###### МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

###### ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

###### НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

###### Факультет информационных технологий

**Кафедра параллельных вычислений**

ОТЧЕТ

О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

«ВЫСОКОУРОВНЕВАЯ РАБОТА С ПЕРИФЕРИЙНЫМИ УСТРОЙСТВАМИ»

студента 2 курса, 23208 группы

**Веретенникова Андрея Александровича**

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

Д.С. Иванишкин

Новосибирск 2024

**СОДЕРЖАНИЕ**

[ЦЕЛЬ 3](#_Toc18443921)

[ЗАДАНИЕ 3](#_Toc18443922)

[ОПИСАНИЕ РАБОТЫ 4](#_Toc18443923)

[ЗАКЛЮЧЕНИЕ 5](#_Toc18443924)

[Приложение 1 (2, …). *Наименование приложения* 6](#_Toc18443925)

# ЦЕЛЬ

*Ознакомиться с программированием периферийных устройств на примере ввода данных с Web-камеры с использованием библиотеки OpenCV.*

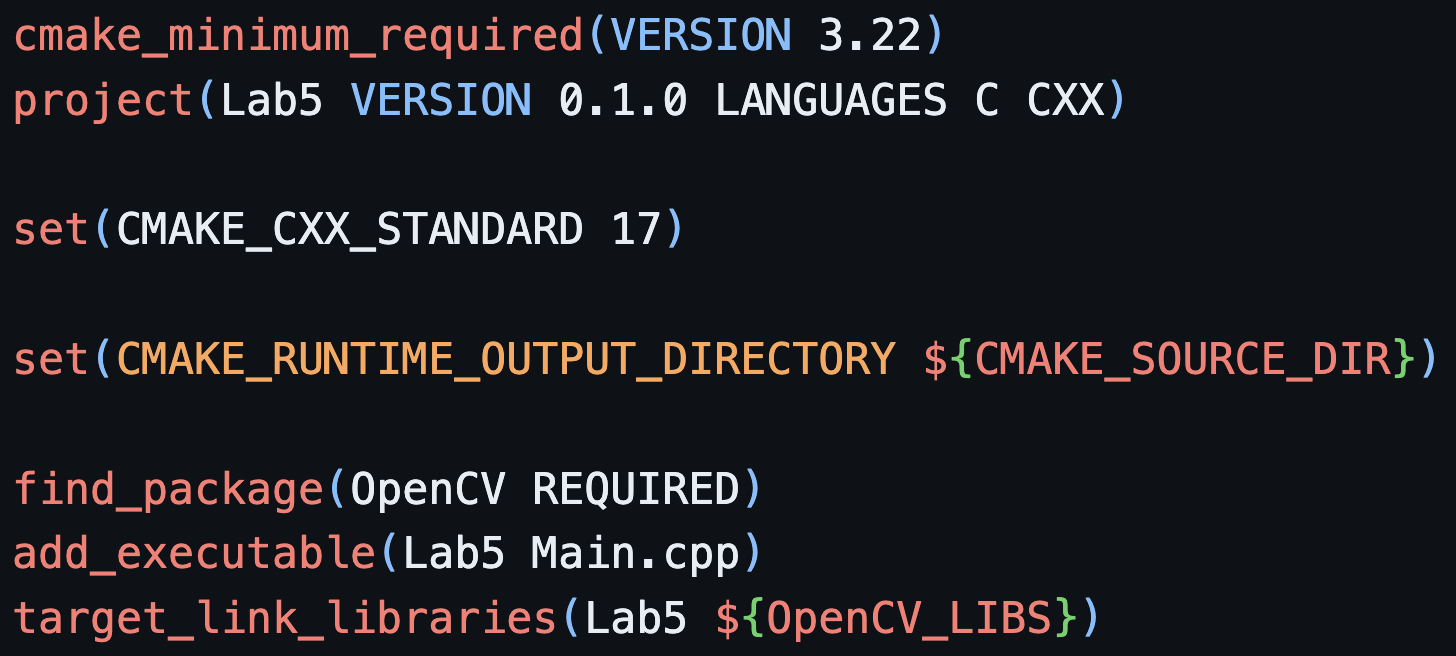
# ОПИСАНИЕ РАБОТЫ

*Пошаговое описание выполненной работы:*

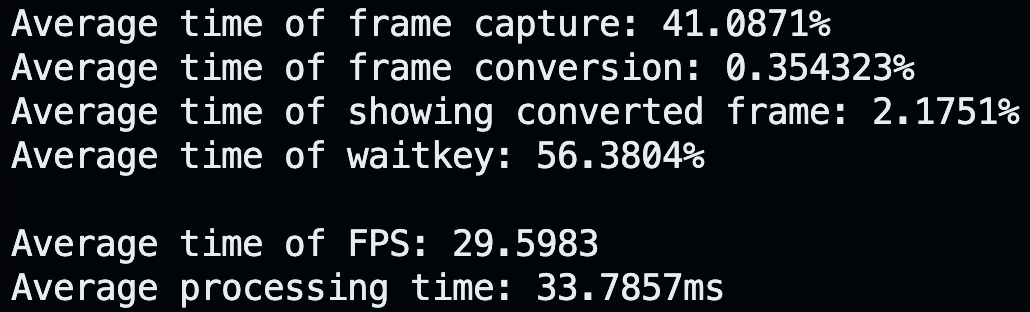
1. *Реализовать программу с использованием OpenCV, которая получает поток видеоданных с камеры и выводит его на экран.*
2. *Выполнить произвольное преобразование изображения.*
3. *Измерить количество кадров, обрабатываемое программой в секунду.*
   1. *Оценить долю времени, затрачиваемого процессором на обработку (ввод, преобразование, показ) видеоданных, получаемых с камеры.*

*Выбранная методика: библиотека OpenCV, преобразование — оттенки серого.*

*Строки CMakeLists.txt для сборки проекта:*

**

*Результаты:*

**

**

# ЗАКЛЮЧЕНИЕ

*В ходе лабораторной работы я ознакомился с программированием периферийных устройств на примере ввода данных с Web-камеры с использованием OpenCV.*

# Приложение 1. *Листинг*

#include <opencv2/opencv.hpp>

#include <iostream>

#include <chrono>

using namespace std;

using namespace chrono;

int main() {

cv::VideoCapture capture(0);

if (!capture.isOpened()) {

cerr << "Ошибка: не удалось открыть камеру!" << endl;

return -1;

}

cv::Mat frame, transformedFrame;

int frame\_count = 0;

int FPS\_count = 0;

double sum\_percentage\_time\_frame\_capture = 0;

double sum\_percentage\_time\_frame\_conversion = 0;

double sum\_percentage\_showing\_converted\_frame = 0;

double sum\_percentage\_waitkey = 0;

auto FPS\_timer = high\_resolution\_clock::now();

auto start = high\_resolution\_clock::now();

while (true) {

auto start\_frame\_capture = high\_resolution\_clock::now();

capture.read(frame);

auto end\_frame\_capture = high\_resolution\_clock::now();

duration<double> time\_frame\_capture = end\_frame\_capture - start\_frame\_capture;

if (frame.empty()) break;

auto start\_frame\_conversion = high\_resolution\_clock::now();

cv::cvtColor(frame, transformedFrame, cv::COLOR\_BGR2GRAY);

auto end\_frame\_conversion = high\_resolution\_clock::now();

duration<double> time\_frame\_conversion = end\_frame\_conversion - start\_frame\_conversion;

auto start\_showing\_converted\_frame = high\_resolution\_clock::now();

cv::imshow("Camera", transformedFrame);

auto end\_showing\_converted\_frame = high\_resolution\_clock::now();

duration<double> time\_showing\_converted\_frame = end\_showing\_converted\_frame - start\_showing\_converted\_frame;

auto start\_waitkey = high\_resolution\_clock::now();

if (cv::waitKey(1) == 'q') break;

auto end\_waitkey = high\_resolution\_clock::now();

duration<double> time\_waitkey = end\_waitkey - start\_waitkey;

auto end\_frame\_processing = high\_resolution\_clock::now();

duration<double> time\_frame\_processing = end\_frame\_processing - start\_frame\_capture;

sum\_percentage\_time\_frame\_capture += (time\_frame\_capture / time\_frame\_processing) \* 100;

sum\_percentage\_time\_frame\_conversion += (time\_frame\_conversion / time\_frame\_processing) \* 100;

sum\_percentage\_showing\_converted\_frame += (time\_showing\_converted\_frame / time\_frame\_processing) \* 100;

sum\_percentage\_waitkey += (time\_waitkey / time\_frame\_processing) \* 100;

frame\_count++;

FPS\_count++;

auto current\_time = high\_resolution\_clock::now();

duration<double> time\_since\_last\_FPS = current\_time - FPS\_timer;

}

auto end = high\_resolution\_clock::now();

duration<double> total\_time = end - start;

double average\_FPS = frame\_count / total\_time.count();

double average\_processing\_time = (total\_time.count() / frame\_count) \* 1000.0;

cout << "Average time of frame capture: " << sum\_percentage\_time\_frame\_capture / frame\_count << "%" << endl <<

"Average time of frame conversion: " << sum\_percentage\_time\_frame\_conversion / frame\_count << "%" << endl <<

"Average time of showing converted frame: " << sum\_percentage\_showing\_converted\_frame / frame\_count << "%" << endl <<

"Average time of waitkey: " << sum\_percentage\_waitkey / frame\_count << "%\n" << endl;

cout << "Average time of FPS: " << average\_FPS << endl;

cout << "Average processing time: " << average\_processing\_time << "ms" << endl;

capture.release();

cv::destroyAllWindows();

return 0;

}