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.*

**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 Х,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.