нет подключения</string>

</property>

<property name="alignment">

<set>Qt::AlignCenter</set>

</property>

</widget>

</item>

<item>

<widget class="QPushButton" name="hourButton">

<property name="text">

<string>График по часам</string>

</property>

</widget>

</item>

</layout>

</widget>

</widget>

</widget>

<resources/>

<connections/>

</ui>

*Исходник находится в папке/Курсовая работа/main.ui*