###### 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 ПРАКТИЧЕСКОЙ РАБОТЫ**

«ВЫСОКОУРОВНЕВАЯ/НИЗКОУРОВНЕВАЯ РАБОТА С ПЕРИФЕРИЙНЫМИ УСТРОЙСТВАМИ»

студента второго курса, группы 22204

Мустакаева Дмитрия Дмитриевича

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

Доцент кафедры параллельных вычислений

А.Ю. Власенко

Новосибирск 2023

**СОДЕРЖАНИЕ**

[ЦЕЛЬ 3](#_gjdgxs)

[ЗАДАНИЕ 3](#_30j0zll)

[ОПИСАНИЕ РАБОТЫ 4](#_3znysh7)

[ЗАКЛЮЧЕНИЕ](#_2et92p0) 10

[Приложение 1. Реализация вывода с преобразованием](#_tyjcwt) 11

[Приложение 2. Реализация вывода списка USB устройств](#_3dy6vkm) 13

# ЦЕЛЬ

Ознакомиться с программированием периферийных устройств на примере ввода данных с Web-камеры с использованием библиотеки OpenCV и с началами низкоуровневого программирования периферийных устройств на примере получения информации о доступных USB-устройствах с помощью библиотеки libusb.

# ЗАДАНИЕ

Лабораторная работа №5:

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

Лабораторная работа №6:

Реализовать программу, получающую список всех подключенных к машине USB устройств с использованием libusb, для каждого найденного устройства напечатать его класс, идентификатор производителя и идентификатор изделия, изучить состав и характеристики обнаруженных с помощью реализованной программ USB устройств и дополнить программу, реализованную в п. 2 функцией печати серийного номера USB устройства.

# ОПИСАНИЕ РАБОТЫ

Лабораторная работа №5:

1. Была реализована программа на языке программирования C++ с использованием библиотеки OpenCV, которая получает поток видеоданных с веб-камеры и выводит его на экран

Была реализована программа с использованием OpenCV, которая получает поток видеоданных с камеры и выводит его на экран. (Рис. 1)



Рис. 1. Изображение с камеры.

2. Также было выполнено преобразование “Наложение клоунского носа” на изображение. (Рис. 2)

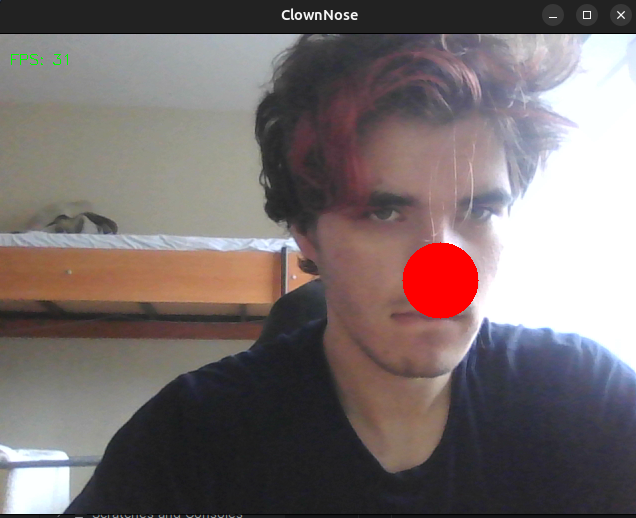


Рис. 2. Изображение с фильтром.

3. Затем было измерено кол-во кадров обрабатываемое камерой в секунду. И добавлен вывод кол-ва FPS на экран

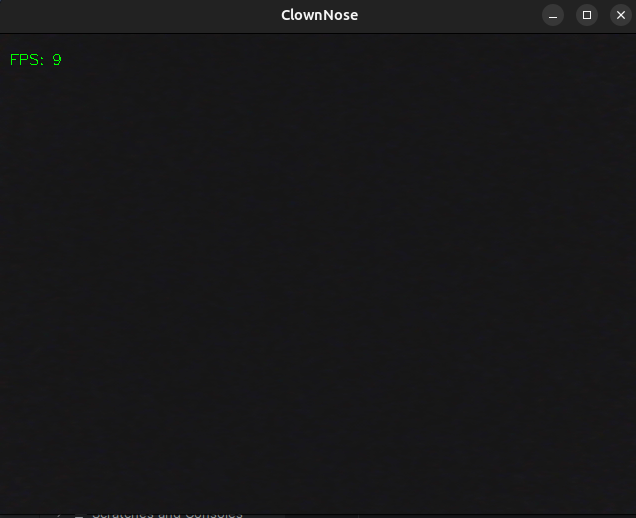


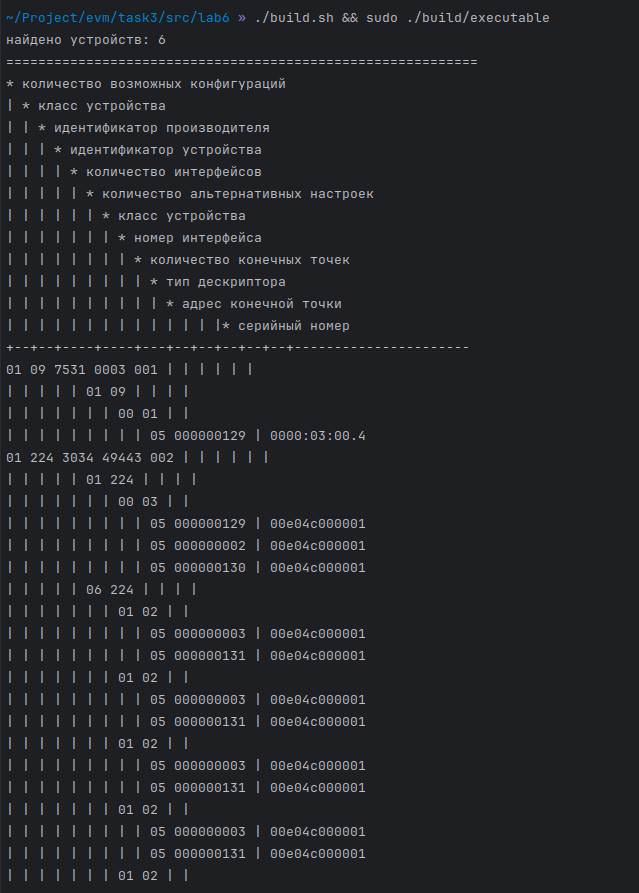
Рис. 3. Замер кадров с закрытой веб-камерой.

Лабораторная работа №6:

1. Была реализована программа, получающая список всех подключенных к машине USB устройств с использованием libusb, чтобы для каждого найденного устройства напечатан его класс, идентификатор производителя и идентификатор изделия. (Рис. 5)

2. Был изучен состав и характеристики обнаруженных с помощью реализованной программ USB устройств.

3. Была дополнена программа, реализованную в п. 2 функцией печати серийного номера USB устройства.



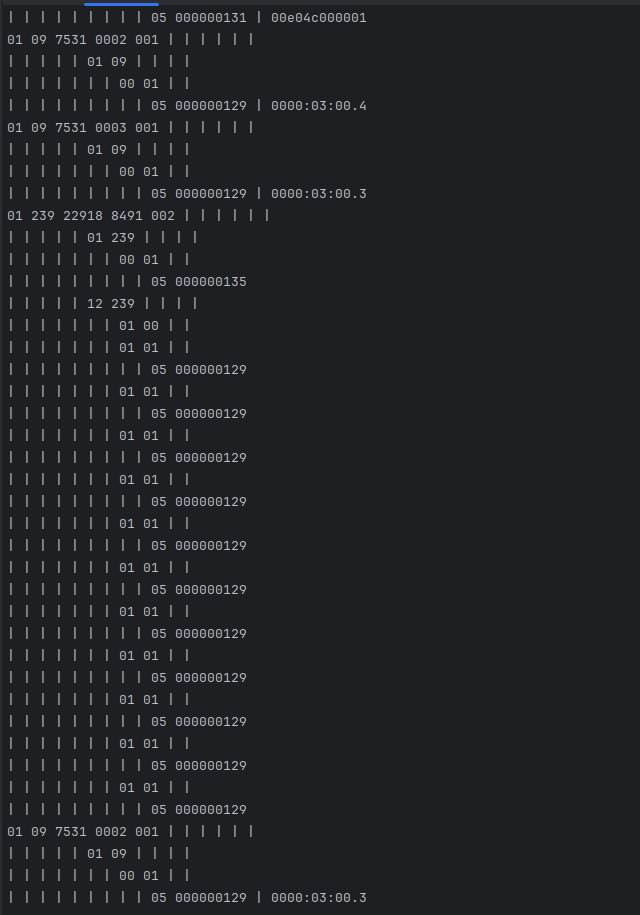


Рис. 5. Использование libusb.

# ЗАКЛЮЧЕНИЕ

В заключение можно сказать, что для ознакомления с программированием периферийных устройств на примере ввода данных с Web-камеры с использованием библиотеки OpenCV и с началами низкоуровневого программирования периферийных устройств на примере получения информации о доступных USB-устройствах с помощью библиотеки libusb были реализованы программы: с использованием OpenCV, которая получает поток видеоданных с камеры и выводит его на экран, и получающую список всех подключенных к машине USB устройств программу с использованием libusb, для каждого найденного устройства печатающего его класс, идентификатор производителя и идентификатор изделия.

# ПРИЛОЖЕНИЕ 1. Реализация вывода с преобразованием

#include <opencv2/highgui/highgui.hpp>

#include <opencv2/opencv.hpp>

using namespace cv;

using namespace std;

int main(void) {

VideoCapture cap(0);

if (!cap.isOpened()) {

cerr << "Camera Error" << endl;

return -1;

}

CascadeClassifier noseCascade;

if (!noseCascade.load("../haarcascade\_mcs\_nose.xml")) {

cerr << "Nose Classificator Error" << endl;

return -1;

}

int fps = 0;

double start\_time = static\_cast<double>(getTickCount());

while (true) {

double frame\_start\_time = static\_cast<double>(getTickCount());

Mat frame;

cap >> frame;

if (frame.empty()) {

cerr << "Empty Camera" << endl;

break;

}

vector<Rect> noses;

noseCascade.detectMultiScale(frame, noses, 1.3, 4);

for (const Rect& noseRect : noses) {

int noseX = noseRect.x + noseRect.width / 2;

int noseY = noseRect.y + noseRect.height / 2;

int radius = noseRect.width / 2;

Scalar red(0, 0, 255);

circle(frame, Point(noseX, noseY), radius, red, -1);

}

double frame\_end\_time = static\_cast<double>(getTickCount());

double frame\_elapsed\_time = (frame\_end\_time - frame\_start\_time) / getTickFrequency();

// putText(frame, "Frame Time: " + to\_string(frame\_elapsed\_time) + "s", Point(10, 60), FONT\_HERSHEY\_SIMPLEX, 0.5, Scalar(255, 255, 255), 1);

double end\_time = static\_cast<double>(getTickCount());

double elapsed\_time = (end\_time - start\_time) / getTickFrequency();

fps = static\_cast<int>(1.0 / elapsed\_time);

start\_time = end\_time;

putText(frame, "FPS: " + to\_string(fps), Point(10, 30), FONT\_HERSHEY\_SIMPLEX, 0.5, Scalar(0, 255, 0), 1);

imshow("ClownNose", frame);

char c = waitKey(1);

if (c == 'z' || c == 27)

break;

}

cap.release();

destroyAllWindows();

return 0;

}

# ПРИЛОЖЕНИЕ 2. Реализация вывода списка USB устройств

#include <iostream>

#include <libusb-1.0/libusb.h>

#include <stdio.h>

using namespace std;

void printdev(libusb\_device \*dev);

int main() {

libusb\_device\*\* devs; // указатель на указатель на устройство,// используется для получения списка устройств

libusb\_context\* ctx = NULL; // контекст сессии libusb

int r; // для возвращаемых значений

ssize\_t cnt; // число найденных USB-устройств

ssize\_t i; // индексная переменная цикла перебора всех устройств

// инициализировать библиотеку libusb, открыть сессию работы с libusb

r = libusb\_init(&ctx);

if (r < 0) {

fprintf(stderr, "Ошибка: инициализация не выполнена, код: %d.\n", r);

return 1;

}

// задать уровень подробности отладочных сообщений

// libusb\_set\_debug(ctx, 3);

// получить список всех найденных USB- устройств

cnt = libusb\_get\_device\_list(ctx, &devs);

if (cnt < 0) {

fprintf(stderr, "Ошибка: список USB устройств не получен.\n");

return 1;

}

printf("найдено устройств: %ld\n", cnt);

printf("===========================================================\n");

printf("\* количество возможных конфигураций\n");

printf("| \* класс устройства\n");

printf("| | \* идентификатор производителя\n");

printf("| | | \* идентификатор устройства\n");

printf("| | | | \* количество интерфейсов\n");

printf(

"| | | | | \* количество "

"альтернативных настроек\n");

printf("| | | | | | \* класс устройства\n");

printf("| | | | | | | \* номер интерфейса\n");

printf(

"| | | | | | | | \* количество "

"конечных точек\n");

printf("| | | | | | | | | \* тип дескриптора\n");

printf(

"| | | | | | | | | | \* адрес "

"конечной точки\n");

printf("| | | | | | | | | | | | | |\* серийный номер\n");

printf(

"+--+--+----+----+---+--+--+--+"

"--+--+----------------------\n");

for (i = 0; i < cnt; i++) { // цикл перебора всех устройств

printdev(devs[i]); // печать параметров устройства

}

printf(

"=============================="

"=============================\n");

// освободить память, выделенную функцией получения списка устройств

libusb\_free\_device\_list(devs, 1);

libusb\_exit(ctx); // завершить работу с библиотекой libusb,

// закрыть сессию работы с libusb

return 0;

}

void printdev(libusb\_device\* dev) {

libusb\_device\_descriptor desc;

libusb\_config\_descriptor\* config;

const libusb\_interface\* inter;

const libusb\_interface\_descriptor\* interdesc;

const libusb\_endpoint\_descriptor\* epdesc;

int r = libusb\_get\_device\_descriptor(dev, &desc);

if (r < 0) {

fprintf(stderr, "Ошибка: дескриптор устройства не получен, код: %d.\n", r);

return;

}

libusb\_device\_handle\* handle;

r = libusb\_open(dev, &handle); // Open the device to get a handle

if (r != 0) {

fprintf(stderr, "Ошибка: не удалось открыть устройство, код: %d.\n", r);

return;

}

// получить конфигурацию устройства

libusb\_get\_config\_descriptor(dev, 0, &config);

printf("%.2d %.2d %.4d %.4d %.3d | | | | | |\n", (int)desc.bNumConfigurations, (int)desc.bDeviceClass,

desc.idVendor, desc.idProduct, (int)config->bNumInterfaces);

for (int i = 0; i < (int)config->bNumInterfaces; i++) {

inter = &config->interface[i];

printf(

"| | | | | "

"%.2d %.2d | | | |\n",

inter->num\_altsetting, (int)desc.bDeviceClass);

for (int j = 0; j < inter->num\_altsetting; j++) {

interdesc = &inter->altsetting[j];

printf(

"| | | | | | | "

"%.2d %.2d | |\n",

(int)interdesc->bInterfaceNumber, (int)interdesc->bNumEndpoints);

for (int k = 0; k < (int)interdesc->bNumEndpoints; k++) {

epdesc = &interdesc->endpoint[k];

printf(

"| | | | | | | | | "

"%.2d %.9d",

(int)epdesc->bDescriptorType, (int)(int)epdesc->bEndpointAddress);

// Print serial number if available

char serialNumber[256];

r = libusb\_get\_string\_descriptor\_ascii(handle, desc.iSerialNumber, (unsigned char\*)serialNumber, sizeof(serialNumber));

if (r > 0) {

printf(" | %s\n", serialNumber);

} else {

printf("\n");

}

}

}

}

libusb\_close(handle); // Close the handle when done

libusb\_free\_config\_descriptor(config);

}