**ЛАБОРАТОРНАЯ РАБОТА №5.**

**ВЫСОКОУРОВНЕВАЯ РАБОТА С ПЕРИФЕРИЙНЫМИ**

**УСТРОЙСТВАМИ**

**Цель:** ознакомиться с программированием периферийных устройств на примере ввода данных с Web-камеры с использованием библиотеки OpenCV

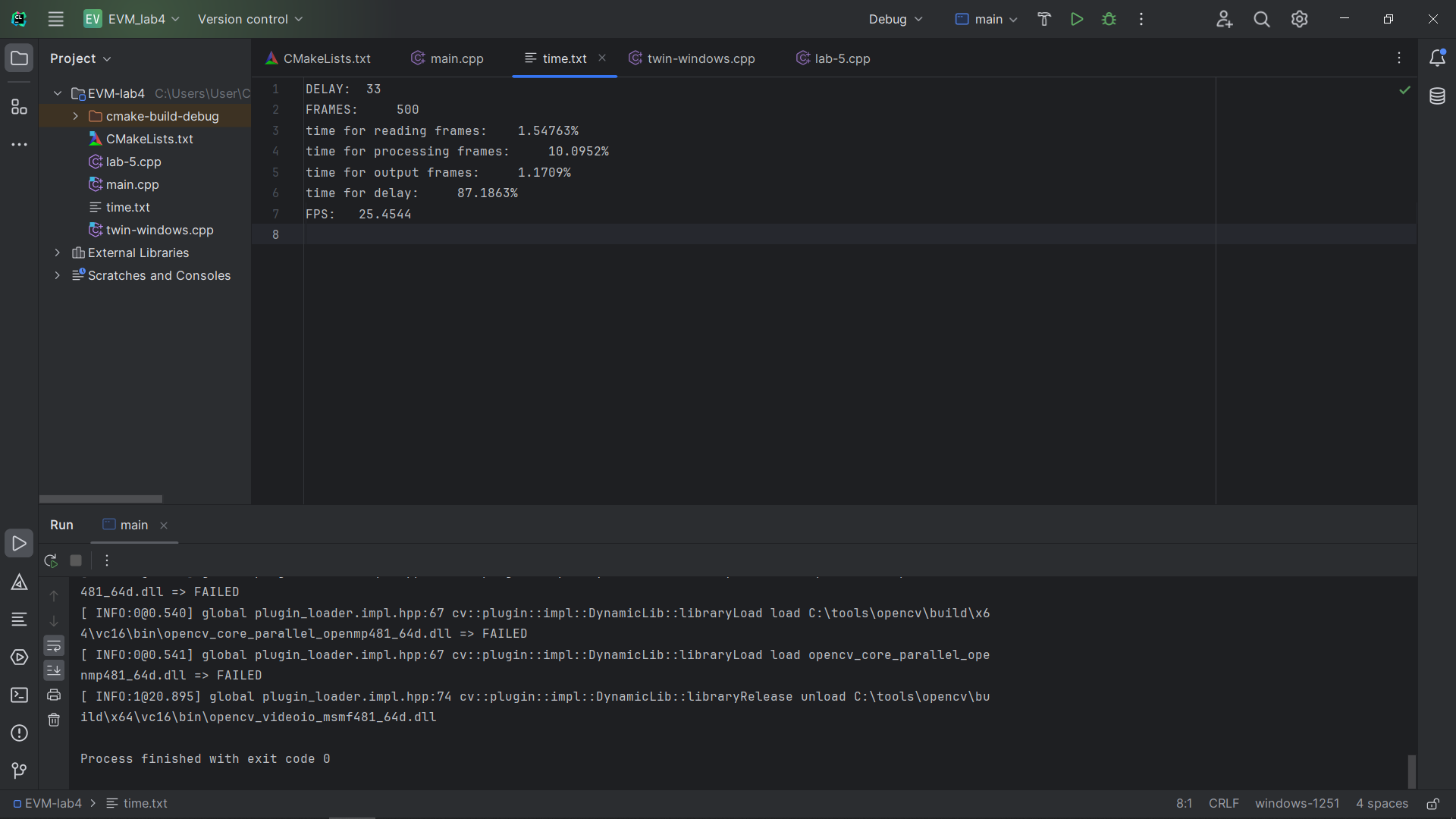
**Задачи:**

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. Оценить долю времени, затрачиваемого процессором на обработку (ввод, преобразование, показ) видеоданных, получаемых с камеры.
4. Составить отчет по лабораторной работе.

**Описание работы**

В процессе выполнения лабораторной работы были написаны 2 программы на языке C++, первая из которых выводит на экран в 2 окна изображение с веб-камеры: в одном окне открывается оригинальная картинка, получаемая с камеры, а в другом – обработанная (переведенная в черно-белые цвета). Вторая программа включает в себя так же подсчет времени и частоты кадров на конкретном количестве кадров, полученных камерой. Для замеров времени использовался системный таймер clock(), подсчитывающих количество тактов процессора. Для подсчета частоты кадров в секунду такты процессора переведены в секунды.

После проведения замеров были получены следующие результаты:



Из данных результатов видно, что при выставленной задержке следующего кадра в 33 мс основная доля времени уходит именно на ожидание кадра, следующий по затратности времени процесс – обработка картинки, на которую уходит около 10%. Реальная частота кадров при такой задержке оказалась равная примерно 25 кадрам в секунду.

**Выводы:** в процессе выполнения лабораторной работы я ознакомился с программированием периферийных устройств на примере ввода данных с Web-камеры с использованием библиотеки OpenCV, а также измерил долю времени на разные части воспроизведения картинки и частоту кадров

**Приложение 1: Листинг программы для отображения двух окон**

1. #include <opencv2/opencv.hpp>
2. #include <iostream>
4. using namespace std;
5. **using namespace cv;**

8. int main() {
9. cv::VideoCapture capOrig(0);
10. **cv::VideoCapture capProc(0);**
11. if (!capOrig.isOpened()) {
12. cout << "ÐÐµ ÑƒÐ´Ð°Ð»Ð¾ÑÑŒ Ð¾Ñ‚ÐºÑ€Ñ‹Ñ‚ÑŒ ÐºÐ°Ð¼ÐµÑ€Ñƒ" << endl;
13. return -1;
14. }
15. **if (!capProc.isOpened()) {**
16. cout << "ÐÐµ ÑƒÐ´Ð°Ð»Ð¾ÑÑŒ Ð¾Ñ‚ÐºÑ€Ñ‹Ñ‚ÑŒ ÐºÐ°Ð¼ÐµÑ€Ñƒ" << endl;
17. return -1;
18. }
19. cv::namedWindow("Video Stream Orig", cv::WINDOW\_NORMAL);
20. **cv::namedWindow("Video Stream Proc", cv::WINDOW\_NORMAL);**
21. while (true) {
22. cv::Mat frameOrig;
23. cv::Mat frameProc;
24. capOrig.read(frameOrig);
25. **capProc.read(frameProc);**
26. if (!frameOrig.empty()) {
27. cv::imshow("Video Stream Orig", frameOrig);
28. }
29. cv::cvtColor(frameOrig, frameProc, cv::COLOR\_BGR2GRAY);
30. **if (!frameProc.empty()) {**
31. cv::imshow("Video Stream Proc", frameProc);
32. }
33. if (cv::waitKey(33) == 'q') {
34. break;
35. **}**
36. }
37. capOrig.release();
38. capProc.release();
39. cv::destroyAllWindows();
40. **return 0;**
41. }

Приложение 2: листинг программы для подсчета времени

1. #include <opencv2/opencv.hpp>
2. #include <iostream>
3. #include <fstream>
5. **#define DELAY 33**
6. #define FRAMES 500
8. using namespace std;
9. using namespace cv;

12. int main() {
13. unsigned long long frameCounter = 0;
14. cv::VideoCapture capOrig(0);
15. **if (!capOrig.isOpened()) {**
16. cout << "ÐÐµ ÑƒÐ´Ð°Ð»Ð¾ÑÑŒ Ð¾Ñ‚ÐºÑ€Ñ‹Ñ‚ÑŒ ÐºÐ°Ð¼ÐµÑ€Ñƒ" << endl;
17. return -1;
18. }
19. cv::namedWindow("Video Stream Proc", cv::WINDOW\_NORMAL);
20. **long double totalTime = 0;**
21. long double inputTime = 0;
22. long double procTime = 0;
23. long double outputTime = 0;
24. long double delayTime = 0;
25. **unsigned long long start1 = 0, end1 = 0;**
26. while (frameCounter < FRAMES) {
27. cv::Mat frameOrig;
28. cv::Mat frameProc;
29. clock\_t start = clock();
30. **capOrig.read(frameOrig);**
31. clock\_t end = clock();
32. double timeElapsedReading = (double)(end - start);
33. totalTime += timeElapsedReading;
34. inputTime += timeElapsedReading;
35. **start = clock();**
36. cv::cvtColor(frameOrig, frameProc, cv::COLOR\_BGR2GRAY);
37. end = clock();
38. double timeElapsedProcessing = (double)(end - start);
39. totalTime += timeElapsedProcessing;
40. **procTime += timeElapsedProcessing;**
41. if (!frameProc.empty()) {
42. start = clock();
43. cv::imshow("Video Stream Proc", frameProc);
44. end = clock();
45. **double timeElapsedShow = (double)(end - start);**
46. totalTime += timeElapsedShow;
47. outputTime += timeElapsedShow;
48. }
49. start = clock();
50. **if (cv::waitKey(DELAY) == 'q') {**
51. break;
52. }
53. end = clock();
54. double timeElapsedDelay = (double)(end - start);
55. **totalTime += timeElapsedDelay;**
56. delayTime += timeElapsedDelay;
57. frameCounter++;
58. }
59. ofstream timeFile;
60. **timeFile.open("../time.txt");**
61. long double totalTimeSec = totalTime / CLOCKS\_PER\_SEC;
62. timeFile << "DELAY: **\t**" << DELAY << endl;
63. timeFile << "FRAMES: **\t**" << FRAMES << endl;
64. timeFile << "time for reading frames: **\t**" << inputTime / totalTime \* 100.0 << "%" << endl;
65. **timeFile << "time for processing frames: \t" << procTime / totalTime \* 100.0 << "%" << endl;**
66. timeFile << "time for output frames: **\t**" << outputTime / totalTime \* 100.0 << "%" << endl;
67. timeFile << "time for delay: **\t**" << delayTime / totalTime \* 100.0 << "%" << endl;
68. timeFile << "FPS: " << (double)(frameCounter) / (totalTimeSec) << endl;
69. timeFile.close();
70. **capOrig.release();**
71. cv::destroyAllWindows();
72. return 0;
73. }