

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Московский институт электроники и математики

Платонов Илья Сергеевич, группа БИВ192
Кусакин Илья Константинович, группа БИВ192

СИСТЕМА КОНТРОЛЯ ПЕШЕХОДНОГО ПЕРЕХОДА

Курсовая работа
по направлению 09.03.01 Информатика и вычислительная техника
студента образовательной программы бакалавриата
«Информатика и вычислительная техника»

Студент _____ И.С. Платонов
Студент _____ И.К. Кусакин

Руководитель
И.И. Романова

Москва 2019 г.

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Московский институт электроники и математики им. Тихонова

ЗАДАНИЕ
на курсовую работу бакалавра

студентам группы БИВ 192 Платонову Илье Сергеевичу, Кусакину Илье
Константиновичу

1. Тема работы

Разработка комплекса контроля пешеходного перехода на базе Raspberry
Pi

2. Требования к работе

2.1. Общие требования

Система контроля перехода, взаимодействующая с видеокамерой,
должна быть реализована на одноплатном компьютере Raspberry
Pi

2.2. Требования к прототипу

К процессу разработки, а также к итоговой реализации прототипа
системы предъявляются следующие требования:

- Обеспечить программную совместимость разработки с ОС
одноплатного компьютера
- Реализовать сценарии использования разрабатываемой системы
на прототипе
- Разработать пользовательский графический интерфейс (GUI) для
демонстрации прототипа

2.3. Требования к программно-аппаратному обеспечению

Построение прототипа осуществляется на базе одноплатного
компьютера с ядром ARM v8 на базе ARM Cortex-A архитектуры

3. Содержание работы

3.1. Анализ существующих технических решений в области видеоконтроля

3.2. Реализация аппаратно-программного обеспечения

3.3. Аprobация системы контроля пешеходного перехода на прототипе

4. Сроки выполнения этапов работы

Проект курсовой работы представляется студентом в срок до «15» февраля 2020 года

Первый вариант КР предоставляется студентом в срок до «1» апреля 2020 года

Итоговый вариант КР предоставляется студентом руководителю до загрузки работы в систему «антиплагиат» в срок до «31» мая 2020 года

Аннотация

Объект разработки в данном проекте – аппаратно-программный по контролю пешеходного перехода. Цель работы – создать систему, которая будет способна отслеживать нарушения на пешеходном переходе, распознавать нарушителей, обновлять имеющиеся данные. В результате выполнения проекта был создан комплекс, состоящий из одноплатного компьютера с камерой и программной части, реализующий контроль нарушений.

Объем данной курсовой работы без учета приложений составляет N страницы. Работа включает 4 иллюстрации, 13 источников.

Оглавление

Аннотация	4
Введение.....	6
1. Обзор аналогов	8
1.1. Введение.....	8
1.2. Описание и анализ аналогов	8
1.3. Анализ выбранных аналогов.....	10
1.4. Выводы	10
2. Описание решений поставленных задач.....	11
2.1. Введение.....	11
2.2. Описание и обоснование решений	11
2.3. Выводы	12
3. Создание комплекса	13
3.1. Введение.....	13
3.2. Пошаговое выполнение сборки	13
3.3. Выводы	15
4. Руководство пользователя.....	16
Заключение	21
Список источников	23
Приложения	24
Приложение 1 - <i>launcher.py</i>	24
Приложение 2 – <i>main.py</i>	32
Приложение 3 – <i>ex_students.xlsx</i>	36

Введение

Цели и задачи работы:

Целью данной курсовой работы является создание аппаратно-программного комплекса по контролю нарушений на пешеходном переходе.

Для достижения цели необходимо выполнить такие задачи:

1. Познакомиться с устройством и принципом работы компьютера Raspberry Pi.
2. Определить, какие электронные компоненты необходимы для создания комплекса.
3. Научиться работать с Raspberry Pi, а также разобраться с подключением дополнительных модулей, необходимых для разработки.
4. Изучить библиотеки компьютерного зрения, необходимые для разработки комплекса. Ознакомиться с принципами их работы, получить знания по реализации технологий машинного обучения.
5. Написать программу для работы устройства.
6. Создать графический интерфейс для взаимодействия с пользователем.

В современном мире на дорогах присутствует большое количество транспортных средств, которые постоянно соседствуют с пешеходами. Данное обстоятельство является источником повышенной опасности на пешеходных переходах, и как следствие предметом внимания со стороны надзорных органов, например, ГИБДД. Особое внимание приковано к проезжей части возле образовательных учреждений: школ, детских садов и университетов. Молодым людям свойственно неосторожное поведение на автомобильных дорогах и тротуарах, находящихся с рядом с проезжей частью. В связи с этим, администрации учебных заведений заинтересованы в контролировании ситуации на дорогах, прилежащих к учебным корпусам. Для того, чтобы отслеживать дорожную конъюнктуру, в частности вести статистику перехода проезжей части учащимися и, как следствие, иметь возможность предпринимать превентивные меры по

воспитанию культуры поведения на дороге, можно обеспечить каждого учащегося навигационным датчиком.

В данном проекте использовался одноплатный компьютер Raspberry Pi, стандартный модуль для видеосъемки Pi Camera, библиотека компьютерного зрения OpenCV. В качестве языка программирования был выбран Python 3, средой для программирования был выбран Jupiter Notebook.

В результате создан комплекс, способный хранить в себе информацию об учащихся, фиксировать нарушения правил дорожного движения на пешеходном переходе и распознавать нарушителя, в случае если о нем присутствует информация.

1. Обзор аналогов

1.1. Введение

Один из самых важных этапов разработки, без которых невозможно выполнение проекта - определение используемых для реализации аппаратных компонентов, фреймворков и общей концепции комплекса.

1.2. Описание и анализ аналогов

Система, использующая устройство с функцией геолокации позволяет синхронизировать передвижения учащегося с циклами светофора, установленного на пешеходном переходе. Для этого необходимо обеспечить школьника или студента устройством, оборудованным датчиком, позволяющим определять его местоположение в режиме реального времени, а соответственно устройство также должно иметь возможность подключения к сети интернет с целью передачи данных о местоположении. В качестве системы определения местоположения предполагается использование российской навигационной спутниковой системы ГЛОНАСС. В качестве такого устройства может выступать индивидуальный смартфон, оснащенный навигацией, смарт-часы или браслет, оснащенные датчиком местоположения и передающие информацию по протоколу Bluetooth.

Смартфон является сложным техническим устройством с высокой себестоимостью. Помимо функции отслеживания перемещения, в том числе контроля поведения на пешеходных переходах, смартфон обладает многими другими функциями, составляющих высокую итоговую цену смартфона. В то же время наличие этих функций не является обязательным требованием со стороны предполагаемого интересанта реализуемого проекта, например администрации учебного заведения, потому переплата за неиспользуемые и необязательные функции не несет в себе прикладного смысла.

Смарт-часы по схожей причине, как и смартфон, обладают избыточными аппаратными возможностями, реализация которых в рамках проекта не является

необходимостью. Потому стоимость обеспечения учащихся смарт-часами велика и необоснованна с экономическо-технической точки зрения.

Браслет с датчиком, позволяющим иметь данные о местоположении в режиме реального времени, обладает лучшим соотношением количества используемых технических возможностей к себестоимости в сравнении с аналогами в виде смартфона и смарт-часов. Однако стоит принимать во внимание то, что количество учащихся очень велико, а главный интерес со стороны администрации возлагает ответственность за материальное обеспечение построения системы контроля на проезжей части на бюджет учебного заведения, что является проблемой в условиях недостатка денежных средств для реализации внеучебных проектов.

Общей проблемой для метода контроля перехода пешеходного перехода путем обеспечения учащихся навигационными датчиками, является этический вопрос обладания конфиденциальными данными о местоположении частных лиц. Прилежащие к учебному корпусу автомобильные дороги, пешеходный переход выступают частным предметом контроля, в то время как датчики местоположения способны передавать гораздо больший объем данных, что идет в разрез с законодательными актами РФ. Потому система контролирования путем обеспечения учащихся датчиками с системой ГЛОНАСС является системой сложной не только с технической точки зрения, но и с юридической.

1.3. Анализ выбранных аналогов

Выбранный метод контролирования ситуации на пространстве пешеходного перехода – установление на переходе комплекса, состоящего из камеры, подключенной к вычислительному устройству, например одноплатному компьютеру Raspberry Pi. Такой комплекс «железа» вкуче с программным обеспечением на основе компьютерного зрения позволяет производить идентификацию учащихся без индивидуальных устройств, распознавать циклы светофора без синхронизации с внутренней системой, регулирующей пешеходный переход, вести статистику по нарушениям порядка на проезжей части для учеников или студентов. Язык программирования Python был выбран в силу своей простоты и скорости реализации на нем готового к использованию программного обеспечения. Среда Jupiter Notebook является удобным инструментом для пошагового тестирования готовых компонентов программы, а вместе со сборкой Anaconda потенциал среды значительно расширяется за счет предустановленных пакетов по созданию необходимого программного обеспечения. Библиотека OpenCV была выбрана в силу качественной и проработанной документации, большого количества вводной литературы, а также популярности реализованных с ее помощью проектов.

1.4. Выводы

На основе анализа аналогов были выявлены преимущества выбранной концепции и компонентов для ее реализации. Была подготовлена техническая база для реализации программных и аппаратных частей проекта.

2. Описание решений поставленных задач

2.1. Введение

Для выполнения курсовой работы были поставлены несколько задач, основные из которых освоить подключение дополнительных модулей к Raspberry Pi, применить в программной части технологии компьютерного зрения, реализованные при помощи OpenCV, разработать концепцию хранения информации об учащихся и работе с ней, связать между собой графический интерфейс и фактический функционал комплекса.

2.2. Описание и обоснование решений

Для ознакомления с принципами работы с Raspberry Pi были изучены реализованные проекты с участием компьютера в качестве центрального вычислительного устройства, в том числе проекты по домашнему видеонаблюдению и концепциям умного дома.

Ознакомление с библиотекой компьютерного зрения OpenCV было решено начать с чтения документации, которая постоянно обновляется разработчиками. Необходимый для работы функционал был подробно изучен в нескольких разделах документации, содержащей подробную информацию.

В ходе выполнения курсовой работы изначальная концепция хранения данных об учащихся, основанная на пакете SQLite была изменена. Выбор был сделан в пользу документов типа `xlsx`. Такая концепция позволяет без применения дополнительных пакетов по работе с базами данных быстро реализовать удобную систему хранения данных о студентах. Не требуется подключение дополнительных пакетов для синхронизации с Python, широта применения файлов `xlsx` обеспечивает различные сценария их использования, а возможность редактирования неподготовленным пользователем при помощи стандартных офисных программ выгодно противопоставляет их традиционным базам данных.

Графический интерфейс программной части проекта был написан при помощи библиотеки Tkinter для языка Python. В отличие от популярного аналога – библиотеки

Pyqt5, синхронизированной со множеством сред разработки, например QtCreator, Tkinter позволяет осуществлять более тонкую настройку объектов, больший уровень персонализации в разработке. Для успешной реализации графического интерфейса была изучена объектно-ориентированная парадигма программирования, изучено ее применение в Python/Tkinter.

Полученные знания и навыки были применены на практике при реализации проекта курсовой работы.

2.3. Выводы

Для решения поставленных задач была изучена литература, интернет-ресурсы, проекты, реализованные на выбранных для курсовой работы компонентах, практически применены полученные знания, продуман ход работы над проектом, доработана изначальная концепция и программная база. Выбранные решения позволили выполнить задачи достаточно быстро, хорошо освоиться в теме курсовой работы.

3. Создание комплекса

3.1. Введение

После того, как были определены функции программно-аппаратного комплекса, выбраны компоненты, участвующие в разработке, получены необходимые для их использования знания и навыки, можно приступить к реализации проекта на практике.

При создании комплекса были выделены такие этапы:

1. Установка всех необходимых для работы библиотек и сред программирования. В том числе пакетов OpenCV, Tkinter, Pandas, Numpy.
2. Создание базового программного функционала комплекса в связке с аппаратной часть.
3. Создание системы данных о студентах.
4. Разработка графического интерфейса для пользователя.
5. Объединение между собой основообразующего программного функционала, системы данных об учащихся и графического интерфейса

3.2. Пошаговое выполнение сборки

Библиотека компьютерного зрения была скачана с официального сайта разработчиков, остальные пакеты были установлены через стандартный установщик Python.

Основа программной части комплекса, задача которой состояла в реализации системы распознавания лиц была написана с использованием библиотеки OpenCV. При помощи стандартных функций пакета, а также возможности через Python управлять потоковым видео с веб-камеры или Pi Camera были написаны три функции. Задача первой состояла в формировании датасета из фотографий предполагаемых учеников для дальнейшего обучения нейросети во второй функции. Функция номер 3

осуществляла распознавание лиц людей на потоковом видео с камеры и вывод информации о них, если обнаруженный человек входил в список предполагаемых студентов. В дальнейшем все три функции были объединены в отдельный модуль для его удобного импортирования, код которого представлен в Приложении 1.

При помощи Microsoft Excel был создан типовый проект файла, который показан в Приложении 3, для хранения в нем информации об обучающихся, в дальнейшем файл `ex_students.xlsx` был дополнен полями для хранения информации о настройках комплекса.

Графический интерфейс, код которого представлен в Приложении 2. разрабатывался с учетом наличия функций добавления, редактирования и удаления информации о студентах. Также в графический интерфейс был добавлен раздел с настройками комплекса, которые может самостоятельно задавать пользователь. В итоге было создано приложение, которое представлено для пользователя несколькими окнами, между которыми он может переключаться для реализации функционала комплекса. В рамках разработки графического интерфейса были освоены навыки объектно-ориентированного программирования.

Последним шагом проектирования комплекса стало объединение в единую программно-аппаратную систему двух модулей, один из которых отвечает за основной функционал комплекса (обнаружение и распознавание), а второй за графический интерфейс, а также системы данных об обучающихся. Была проделана работа по введению внутрисистемного `id` каждого обучающегося, для единого типа обращения и передачи информации по данному ключу. Были синхронизированы между собой датасет из фотографий, файл `ex_students.xlsx`, прописаны дополнительные функции в модуле `launcher.py` для взаимодействия с функциями из модуля `main`. В модуле `main` в свою очередь были реализованы элементы для обращения к директориям проекта, независимо от устройства, на котором используется комплекс. Были разработаны элементы комплекса по проверке соответствия данных в файле `ex_students.xlsx` и датасете фотографий. Реализован сборщик мусора при наличии ошибок (несовпадений в хранящейся информации).

Также программные модули были дополнены функциями для настройки комплекса под различные типы пешеходных переходов.

Итоговый код программной части проекта, а также вид файла для хранения данных показаны в Приложении 1, Приложении 2, Приложении 3.

3.3. Выводы

В процессе создания программно-аппаратного комплекса были применены на практике все те знания, которые были получены во время учебного процесса, найдены в литературе, документациях к программным и аппаратным компонентам. При выполнении проекта возникали трудности, связанные с новизной используемых ресурсов, отвергались старые концепции, происходил переход к новым методам работы. Тщательный анализ собственных ошибок и упорная работа позволили создать комплекс, полностью соответствующий техническому заданию и готовый к работе.

4. Руководство пользователя

Для начала работы с программой необходимо скачать архив «Kurs_1» по [ссылке](#). Далее распакуйте скачанный архив на карту памяти типа SD. Поместите карту памяти в Raspberry Pi, предварительно подключив Pi Camera. Запустите на одноплатном компьютере командную строку. Перейдите в директорию «Kurs_1» на карте памяти, затем в директорию «bin» и запустите файл «launcher.py». Перед вами появится стартовое окно графического интерфейса комплекса.

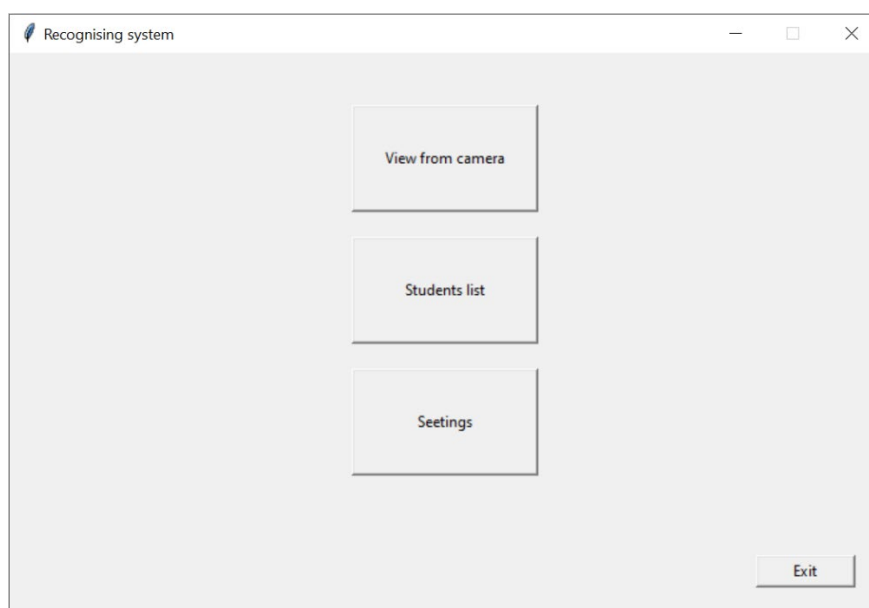


Рис.1 Стартовое окно программы

В правом нижнем углу стартового окна находится кнопка «Exit». При ее нажатии вы совершите выход из программы.

По центру окна находятся три большие кнопки. Верхняя кнопка «View from camera» запустит слежение за ситуацией на пешеходном переходе. Откроется новое окно, в котором отобразится потоковое видео с камеры наблюдения, начнется фиксация нарушений. При условии распознавания нарушителя, информация о переходе на запрещенный сигнал светофора занесется в таблицу данных.

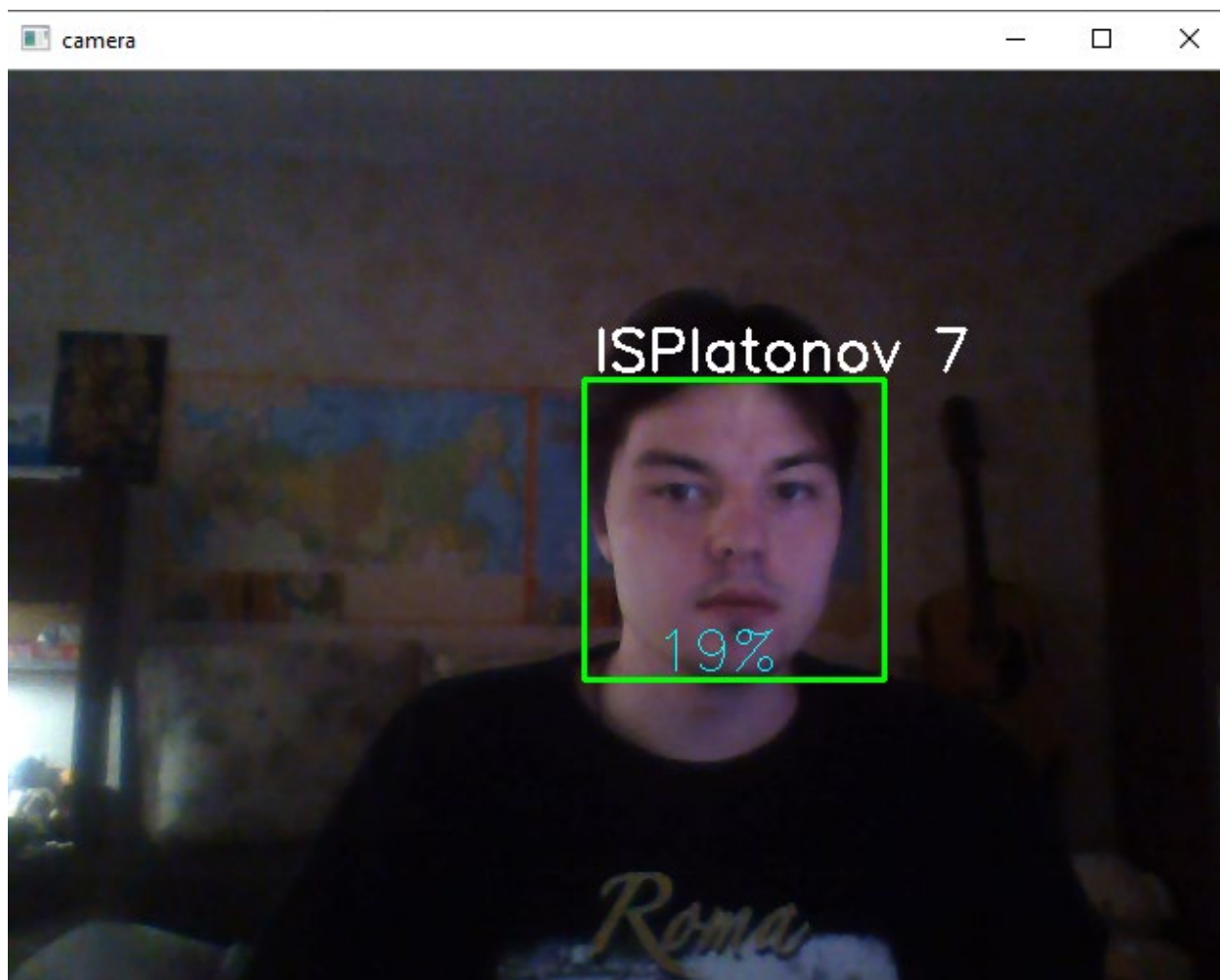


Рис.2 Потокное видео с камеры

При нажатии кнопки «Students list» на месте стартового окна появится окно, отображающее информацию о студентах, занесенных в таблицу данных.

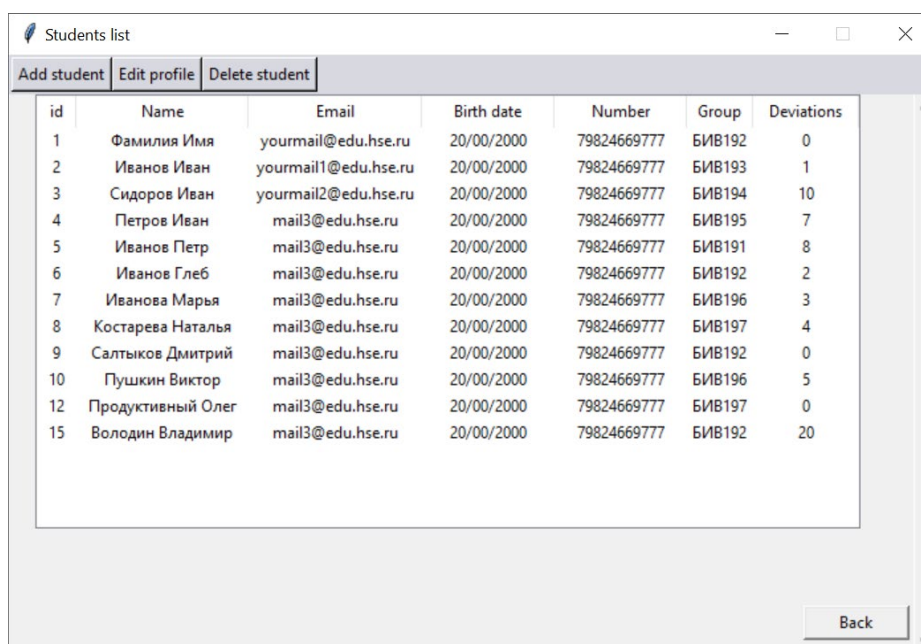


Рис.3 Окно со списком студентов

Кнопка «Back» в правом нижнем углу окна со списком студентов вернет Вас на стартовое окно программы.

В средней части окна представлен список студентов, занесенных в таблицу данных. О каждом из студентов есть информация из перечня: внутрисистемный id, имя и фамилия, email, дата рождения, номер телефона, номер группы, число нарушений.

Сверху от списка студентов находится ряд из трех кнопок, отвечающих за модерацию списка студентов.

При нажатии кнопки «Add student» появится окно, с полями ввода информации о новом студенте.

The 'Add student' window features a form with the following fields and controls:

- id:** 19
- Name:**
- Email:**
- Birth date:**
- Number:**
- Group:** BIB192 (dropdown menu)
- Deviations:**
- Buttons:** Add, Back

Рис. 4 Окно добавления нового студента

Введите информацию о студенте. После нажатия кнопки «Add» студент, чьи данные добавляются, должен в течение 10 секунд посмотреть в веб-камеру компьютера. Нажмите «Back» чтобы вернуться к обновившемуся списку студентов.

Для редактирования данных о студенте нажмите кнопку «Edit profile». Откроется следующее окно:

The 'Edit profile' window features a search bar and a form with the following fields and controls:

- Search:** Search by Name (dropdown), (input), Find (button)
- id:** Unknown
- Name:**
- Email:**
- Birth date:**
- Number:**
- Group:** BIB192 (dropdown menu)
- Deviations:**
- Buttons:** Edit, Back

Рис.5 Окно редактирования данных

Воспользуйтесь поиском студента по ФИО/email/id. После нажатия кнопки «Find» при наличии студента, которого вы ищете в полях ввода отобразится текущая информация о студенте.

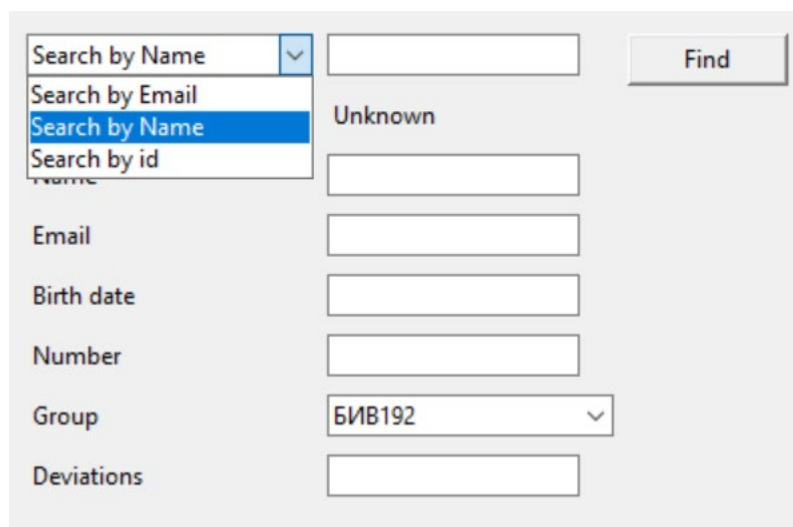


Рис.6 Блок поиска студента и поля ввода

Отредактируйте необходимые данные и нажмите на кнопку «Edit», чтобы обновить данные. Нажмите на кнопку «Back», чтобы вернуться к окну с обновленным списком студентов.

Для того, чтобы удалить студента из списка, выделите необходимую строку нажатием левой кнопки мыши. Далее нажмите кнопку «Delete student», студент пропадет из списка.

Кнопка «Settings» на стартовом окне программы откроет окно с настройками комплекса под фазы светофора и положение пешеходного перехода. Задайте необходимое количество секунд для запрещающей и разрешающей фаз светофора, а также сдвиг по фазам, в случае если происходит отставание от графика пешеходного перехода. Также доступен выбор доли экрана с отчетом от нижней части окна, в котором будет происходить распознавание нарушений. По умолчанию задано значение в 50%, что будет соответствовать нижней половине окна, в которой будет возможно распознавание нарушителей.

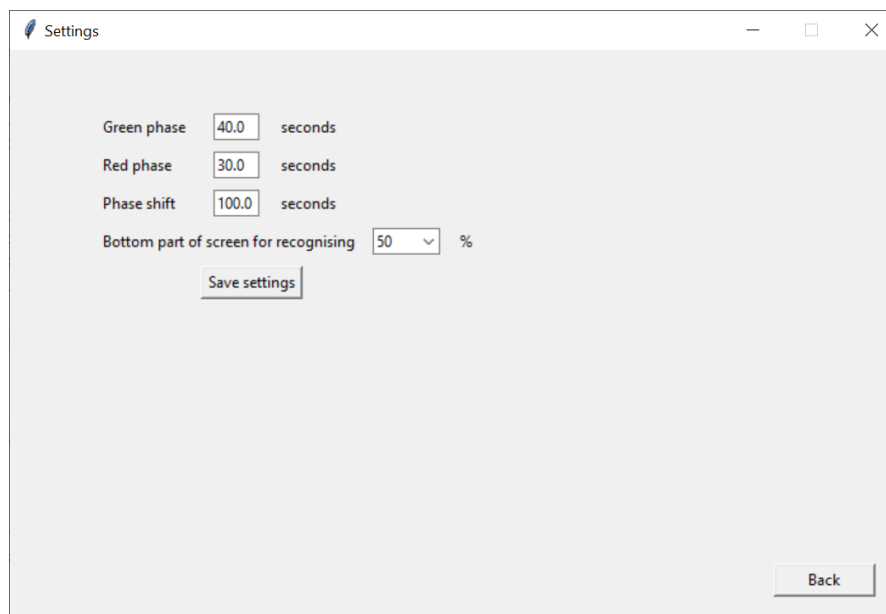


Рис. 7 Окно с настройками комплекса

Нажатие кнопки «Save settings» сохранит внесенные изменения. Вернуться к стартовому окну можно, воспользовавшись кнопкой «Back».

Заключение

В результате применения полученных знаний и умений был создан программно-аппаратный комплекс, отвечающий всем требованиям технического задания, способный работать многих платформах и операционных системах за счет свойств совместимости, изначально выбранной элементной базы. Дополнения в python-модулях, а также разработанная структура директорий позволяют запустить комплекс как на одноплатном компьютере Raspberry Pi, так и на персональном компьютере с веб-камерой. Также стоит отметить важность полученного опыта по созданию программно-аппаратного комплекса. В рамках курсовой работы были обретыны навыки работы с неизвестными ранее компонентами, расширены умения

работы с документацией библиотек и пакетов, выучен новый язык программирования и успешно использован в разработке.

Список источников

1. [OpenCV-Python Cheat Sheet: From Importing Images to Face Detection](#)
2. [Wrapper package for OpenCV python bindings](#)
3. [ОБНАРУЖЕНИЕ И РАСПОЗНАВАНИЕ ЛИЦА НА PYTHON](#)
4. [Распознаем лица на фото с помощью Python и OpenCV](#)
5. [Установка OpenCV-Python на виртуальной среде](#)
6. [Устанавливаем python-пакеты с помощью pip](#)
7. [Установка PIP для Python на Windows, Mac и Linux](#)
8. [ООП на Python](#)

Приложения

Приложение 1 - *launcher.py*

```
from tkinter import *
from tkinter import ttk
import sqlite3
import cv2
import os
import numpy as np
from PIL import Image
import main
import pandas as pd

root = Tk()

class Main(Frame):
    def __init__(self, master):
        super().__init__(master)
        self.proove()
        self.root_frame = Frame(master)
        self.root_frame.pack(expand = 1)
        self.btn_view = Button(self.root_frame, text = 'View from camera', width = 20, height = 7, command =
self.open_view)
        self.btn_view.pack(expand = 1, pady = 10)

        self.btn_settings = Button(self.root_frame, text = 'Seetings', width = 20, height = 5, command =
self.open_settings)
        self.btn_settings.pack(expand = 1, pady = 10)

        self.btn_students = Button(self.root_frame, text = 'Students list', width = 20, height = 7, command =
self.open_students)
        self.btn_students.pack(expand = 1, pady = 10)

        self.btn_exitroot = Button(master, text = 'Exit', width = 10, command = self.exitroot)
        self.btn_exitroot.pack(side = BOTTOM, anchor = SE, pady = 20, padx = 20)

    def open_students(self):
        Students()

    def exitroot(self):
        root.destroy()

    def open_settings(self):
        Settings()

    def open_view(self):
        main.cam_2()
#проверка корректности датасета
    def proove(self):
        import pandas as pd
        ex_list_id = [str(i) for i in pd.read_excel('ex_students.xlsx')['id']]
```



```

path_ds = os.getcwd() + "/../dataset"
list_photos = os.listdir(path_ds)
#print(list_photos)
for name in list_photos:
    if name.split('.')[1] not in ex_list_id:
        os.remove(path_ds + '/' + name)
main.trainer_0()

class Settings(Toplevel):
    def __init__(self):
        super().__init__()
        data = pd.read_excel('ex_students.xlsx')
        self.title('Settings')
        self.geometry('700x450+{}+{}'.format(w,h))
        self.resizable(False,False)
        self.grab_set()
        self.focus_set()

        self.label_green = Label(self, text='Green phase')
        self.label_green.place(x = 70, y = 50)
        self.entry_green = ttk.Entry(self, width=5)
        self.entry_green.place(x=160, y=50)
        self.label_green_sec = Label(self, text='seconds')
        self.label_green_sec.place(x = 210, y = 50)

        self.label_red = Label(self, text='Red phase')
        self.label_red.place(x = 70, y = 80)
        self.entry_red = ttk.Entry(self, width=5)
        self.entry_red.place(x=160, y=80)
        self.label_red_sec = Label(self, text='seconds')
        self.label_red_sec.place(x = 210, y = 80)

        self.label_sdvig = Label(self, text='Phase shift')
        self.entry_sdvig = ttk.Entry(self,width=5)
        self.label_sdvig.place(x=70,y=110)
        self.entry_sdvig.place(x=160,y=110)
        self.label_sdvig_sec = Label(self, text='seconds')
        self.label_sdvig_sec.place(x = 210, y = 110)

        self.label_part = Label(self,text='Bottom part of screen for recognising')
        self.label_part.place(x=70,y=140)
        self.combobox_part = ttk.Combobox(self, values = ['20', '30','40', '50','60', '70','80'],width=5)
        for j in range(len(self.combobox_part['values'])):
            if str(int(data['Part'][0])) == self.combobox_part['values'][j]:
                self.combobox_part.current(j)
        self.combobox_part.place(x=285,y=140)
        self.label_percent = Label(self,text='%')
        self.label_percent.place(x=350,y=140)

        self.entry_sdvig.insert(0,data['Sdvig'][0])
        self.entry_green.insert(0,data['Green'][0])
        self.entry_red.insert(0,data['Red'][0])

```

```

self.btn_save_set = Button(self, text='Save settings', width=10, command=self.save_settings)
self.btn_save_set.place(x=150, y=170)

self.btn_backfromset = Button(self, text = 'Back', width = 10, command = self.backfromset)
self.btn_backfromset.pack(side = BOTTOM, anchor = SE, pady = 20, padx = 20)

def save_settings(self):
    data = pd.read_excel('ex_students.xlsx')
    data['Green'][0] = self.entry_green.get()
    data['Red'][0] = self.entry_red.get()
    data['Sdvig'][0] = self.entry_sdvig.get()
    data['Part'][0] = self.combobox_part.get()
    data.to_excel('ex_students.xlsx', index = False)
    main.update_settings()

def backfromset(self):
    self.destroy()

#окно с таблицей студентов
class Students(Toplevel):
    def __init__(self):
        super().__init__()
        self.init_main()
        self.view_records()

    def init_main(self):
        self.title('Students list')
        self.geometry('700x450+{}+{}'.format(w,h))
        self.resizable(False,False)
        self.grab_set()
        self.focus_set()
        #тулбар с функциональными кнопками
        self.toolbar = Frame(self, bg='#d7d8e0', bd='2')
        self.toolbar.pack(side = TOP, fill=X)

        btn_opendialog = Button(self.toolbar, text='Add student', bg='#d7d8e0',
bd='2', command=self.open_dialog, compound = TOP)
        btn_opendialog.pack(side = LEFT)

        btn_edit_dialog = Button(self.toolbar, text='Edit profile', bg='#d7d8e0', bd='2', command =
self.open_updatedialog, compound = TOP)
        btn_edit_dialog.pack(side = LEFT)

        btn_delete_info = Button(self.toolbar, text='Delete student', bg='#d7d8e0', bd='2', command
=self.delete_info, compound = TOP)
        btn_delete_info.pack(side = LEFT)

        self.tree = ttk.Treeview(self, column=('id','name','email','date','number', 'groups', 'devs'),
height = 15, show = 'headings')
        #описываем колонки таблички
        self.tree.column('id', width=30, anchor=CENTER)
        self.tree.column('name', width=130, anchor=CENTER)
        self.tree.column('email', width=130, anchor=CENTER)

```

```

self.tree.column('date',width=100,anchor=CENTER)
self.tree.column('number',width=100,anchor=CENTER)
self.tree.column('groups',width=50,anchor=CENTER)
self.tree.column('devs',width=80,anchor=CENTER)

#даем колонкам заголовки
self.tree.heading('id',text='id')
self.tree.heading('name',text='Name')
self.tree.heading('email',text='Email')
self.tree.heading('date',text='Birth date')
self.tree.heading('number',text='Number')
self.tree.heading('groups',text='Group')
self.tree.heading('devs',text='Deviations')

self.tree.pack(side= LEFT, padx=20, anchor = N)

scroll = Scrollbar(self,command=self.tree.yview)
scroll.pack(side=RIGHT,fill=Y)
self.tree.config(yscrollcommand=scroll.set)

self.btn_backroot = Button(self, text = 'Back', width =10, command = self.backtoroot)
self.btn_backroot.place(x = 600, y = 415)

def backtoroot(self):
    self.destroy()

#функция отображения информации из эксель в виджете tree, удаляем старые данные, и
заполняем новыми, импортируя их из бд
def view_records(self):
    [self.tree.delete(i) for i in self.tree.get_children()]
    data = pd.read_excel('ex_students.xlsx')
    data_matrix = []
    local=[]
    for i in range(len(data['Name'])):
        local = []
        for j in data.columns:
            local.append(data[j][i])
        data_matrix.append(local)
    [self.tree.insert('','end', values = row ) for row in data_matrix]

#функция удаления строки в экселе
def delete_info(self):
    data = pd.read_excel('ex_students.xlsx')
    treeid = self.tree.selection()[0] #внутренний id в treeview
    for i in range(len(data['Name'])):
        if data['Name'][i] == self.tree.item(treeid, option="values")[1]: #получаем значение из treeview
            data = data.drop(i)
    data.to_excel('ex_students.xlsx', index = False)
    #удаление фотографий
    del_id = self.tree.item(treeid, option="values")[0]
    #print(del_id)
    path_ds = os.getcwd() + '/../dataset"
    list_photos = os.listdir(path_ds)

```

```

#print(list_photos)
for name in list_photos:
    if name.split('.')[1] == str(del_id):
        os.remove(path_ds + '/' + name)
#отображаем изменения и заново обучаем
self.view_records()
main.trainer_0()

#вызываем дочернее окно
def open_dialog(self):
    Students.destroy(self)
    AddStudent()

#вызываем окно редактирования
def open_updatedialog(self):
    self.destroy()
    Update()

class AddStudent(Toplevel):
    def __init__(self):
        super().__init__()
        self.title('Add student')
        self.geometry('700x450+{}+{}'.format(w,h))
        self.resizable(False,False)

        #подписи к виджетам
        self.label_id = Label(self, text='id')
        self.label_id.place(x = 50, y = 50)
        self.label_name = Label(self, text='Name')
        self.label_name.place(x = 50, y = 80)
        self.label_email = Label(self, text='Email')
        self.label_email.place(x = 50, y = 110)
        self.label_date = Label(self, text='Birth date')
        self.label_date.place(x = 50, y = 140)
        self.label_number = Label(self, text='Number')
        self.label_number.place(x = 50, y = 170)
        self.label_groups = Label(self, text='Group')
        self.label_groups.place(x = 50, y = 200)
        self.label_devs = Label(self, text='Deviations')
        self.label_devs.place(x = 50, y = 230)
        self.label_add = Label(self, text = 'Please look at the camera after adding student until ending Capture
programm')
        self.label_add.place(x = 50, y = 20)

        #виджеты ввода
        data = pd.read_excel('ex_students.xlsx')
        #подставляем исходный id
        self.label_id_entry = Label(self, text=int(data['id'][len(data['id'])-1]+1))
        self.label_id_entry.place(x = 200, y = 50)

        self.entry_name = ttk.Entry(self)
        self.entry_name.place(x=200, y=80)

```

```

self.entry_email = ttk.Entry(self)
self.entry_email.place(x=200, y=110)

self.entry_date = ttk.Entry(self)
self.entry_date.place(x=200, y=140)

self.entry_number = ttk.Entry(self)
self.entry_number.place(x=200, y=170)

self.combobox = ttk.Combobox(self, values = ['БИВ191', 'БИВ192','БИВ193', 'БИВ194','БИВ195',
'БИВ196','БИВ197'])
self.combobox.current(1)
self.combobox.place(x=200,y=200)

self.entry_devs = ttk.Entry(self)
self.entry_devs.place(x=200, y=230)

#кнопка закрытия дочернего окна
self.btn_cancel = ttk.Button(self,text='Back')
self.btn_cancel.bind('<Button-1>', lambda event: self.vspom())
self.btn_cancel.place(x = 300, y = 280)

#кнопка добавления информации введенной через виджеты
self.btn_ok = ttk.Button(self, text='Add')
self.btn_ok.place(x = 200, y = 280)
self.btn_ok.bind('<Button-1>', lambda event: self.records())
self.grab_set()
self.focus_set()
def vspom(self):
    self.destroy()
    Students()
#запись нового студента
def records(self):
    data = pd.read_excel('ex_students.xlsx')
    data = data.append({'id':int(data['id'][len(data['id'])-1]+1),'Name':self.entry_name.get(),
'Email':self.entry_email.get(), 'Birth Date':self.entry_date.get(),'Number':self.entry_number.get(),
'Group':self.combobox.get(), 'Deviations':self.entry_devs.get()}, ignore_index=True)
    data.to_excel('ex_students.xlsx', index = False)
    print(data['id'][len(data['id']) - 1]) #id добавляемого
    main.cam_1(data['id'][len(data['id']) -1])

class Update(AddStudent):
    def __init__(self):
        super().__init__()
        self.init_edit()
        self.label_add.destroy()

    def init_edit(self):
        #блок из комбобокса ентри и кнопки для поиска студентов
        self.combobox_search = ttk.Combobox(self, values = ['Search by Email', 'Search by Name','Search by
id'])
        self.combobox_search.current(1)

```

```

self.combobox_search.place(x=50,y=50)

self.entry_search = ttk.Entry(self)
self.entry_search.place(x=200, y=50)

self.btn_find = Button(self, text='Find', command = self.find, width = 10)
self.btn_find.place(x = 350 , y = 50)

self.label_id.place(x = 50, y = 80)
self.label_name.place(x = 50, y = 110)
self.label_email.place(x = 50, y = 140)
self.label_date.place(x = 50, y = 170)
self.label_number.place(x = 50, y = 200)
self.label_groups.place(x = 50, y = 230)
self.label_devs.place(x = 50, y = 260)
#вставка/редактирование информации о найденном студенте
self.label_id_entry.place(x = 200, y = 80)
self.entry_name.place(x = 200, y = 110)
self.entry_email.place(x = 200, y = 140)
self.entry_date.place(x = 200, y = 170)
self.entry_number.place(x = 200, y = 200)
self.combobox.place(x = 200, y = 230)
self.entry_devs.place(x = 200, y = 260)

self.title('Edit profile')
#кнопка редактирования данных
btn_edit = ttk.Button(self, text = 'Edit', width = 10)
btn_edit.place(x = 200, y = 310)
btn_edit.bind('<Button-1>', lambda event: self.update_records())
self.btn_ok.destroy()
self.btn_cancel.place(x = 300, y = 310)

self.label_id_entry = Label(self, text = 'Unknown')
self.label_id_entry.place(x = 200, y = 80)

#self.label_find = Label(self, text='Search student by email')
#self.label_find.place(x = 350, y = 80)

#находим по id/имени/почте студента и вставляем его данные в entry_
def find(self):
    #верхний поисковой блок студента
    if self.combobox_search.get() == 'Search by id':
        self.search_string = 'id'
    if self.combobox_search.get() == 'Search by Name':
        self.search_string = 'Name'
    if self.combobox_search.get() == 'Search by Email':
        self.search_string = 'Email'

    #удаляем старые данные
    self.entry_name.delete(0, 'end')
    self.entry_email.delete(0, 'end')
    self.entry_date.delete(0, 'end')

```

```

self.entry_number.delete(0, 'end')
self.entry_devs.delete(0, 'end')

#вставляем данные найденного студента
data = pd.read_excel('ex_students.xlsx')
for i in range(len(data[self.search_string])):
    if str(data[self.search_string][i]) == self.entry_search.get():
        self.label_id_entry['text'] = data['id'][i]
        self.entry_name.insert(0,data['Name'][i])
        self.entry_email.insert(0, data['Email'][i])
        self.entry_date.insert(0,data['Birth Date'][i])
        self.entry_number.insert(0,data['Number'][i])
        self.entry_devs.insert(0,data['Deviations'][i])
        for j in range(len(self.combobox['values'])):
            if data['Group'][i] == self.combobox['values'][j]:
                self.combobox.current(j)

```

```

#запись в эксель релактированной информации
def update_records(self):
    data = pd.read_excel('ex_students.xlsx')
    for i in range(len(data['Name'])):
        if str(data['id'][i]) == self.label_id_entry['text']:
            data['Name'][i] = self.entry_name.get()
            data['Email'][i] = self.entry_email.get()
            data['Birth Date'][i] = self.entry_date.get()
            data['Number'][i] = self.entry_number.get()
            data['Group'][i] = self.combobox.get()
            data['Deviations'][i] = self.entry_devs.get()
            data.to_excel('ex_students.xlsx', index = False)

```

```

w = root.winfo_screenwidth() // 2 - 325
h = root.winfo_screenheight() // 2 - 225
appmain = Main(root)
appmain.pack()
root.title('Recognising system')
root.geometry('700x450+{}+{}'.format(w,h))
root.resizable(False, False)
root.mainloop()
'''records и update_records прописаны в классах дочерних окон и вызываются из этих же классов.
когда закрываются дочерние окна, заново открывается окно Students, при открытии которого
выполняется
view_records показывающие в таблице текущие данные в экселе.
окно Students закрывается при нажатии на Add student или Edit'''

```

Приложение 2 – main.py

```
def update_settings():
    import pandas as pd
    data = pd.read_excel('ex_students.xlsx')
    greentime = int(data['Green'][0])
    redtime = int(data['Red'][0])
    time_s = int(data['Sdvig'][0])
    part_screen = 480 - int(data['Part'][0]*0.01*480)
    return greentime, redtime, time_s, part_screen

def cam_1(face_id):
    import os
    import cv2

    cam = cv2.VideoCapture(0)
    cam.set(3, 640) # установка ширины видео
    cam.set(4, 480) # установка высоты видео

    path = os.getcwd() + '/../' # директория приложения

    # Детектор лиц в библиотеке opencv
    face_detector = cv2.CascadeClassifier(path +
    '/libraries/opencv/build/etc/haarcascades/haarcascade_frontalface_default.xml')

    print("Initializing face capture. Look the camera and wait ...")
    # Инициализация переменной для подсчёта количества фото
    count = 0

    while(True):
        ret, img = cam.read()
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        faces = face_detector.detectMultiScale(gray, 1.3, 5)

        for (x, y, w, h) in faces:
            cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)
            count += 1

            # Сохранение изображения лица в базу данных
            cv2.imwrite(path + "/dataset/User." + str(face_id) + '.' + str(count) + ".jpg", gray[y:y+h,x:x+w])

        cv2.imshow('image', img)

        k = cv2.waitKey(100) & 0xff # 'ESC' для остановки
        if k == 27:
            break
        elif count >= 30: # остановка записи при сделанных 30 фото
            break

    # Заккрытие окон записи
    print("Exiting Capure program and cleanup stuff")
    cam.release()
    cv2.destroyAllWindows()
```



```

trainer_0() # Перезапись файла распознавания

def cam_2():
    import cv2
    import numpy as np
    import os
    import time
    import pandas as pd

    path = os.getcwd() + '/../' # директория приложения

    names = UpdateNames() # Загрузка имён из списка студентов Excel
    recognizer = cv2.face.LBPHFaceRecognizer_create()
    recognizer.read(path + '/trainer/trainer.yml')
    cascadePath = path + '/libraries/opencv/build/etc/haarcascades/haarcascade_frontalface_default.xml'
    faceCascade = cv2.CascadeClassifier(cascadePath);

    font = cv2.FONT_HERSHEY_SIMPLEX

    id = 0 # Объявление переменной id

    # Начало видеозахвата
    cam = cv2.VideoCapture(0)
    cam.set(3, 640) # set video width
    cam.set(4, 480) # set video height

    # Минимальный размер окна, который может быть распознан как лицо
    minW = 0.1*cam.get(3)
    minH = 0.1*cam.get(4)

    # Время горения
    greentime, redtime, time_s, part_screen = update_settings()

    #greentime = 20 # зелёный свет
    #redtime = 10 # красный свет
    #time_s = 1583769600 # Время начала (18:00 09.03.2020) в секундах
    itrn = 0 # Количество итераций за цикл
    timedict = {} # Список нарушений (количество итераций в зоне) за период горения красного

    while True:
        ret, img = cam.read()
        if (time.time() - time_s) % (greentime + redtime) < redtime:
            #print(time.time())
            itrn += 1

            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

            faces = faceCascade.detectMultiScale(gray, scaleFactor = 1.2, minNeighbors = 5, minSize =
(int(minW), int(minH)),)

            for(x,y,w,h) in faces:

```

```

# Координаты центра лица
center_x = x + w / 2 # X
center_y = y + h / 2 # Y

# Запись при расположении лица ниже середины экрана (дописать)
if center_y > part_screen:
    cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2)
    id, confidence = recognizer.predict(gray[y:y+h,x:x+w])

# Проверка уверенности (меньше 100 - опознан, 0 - совершенная уверенность)
if (confidence < 100):
    name = names[id]
    confidence = " {0}%".format(round(100 - confidence))
    if id in timedict:
        timedict[id] += 1
    else:
        timedict[id] = 1
else:
    name = "unknown"
    confidence = " {0}%".format(round(100 - confidence))

cv2.putText(img, str(name) + ' ' + str(id), (x+5,y-5), font, 1, (255,255,255), 2)
cv2.putText(img, str(confidence), (x+5,y+h-5), font, 1, (255,255,0), 1)

else:
    # Время горения красного света вышло,
    # запускается цикл записи нарушивших в БД
    if itrn > 30: # Нижний порог для запуска записи (30)
        data = pd.read_excel('ex_students.xlsx')
        #print(itrn)
        porog = itrn * 3 // redtime # Нижний порог для записи нарушения (около 3 секунд в зоне)
        for key in timedict:
            if timedict[key] >= porog:
                for i in range(len(data['id'])):
                    if data['id'][i]==key:
                        data['Deviations'][i] = int(data['Deviations'][i])+1
                        #print(key) # id нарушавшего
                        break

        data.to_excel('ex_students.xlsx', index = False) # Запись нарушений в базу данных
    #Сброс счётчиков
    if itrn != 0:
        itrn = 0
    timedict = {}

cv2.imshow('camera', img)

k = cv2.waitKey(10) & 0xff # 'ESC' для остановки
if k == 27:
    break

# Закрытие окон
#print("Exiting Program and cleanup stuff")

```

```

cam.release()
cv2.destroyAllWindows()

def trainer_0():
    import cv2
    import numpy as np
    from PIL import Image
    import os

    path = os.getcwd() + '/../' # Директория приложения
    path_ds = path + '/dataset' # Путь к базе изображений лиц

    recognizer = cv2.face.LBPHFaceRecognizer_create()
    detector = cv2.CascadeClassifier(path +
    + '/libraries/opencv/build/etc/haarcascades/haarcascade_frontalface_default.xml');

    # Функция записи лица с созданием ярлыка
    def getImagesAndLabels(path_ds):
        imagePath = [os.path.join(path_ds, f) for f in os.listdir(path_ds)] # Изображения лиц
        faceSamples = []
        ids = []
        for imagePath in imagePath:
            PIL_img = Image.open(imagePath).convert('L') # Конвертирование изобр. в чёрно-белое
            img_numpy = np.array(PIL_img, 'uint8')
            #print(os.path.splitext(imagePath)[-1].split(".")[1])
            face_id = int(os.path.splitext(imagePath)[-1].split(".")[1])
            faces = detector.detectMultiScale(img_numpy)
            for (x,y,w,h) in faces:
                faceSamples.append(img_numpy[y:y+h,x:x+w])
                ids.append(face_id)
            #print(ids)
        return faceSamples, ids

    print ("Training faces. It will take a few seconds. Wait ...")
    faces,ids = getImagesAndLabels(path_ds)
    recognizer.train(faces, np.array(ids))

    # Сохранение лица в trainer/trainer.yml
    recognizer.write(path + '/trainer/trainer.yml') # recognizer.save() worked on Mac, but not on Pi

    # Вывод числа записанных лиц
    print("{0} faces trained. Exiting Program".format(len(np.unique(ids))))

def UpdateNames():
    import pandas as pd
    data = pd.read_excel('ex_students.xlsx')
    #print(data)

    names = {} # names[id] = smbds_name
    for i in range(len(data['id'])):
        names[data['id'][i]] = data['Name'][i]
    #print(names)
    return names

```

Приложение 3 – ex_students.xlsx

id	Name	Email	Birth Date	Number	Group	Deviations	Green	Red	Sdvig	Part
0	Example	example@	00.00.0000	898200000	БИВ191	0	1	5	0	50
4	Angelina	12	rg	hsw	БИВ192	0				
5	Ikkusakin	adfs	baa	badf	БИВ192	2				
6	Timifey	kiyg	kj	,jbkjb	БИВ192	0				
7	ISPlatono	ghmnbm	bnm	vbnmu	БИВ192	15				