Балтийский государственный технический университет
«ВОЕНМЕХ» им. Д. Ф. Устинова

Кафедра О7 «Информационные системы и программная инженерия»

Практическая работа №1
по дисциплине «Системное программное обеспечение»
Взаимодействие с устройством USB-HID

Выполнил:
Студент *Альков В.С.*
Группа *И407Б*

Преподаватель:
Никитин С.С.

Санкт-Петербург
2022 г.

Цель работы - создание приложения, управляющего устройством через интерфейс USB HID.

Задачи:

1. Читать в цикле значение состояния переменного резистора, возвращаемого функцией ADC_V(), преобразовывать его значение к десятичному виду и выводить на экран.
2. Полученное значение в каждом выполнении цикла передавать на устройство в функцию изменения яркости светодиода.
3. Вывести на экран точку используя функцию 0x04 (требуется передать дри 8-и разрядных числа — координаты по X,Y, и байт цвета)
4. Закрасить точками весь экран (размеры экрана 128*64 пикселя)
5. Обработать нажатие кнопки и при нажатии первой включить все светодиоды на полную яркость, а при нажатии второй — все погасить.

Список функций:

`void ClearScreen(unsigned char* buf,hid_device *handle, char color)` - функция попиксельной закраски экрана заданным цветом.

`void ChangeBright(unsigned char* buf,hid_device *handle)` - функция изменения яркости диодов по положению резистора.

`int Buttons(unsigned char* buf,hid_device *handle, int* flag)` - функция проверки состояния кнопок и изменения яркости диода в зависимости от результата проверки.

Листинг программы:

```
#define WIN32
#ifdef WIN32
#include <windows.h>
#else
#include <stdlib.h>
#endif
#include <stdio.h>
#include <stdlib.h>
#include "hidapi.h"
#include <stdint.h>
#define MAX_STR 255
#include <locale.h>
#include <string.h>
#include <wchar.h>

void ClearScreen(unsigned char* buf,hid_device *handle, char
color)
{
    int res;
    buf[0]=0x04;
    buf[3]=color;
    for (int i=0;i<128;i++)
    {
        for (int j=0;j<64;j++)
        {
            buf[1]=i;
            buf[2]=j;
            res = hid_send_feature_report(handle,buf,4);
        }
    }

}

void ChangeBright(unsigned char* buf,hid_device *handle)
{
    int res;
    buf[0]=0x02;//формируем команду для изменения яркости
светодиода
    buf[3]=buf[5]=buf[1];
    buf[4]=buf[6]=buf[2];
    res = hid_send_feature_report(handle,buf,7);
}

int Buttons(unsigned char* buf,hid_device *handle, int* flag)
{
    int res;
    buf[0]=0x1;
    res = hid_get_feature_report(handle, buf, sizeof(buf));
    if (res < 0)
     printf("Unable to read indexed string 1\n");
    else{
```

```
            if (buf[1]==1)
            {
                buf[0] = 0x02;
                buf[1]=buf[2]=buf[3]=buf[4]=0;
                buf[5]=buf[6]=0xff;
                res = hid_send_feature_report(handle,buf,7);
                return 1;
            }
            else
            {
                if (buf[1]==2)
                {
                    buf[0] = 0x02;
                    buf[1]=buf[2]=buf[3]=buf[4]=buf[5]=buf[6]=0;
                    res = hid_send_feature_report(handle,buf,7);
                    return 1;
                }
            }
        }
        return 0;
    }

    int main(int argc, char* argv[])
    {
        (void)argc;
        (void)argv;

    int res;
    unsigned char buf[256];
    #define MAX_STR 255
    wchar_t wstr[MAX_STR];
    hid_device *handle;
    int i;

    struct hid_device_info *devs, *cur_dev;

    printf("hidapi test/example tool. Compiled with hidapi version
%s, runtime version %s.\n", HID_API_VERSION_STR, hid_version_str());
    if (hid_version()->major == HID_API_VERSION_MAJOR &&
hid_version()->minor == HID_API_VERSION_MINOR && hid_version()-
>patch == HID_API_VERSION_PATCH) {
        printf("Compile-time version matches runtime version of
hidapi.\n\n");
    }
    else {
        printf("Compile-time version is different than runtime
version of hidapi.\n]n");
    }

    if (hid_init())
        return -1;
```

```
        // находим все устройства USB HID, печатаем содержимое
дескриптора устройства, доступное через драйвер
        devs = hid_enumerate(0x0, 0x0);
    cur_dev = devs;
    while (cur_dev) {
        printf("Device Found\n  type: %04hx %04hx\n  path: %s\n
serial_number: %ls", cur_dev->vendor_id, cur_dev->product_id,
cur_dev->path, cur_dev->serial_number);
        printf("\n");
        printf("  Manufacturer: %ls\n", cur_dev-
>manufacturer_string);
        printf("  Product:      %ls\n", cur_dev->product_string);
        printf("  Release:      %hx\n", cur_dev->release_number);
        printf("  Interface:    %d\n",  cur_dev-
>interface_number);
        printf("  Usage (page): 0x%hx (0x%hx)\n", cur_dev->usage,
cur_dev->usage_page);
        printf("\n");
        cur_dev = cur_dev->next;
    }
    hid_free_enumeration(devs);

    // Set up the command buffer.
    memset(buf,0x00,sizeof(buf));
    buf[0] = 0x01;
    buf[1] = 0x81;


    // Open the device using the VID, PID,
    // and optionally the Serial number.
    ////handle = hid_open(0x4d8, 0x3f, L"12345");
    handle = hid_open(0x1234, 0x0001, NULL);
    if (!handle) {
        printf("unable to open device\n");
            return 1;
    }


    // Read the Manufacturer String
    wstr[0] = 0x0000;
    res = hid_get_manufacturer_string(handle, wstr, MAX_STR);
    if (res < 0)
        printf("Unable to read manufacturer string\n");
    printf("Manufacturer String: %ls\n", wstr);

    // Read the Product String
    wstr[0] = 0x0000;
    res = hid_get_product_string(handle, wstr, MAX_STR);
    if (res < 0)
        printf("Unable to read product string\n");
    printf("Product String: %ls\n", wstr);

    // Read the Serial Number String
```

```
    wstr[0] = 0x0000;
    res = hid_get_serial_number_string(handle, wstr, MAX_STR);
    if (res < 0)
         printf("Unable to read serial number string\n");
    printf("Serial Number String: (%d) %ls", wstr[0], wstr);
    printf("\n");

    // Read Indexed String 1
    wstr[0] = 0x0000;
    res = hid_get_indexed_string(handle, 1, wstr, MAX_STR);
    if (res < 0)
         printf("Unable to read indexed string 1\n");
    printf("Indexed String 1: %ls\n", wstr);

        // LEDs lights
            buf[0] = 0x02; // descriptor number
            buf[1] = 0xff; //
            buf[2] = 0xff; // 2 byte = uint16_t = power of light
color 1
            buf[3] = 0x00; //
            buf[4] = 0x00; // 2 byte = uint16_t = power of light
color 2
            buf[5] = 0xff; //
            buf[6] = 0xff; // 2 byte = uint16_t = power of light
color 3

            res = hid_send_feature_report(handle,buf,7); // send
report, 7 byte
            if(res == -1) {
                printf("hid_write error.\n");
            }
         // keys
         // Read a Feature Report from the device
    buf[0] = 0x1;
    res = hid_get_feature_report(handle, buf, sizeof(buf));
    if (res < 0) {
         printf("Unable to get a feature report.\n");
         printf("%ls", hid_error(handle));
    }
    else {
         // Print out the returned buffer.
         printf("Feature Report\n   ");
         for (i = 0; i < res; i++)
              printf("%02hhx ", buf[i]);
         printf("\n");
    }
    uint16_t tmpi;
    int flag=0;
    ClearScreen(buf,handle,0x00);
    Sleep(200);
    ClearScreen(buf,handle,0x01);
    while (1)
        {
```

```
            if(!Buttons(buf,handle,&flag)) //проверка состояния
кнопки 1 (если не нажата - считываем с резистора, если нажата -
светодиод делаем зеленым)
            {
                buf[0] = 0x3; //состояние слайдера
                res = hid_get_feature_report(handle, buf,
sizeof(buf));
                if (res < 0)
                {
                    printf("Unable to get a feature report.\n");
                    printf("%ls", hid_error(handle));
                }
                else
                {
                    tmpi=buf[1]+buf[2]*0x100; //переводим в 10 вид
                    printf("%d\n", tmpi); //печатаем значение
слайдера
                    ChangeBright(buf, handle);
                }
            }
            Sleep(50);
        }
        return 0;
    }
```

Вывод

В данной практической работе мы научились разрабатывать приложение, управляющие устройством через интерфейс USB HID.