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 и низкоуровневая работы с периферийными устройствами»

студента 2 курса, группы 21206

Балашова Вячеслава Вадимовича

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:
Кандидат технических наук
А. Ю. Власенко

Новосибирск 2022

Содержание

Цель

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

2. Ознакомиться с началами низкоуровневого программирования периферийных устройств на примере получения информации о доступных USB-устройствах с помощью библиотеки libusb.

Задание

1. Реализовать программу№1 с использованием OpenCV, которая получает поток видеоданных с камеры и выводит его на экран.

2. Выполнить произвольное преобразование изображения (**кроме** указанных в computerlab5.pdf сглаживания и установки значений цветовых каналов в константу).

3. Измерить количество кадров, обрабатываемое программой в секунду.

Оценить долю времени, затрачиваемого процессором на обработку (ввод, преобразование, показ) видеоданных, получаемых с камеры.

4. Реализовать программу №2, получающую список всех подключенных к машине USB устройств с использованием libusb. Для каждого найденного устройства напечатать его класс, идентификатор производителя, идентификатор изделия и серийный номер.

5. Требования к содержанию отчета:

- «чистое» неоткорректированное изображение, полученное с камеры;
- это же изображение в преобразованном виде;
- полный код программы №1, выполняющей преобразование изображения;
- оценку скорости обработки видео (кадров в секунду) и долю времени, затрачиваемого процессором на ввод, обработку и показ видеоданных;

6. Полный код программы №2, выводящей информацию по USB-устройствам.

7. Описание обнаруженных USB-устройств

Описание работы

Ход работы

1. Была скачана библиотека OpenCV для языка C++
2. Был создан файл main.cpp с исходным кодом программы (см. Приложение 1.), преобразующей изображение с камеры ноутбука с помощью библиотеки (OpenCV)

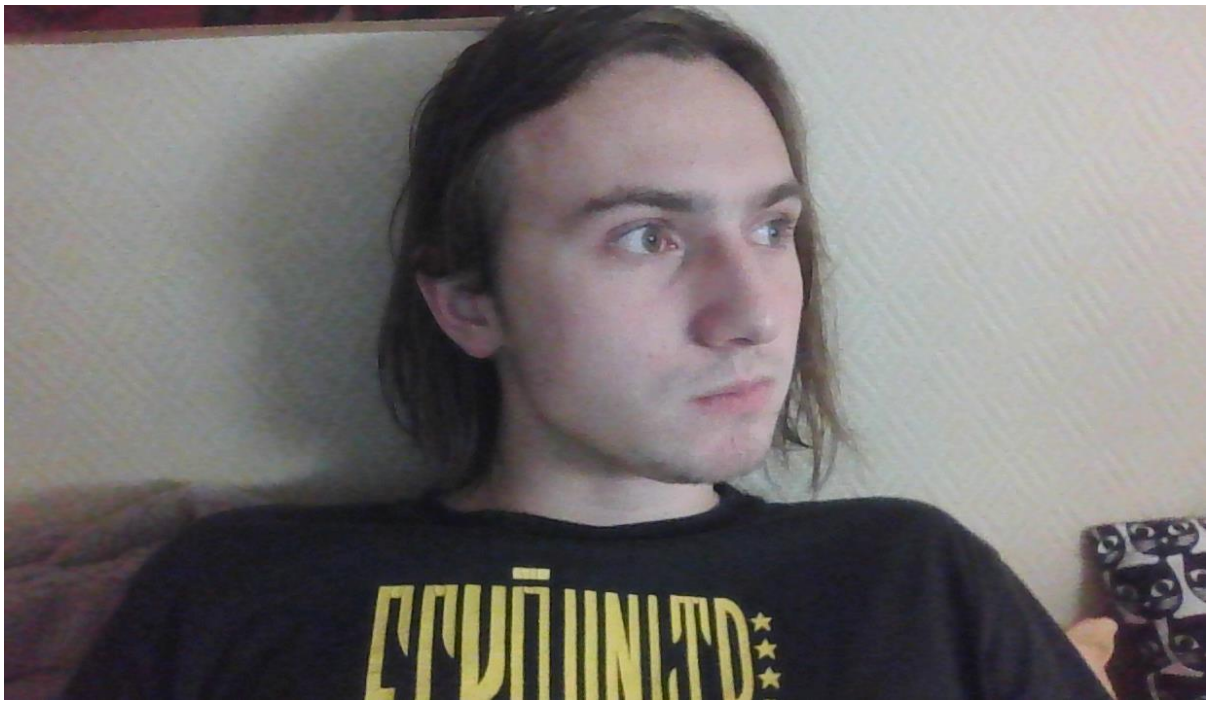

Рис. 1. Изображение, полученное с камеры без обработки

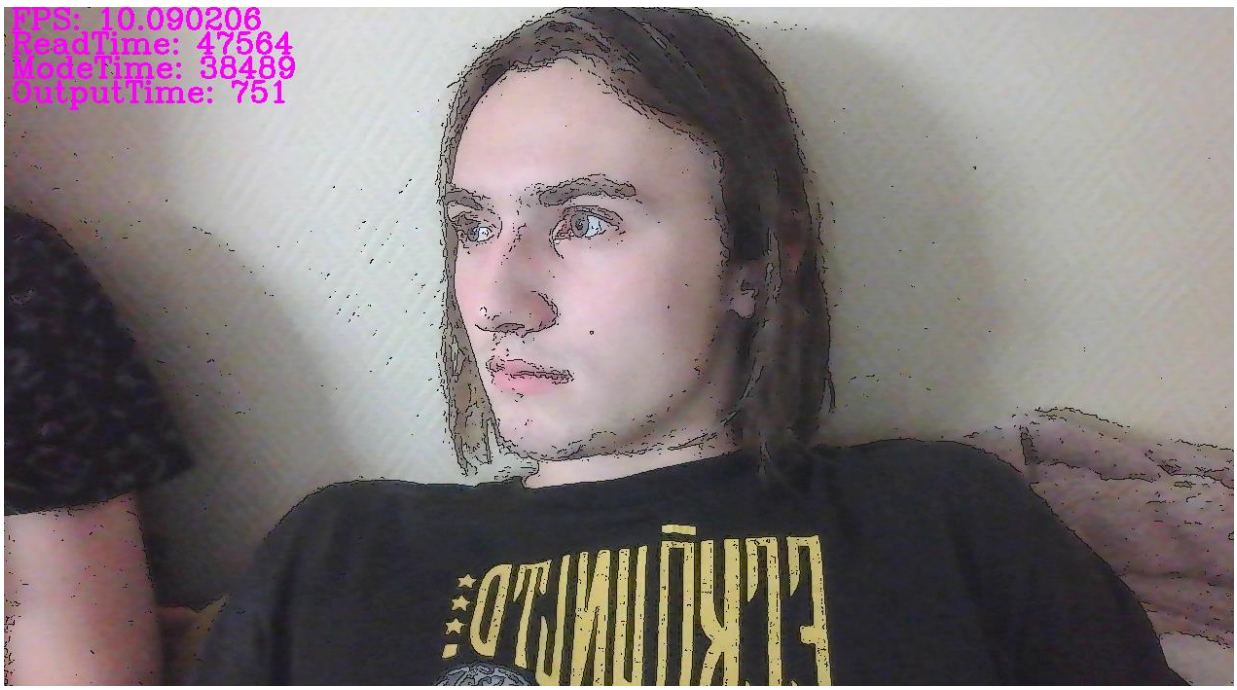

Рис. 2. Изображение, полученное с камеры с обработкой

3. Была скачана библиотека libusb для языка C++

4. Был создан файл usb.cpp с исходным кодом программы, выводящей информацию о usb устройствах, подключенных к ПК (см. Приложение 3.), используя библиотеку libusb.

5. Оба файла были скомпилированы с нужными флагами и результат работы программ будет описан в заключении

Заключение

В ходе выполнения работы было произведено ознакомление с программированием периферийных устройств.

Было выполнено ознакомление с библиотекой OpenCV для языка программирования C++ и сделаны следующие преобразования видео:

a. Изображение отражается (зеркалится)

b. Создается размытая копия изображения

c. На основе размытой копии создается изображение с повышенной резкостью

d. Создается черно-белое изображение, показывающее обрамление предметов

e. Набор точек с этого изображения сохраняется в массив

f. Эти точки накладываются на отзеркаленное изображение

g. Добавляется счетчик кадров, время считывания кадра, время обработки кадра, время вывода кадра

Тем самым были изучены базовые функции данной библиотеки. Путем простых расчетов было выяснено, что на считывание кадра тратится 54% от общего времени, на обработку – 44%, на вывод – 0.8%.

Также было произведено ознакомление с низкоуровневой библиотекой libusb, позволяющей получить информацию о подключенных usb-устройствах

Исходный код программы можно найти в Приложении 2. Результат работы программы можно найти в Приложении 3. Программа выводит всю информацию о подключенных устройствах, основное:

1) Количество подключенных устройств

2) Описание устройства

3) Серийный номер устройства

4) Производителя устройства

Полный список того, что выводит программа можно найти в Приложении 3.

Приложение 1. (Программа на языке C++ с OpenCV)

```
#include <opencv2/opencv.hpp>
#include <ctime>
#include <sys/time.h>

using namespace std;
using namespace cv;

int main()
{
    VideoCapture cap(0);
    Mat src;
    Mat flipped;
    Mat canny_;

    Mat filtered;

    timeval frameStart;
    size_t  frameStart_ms = 0;

    timeval readTime;
    size_t  readTime_ms = 0;

    timeval modeTime;
    size_t  modeTime_ms = 0;

    timeval outputTime;
    size_t  outputTime_ms = 0;

    timeval finalTime;
    size_t  finalTime_ms = 0;

    double FPS = 0;
```

```
for (;;)
{
    gettimeofday(&frameStart, nullptr);
    cap.read(src);
    gettimeofday(&readTime, nullptr);

    flip(src, flipped, 1);

    GaussianBlur(flipped, filtered, Size(0, 0), 15);

    addWeighted(flipped, 1.5, filtered, -0.5, 10, filtered);

    Canny(filtered, canny_, 100, 100);

    vector<vector<Point>> contours;
    vector<Vec4i> hierarchy;

    findContours(canny_, contours, hierarchy, RETR_EXTERNAL,
CHAIN_APPROX_SIMPLE);
    drawContours(flipped, contours, -1, Scalar(25, 25, 25), 1);

    putText(flipped, "FPS: " + to_string(FPS),
        Point(10, 25), FONT_HERSHEY_COMPLEX,
        1, Scalar(255, 0, 255), 2);

    putText(flipped, "ReadTime: " + to_string(readTime_ms - frameStart_ms),
        Point (10, 50), FONT_HERSHEY_COMPLEX,
        1, Scalar(255, 0, 255), 2);

    putText(flipped, "ModeTime: " + to_string(modeTime_ms - readTime_ms),
        Point (10, 75), FONT_HERSHEY_COMPLEX,
        1, Scalar(255, 0, 255), 2);

    putText(flipped, "OutputTime: " + to_string(outputTime_ms - modeTime_ms),
        Point (10, 100), FONT_HERSHEY_COMPLEX,
```

```
            1, Scalar(255, 0, 255), 2);

        gettimeofday(&modeTime, nullptr);

        imshow("Final", flipped);
        imshow("Original", src);

        gettimeofday(&outputTime, nullptr);

        if (waitKey(10) >= 0)
        {
           break;
        }

        frameStart_ms = (frameStart.tv_sec * 1000000) + frameStart.tv_usec;
        readTime_ms   = (readTime  .tv_sec * 1000000) + readTime  .tv_usec;
        modeTime_ms   = (modeTime  .tv_sec * 1000000) + modeTime  .tv_usec;
        outputTime_ms = (outputTime.tv_sec * 1000000) + outputTime.tv_usec;

        gettimeofday(&finalTime, nullptr);
        finalTime_ms  = (finalTime .tv_sec * 1000000) + finalTime .tv_usec;
        FPS = (1. * 1000000.) / double(finalTime_ms - frameStart_ms);
    }

    return 0;
}
```

Приложение 2. (Программа на языке C++ с libusb)

```
#include <libusb-1.0/libusb.h>
#include <stdio.h>

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_option(ctx, LIBUSB_OPTION_MAX);

   // 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("+--+----+------+------+---+--+---+--+"
       "--+--+----------------------\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;                  // дескриптор устройства
    libusb_config_descriptor *config;               // дескриптор конфигурации объекта
    const libusb_interface *inter;                  // Интерфейс
    const libusb_interface_descriptor *interdesc;   // Дескриптор интерфейса
    const libusb_endpoint_descriptor *epdesc;       // Дескриптор конечной точки

    // Строки для вывода основных данных
    unsigned char uSerialNumber[256];
    unsigned char uManufacturer[256];
    unsigned char uProduct[256];

    libusb_device_handle* devicehandle;

    // Получает дескриптор устройства
    {
        int error = libusb_get_device_descriptor(dev, &desc); // Получает дескриптор устройства
        if (error < 0)  // Если ошибка
        {
            fprintf(stderr,
                    "Ошибка: дескриптор устройства не получен, код: %d.\n", error);
            return;
        }
    }

    // Позволяет получить доп инфо об устройстве
    {
        int error = libusb_open(dev, &devicehandle);
        if (error < 0)
        {
```

```
        printf("Ошибка: дескриптор устройства не получен, код: %d.\n", error);
        return;
    }
}

// Получает информацию о серийном номере устройства
libusb_get_string_descriptor_ascii(devicehandle, desc.iSerialNumber,
                                   uSerialNumber, 256);

// Получает информацию о устройстве
{
    int error =
        libusb_get_string_descriptor_ascii(devicehandle, desc.iProduct,
                                   uProduct, 256);
    if (error < 0)
    {
        fprintf(stderr,
            "Ошибка: строковый дескриптор устройства "
            "для названия не получен, код: %d.\n", error);
        return;
    }
}

// Получает информацию о производителе устройства
{
    int error =
        libusb_get_string_descriptor_ascii(devicehandle, desc.iManufacturer,
                                   uManufacturer, 256);
    if (error < 0)
    {
        fprintf(stderr,
            "Ошибка: строковый дескриптор устройства "
            "для названия производителя не получен, код: %d.\n", error);
        return;
    }
```

```
        }

        printf("Описание устройства: %s\n", uProduct);
        printf("Серийный номер устройства: %s\n", uSerialNumber);
        printf("Производитель  устройства: %s\n", uManufacturer);

        // получить конфигурацию устройства
        libusb_get_config_descriptor(dev, 0, &config);

        printf("%.2d %.4d %.6d %.6d %.3d | |  | | | |\n",  // Выводит
            (int) desc.bNumConfigurations,   // Число конфигураций
            (int) desc.bDeviceClass ? (int) desc.bDeviceClass : interdesc->bInterfaceClass,   // Класс
            // Проверяет на то, является ли класс пустым. Если да, то берет значение из дескриптора интерфейса
            desc.idVendor,   // ID Вендора (Производителя)
            desc.idProduct,  // ID Устройства
            (int) config->bNumInterfaces // Количество интерфейсов
        );

        for (int i = 0; i < (int) config->bNumInterfaces; i++)  // Проходится по всем интерфейсам
        {
           inter = &config->interface[i];      // Получает текущий интерфейс
           printf("| |   |     |     |   " // Выводит
                "%.2d %.3d | | | |\n",
                inter->num_altsetting,       // Количество альтернативных настроек
                (int) desc.bDeviceClass ? (int) desc.bDeviceClass : interdesc->bInterfaceClass   // Класс
           );

           for (int j = 0; j < inter->num_altsetting; j++) // Проходит по всем альтернативным настройкам
           {
              interdesc = &inter->altsetting[j];
```

```
            printf("| |   |     |     | | |   "  // Выводит
                "%.2d %.2d |  |\n",
                (int) interdesc->bInterfaceNumber,   // Номер интерфейса
                (int) interdesc->bNumEndpoints       // Число конечных точек
            );
            for (int k = 0; k < (int) interdesc->bNumEndpoints; k++)     // Проходит по всем конечным точкам
            {
               epdesc = &interdesc->endpoint[k];
               printf(                              // Выводит
                    "| |   |     |     | | |  | | "
                    "%.2d %.9d\n",
                    (int) epdesc->bDescriptorType,  // Тип дескриптора
                    (int) epdesc->bEndpointAddress  // Адрес конечной точки
               );
            }
          }
       }
       libusb_close(devicehandle);
       libusb_free_config_descriptor(config); // освобождает выделенную память
       printf("================================="
            "=================================\n");
    }
```

Приложение 3. (Результат работы программы в файле usb.cpp)

```
найдено устройств: 8
================================================================
=
* количество возможных конфигураций
| * класс устройства
| |   * идентификатор производителя
| |   |     * идентификатор устройства
| |   |     |     * количество интерфейсов
| |   |     |     |  * количество альтернативных настроек
| |   |     |     | | * класс устройства
| |   |     |     | | |  * номер интерфейса
| |   |     |     | | | | * количество конечных точек
| |   |     |     | | | | | * тип дескриптора
| |   |     |     | | | | | | * адрес конечной точки
+--+----+------+------+---+--+---+--+--+--+----------------------
Описание устройства: xHCI Host Controller
Серийный номер устройства: 0000:04:00.4
Производитель  устройства: Linux 5.15.74-3-MANJARO xhci-hcd
01 0009 007531 000003 001 | |  | | | |
| |   |     |     |  01 009 | | | |
| |   |     |     | | |  00 01 | |
| |   |     |     | | | | | 05 000000129
================================================================
=
Описание устройства: Bluetooth Radio
Серийный номер устройства: 00e04c000001
Производитель  устройства: Realtek
01 0224 003034 049443 002 | |  | | | |
| |   |     |     |  01 224 | | | |
| |   |     |     | | |  00 03 | |
| |   |     |     | | | | | 05 000000129
| |   |     |     | | | | | 05 000000002
```

| | | | | | | | 05 000000130

| | | | | 06 224 | | | |

| | | | | | | 01 02 | |

| | | | | | | | 05 000000003

| | | | | | | | 05 000000131

| | | | | | | 01 02 | |

| | | | | | | | 05 000000003

| | | | | | | | 05 000000131

| | | | | | | 01 02 | |

| | | | | | | | 05 000000003

| | | | | | | | 05 000000131

| | | | | | | 01 02 | |

| | | | | | | | 05 000000003

| | | | | | | | 05 000000131

| | | | | | | 01 02 | |

| | | | | | | | 05 000000003

| | | | | | | | 05 000000131

| | | | | | | 01 02 | |

| | | | | | | | 05 000000003

| | | | | | | | 05 000000131

==========

=

Описание устройства: ELAN:Fingerprint

Серийный номер устройства: 00e04c000001

Производитель устройства: ELAN

01 0224 001267 003149 001 | | | | | |

| | | | | 01 224 | | | |

| | | | | | | 00 05 | |

| | | | | | | | 05 000000129

| | | | | | | | 05 000000001

| | | | | | | | 05 000000130

| | | | | | | | 05 000000131

| | | | | | | | 05 000000003

==========

=

Описание устройства: xHCI Host Controller

Серийный номер устройства: 0000:04:00.4

Производитель устройства: Linux 5.15.74-3-MANJARO xhci-hcd

01 0009 007531 000002 001 | | | | | |

| | | | | 01 009 | | | |

| | | | | | | 00 01 | |

| | | | | | | | | 05 000000129

==========

=

Описание устройства: xHCI Host Controller

Серийный номер устройства: 0000:04:00.3

Производитель устройства: Linux 5.15.74-3-MANJARO xhci-hcd

01 0009 007531 000003 001 | | | | | |

| | | | | 01 009 | | | |

| | | | | | | 00 01 | |

| | | | | | | | | 05 000000129

==========

=

Описание устройства: Integrated Camera

Серийный номер устройства: 0000:04:00.3

Производитель устройства: SunplusIT Inc

01 0239 022918 008491 002 | | | | | |

| | | | | 01 239 | | | |

| | | | | | | 00 01 | |

| | | | | | | | | 05 000000135

| | | | | 12 239 | | | |

| | | | | | | 01 00 | |

| | | | | | | 01 01 | |

| | | | | | | | | 05 000000129

| | | | | | | 01 01 | |

| | | | | | | | | 05 000000129

| | | | | | | 01 01 | |

| | | | | | | | | 05 000000129

| | | | | | | 01 01 | |

| | | | | | | | | 05 000000129

| | | | | | 01 01 | |

| | | | | | | | 05 000000129

| | | | | | 01 01 | |

| | | | | | | | 05 000000129

| | | | | | 01 01 | |

| | | | | | | | 05 000000129

| | | | | | 01 01 | |

| | | | | | | | 05 000000129

| | | | | | 01 01 | |

| | | | | | | | 05 000000129

| | | | | | 01 01 | |

| | | | | | | | 05 000000129

| | | | | | 01 01 | |

| | | | | | | | 05 000000129

==========

=

Описание устройства: USB Gaming Mouse

Серийный номер устройства: 0000:04:00.3

Производитель устройства: E-Signal

01 0014 004809 004119 003 | | | | | |

| | | | | 01 014 | | | |

| | | | | | 00 01 | |

| | | | | | | | 05 000000129

| | | | | 01 003 | | | |

| | | | | | 01 01 | |

| | | | | | | | 05 000000130

| | | | | 01 003 | | | |

| | | | | | 02 02 | |

| | | | | | | | 05 000000131

| | | | | | | | 05 000000004

==========

=

Описание устройства: xHCI Host Controller

Серийный номер устройства: 0000:04:00.3

Производитель устройства: Linux 5.15.74-3-MANJARO xhci-hcd

01 0009 007531 000002 001 | | | | | |

| | | | | 01 009 | | | |

| | | | | | | 00 01 | |

| | | | | | | | 05 000000129

==========

=

==========

=