Федеральное государственное автономное образовательное учреждение высшего образования

Университет ИТМО

Дисциплина: Проектирование вычислительных систем

Лабораторная работа 3

Вариант 4

Выполнили:
Марков Петр Денисович
Кривоносов Егор Дмитриевич

Группа: P34111

Преподаватель:
Пинкевич Василий Юрьевич

2022 г.

Санкт-Петербург

Содержание

Задание

Разработать программу, которая использует таймеры для управления яркостью светодиодов и излучателем звука (по прерыванию или с использованием аппаратных каналов). Блокирующее ожидание (функция HAL_Delay()) в программе использоваться не должно.

Стенд должен поддерживать связь с компьютером по UART и выполнять указанные действия в качестве реакции на нажатие кнопок на клавиатуре компьютера. В данной лабораторной работе каждая нажатая кнопка (символ, отправленный с компьютера на стенд) обрабатываются отдельно, ожидание ввода полной строки не требуется.

Для работы с UART на стенде можно использован один из двух вариантов драйвера (по прерыванию и по опросу) на выбор исполнителя. Поддержка двух вариантов не требуется.

Вариант 4

Реализовать «музыкальную шкатулку» с мелодиями, которые состоят из последовательности звуков определенной частоты и длительности, а также пауз. Шкатулка должна иметь четыре стандартные мелодии и одну пользовательскую, которую можно настроить.

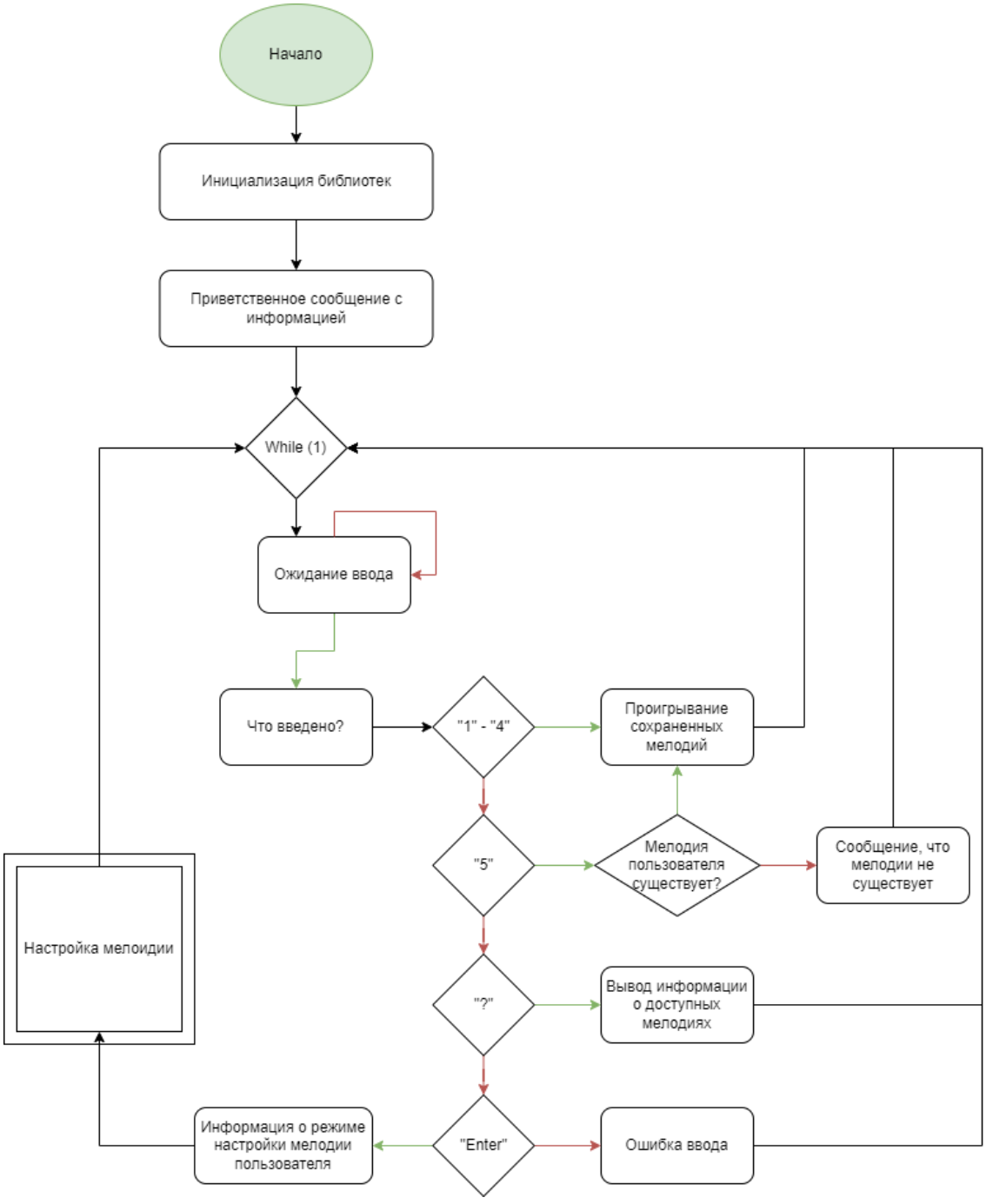
Действия стенда при получении символов от компьютера:

Символ	Действие
«1» – «4»	Воспроизведение одной из стандартных мелодий.
«5»	Воспроизведение пользовательской мелодии.
«Enter»	Вход в меню настройки.
На усмотрение исполнителей	Ввод параметров пользовательской мелодии: частота (нота, октава), длительность, конец мелодии и т.п.

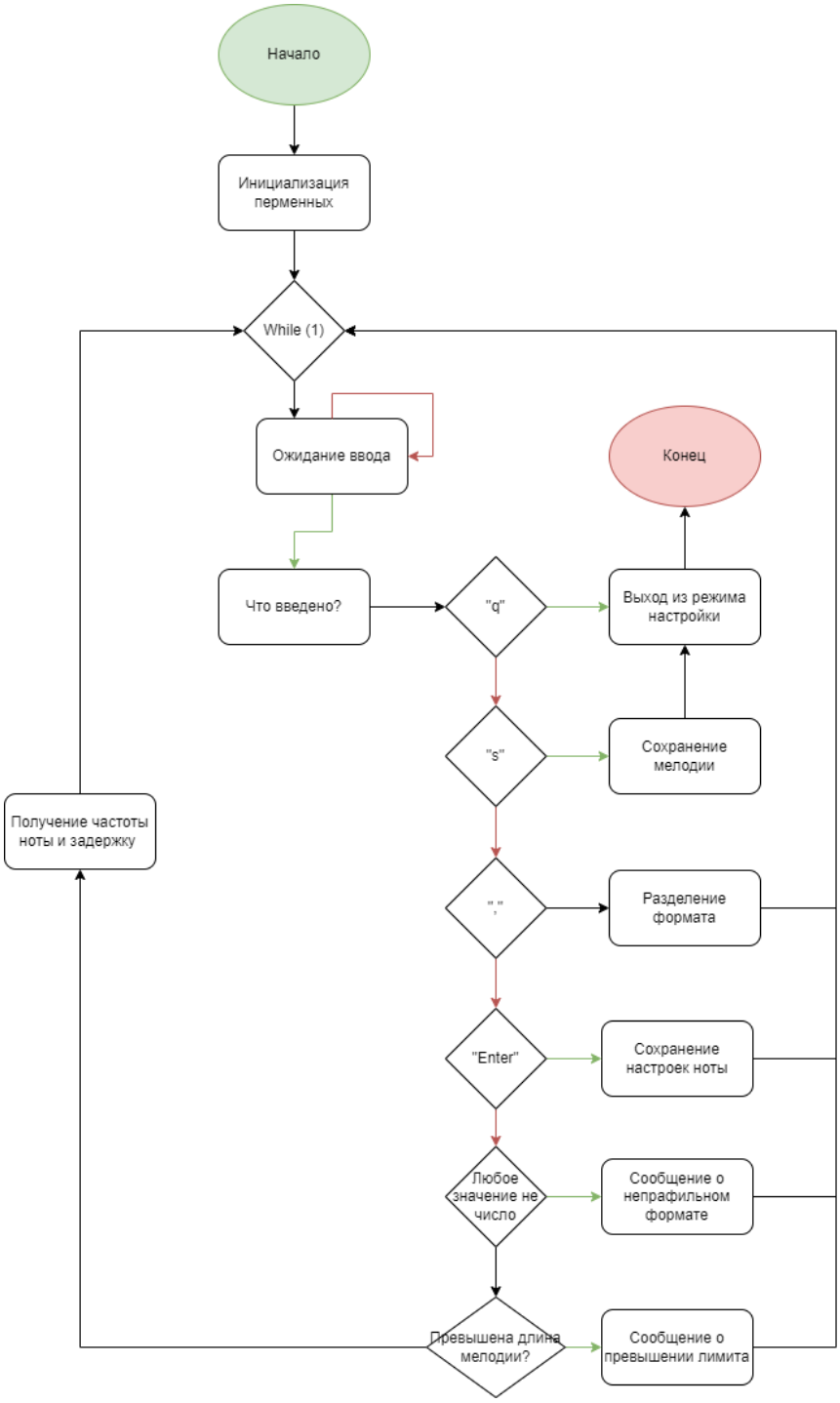
После ввода каждого символа в UART должно выводиться сообщение о том, какой режим активирован, или текущие настройки, вводимые в меню

Блок-схемы

Основная программа



Режим настройки мелодии



Исходный код

notes.h

```
#ifndef NOTES_H_
#define NOTES_H_

#include "stdint.h"

#define N_B0  31
#define N_C1  33
#define N_CS1 35
#define N_D1  37
#define N_DS1 39
#define N_E1  41
#define N_F1  44
#define N_FS1 46
#define N_G1  49
#define N_GS1 52
#define N_A1  55
#define N_AS1 58
#define N_B1  62
#define N_C2  65
#define N_CS2 69
#define N_D2  73
#define N_DS2 78
#define N_E2  82
#define N_F2  87
#define N_FS2 93
#define N_G2  98
#define N_GS2 104
#define N_A2  110
#define N_AS2 117
#define N_B2  123
#define N_C3  131
#define N_CS3 139
#define N_D3  147
#define N_DS3 156
#define N_E3  165
#define N_F3  175
#define N_FS3 185
#define N_G3  196
#define N_GS3 208
#define N_A3  220
#define N_AS3 233
#define N_B3  247
#define N_C4  262
#define N_CS4 277
#define N_D4  294
#define N_DS4 311
#define N_E4  330
#define N_F4  349
```

```
#define N_FS4 370
#define N_G4  392
#define N_GS4 415
#define N_A4  440
#define N_AS4 466
#define N_B4  494
#define N_C5  523
#define N_CS5 554
#define N_D5  587
#define N_DS5 622
#define N_E5  659
#define N_F5  698
#define N_FS5 740
#define N_G5  784
#define N_GS5 831
#define N_A5  880
#define N_AS5 932
#define N_B5  988
#define N_C6  1047
#define N_CS6 1109
#define N_D6  1175
#define N_DS6 1245
#define N_E6  1319
#define N_F6  1397
#define N_FS6 1480
#define N_G6  1568
#define N_GS6 1661
#define N_A6  1760
#define N_AS6 1865
#define N_B6  1976
#define N_C7  2093
#define N_CS7 2217
#define N_D7  2349
#define N_DS7 2489
#define N_E7  2637
#define N_F7  2794
#define N_FS7 2960
#define N_G7  3136
#define N_GS7 3322
#define N_A7  3520
#define N_AS7 3729
#define N_B7  3951
#define N_C8  4186
#define N_CS8 4435
#define N_D8  4699
#define N_DS8 4978

#define VOLUME_MAX 10
#define VOLUME_MIN 0

// https://vk.cc/cirJEC
uint32_t starwars_melody[] = {
    N_E4, N_E4, N_E4,
    N_C4, N_G4, N_E4, 0,
    N_C4, N_G4, N_E4,
```

```
    N_B5, N_B5, N_B5,
    N_C5, N_G4, N_E4, 0,
    N_C4, N_G4, N_E4
};

uint32_t starwars_delays[] = {
    218, 218, 218,
    218, 109, 109, 30,
    218, 109, 109,

    218, 218, 218,
    218, 109, 109, 30,
    218, 109, 109,
};

/*
uint32_t starwars2_melody[] = {
    N_A4, N_A4, N_A4, N_F4,
    N_CS4, N_A4, N_F4, N_CS4,
    N_A4,
    N_E4, N_E4, N_E4,
    N_F4, N_C4, N_GS4, N_F4,
    N_C4, N_A4,

    N_A4, N_A4, N_A4, N_A4,
    N_GS4, N_G4, N_FS4, N_F4,
    N_FS4,
    N_AS4, N_DS4, N_D4, N_CS4,
    N_C4, N_B4, N_C4,

    N_F4, N_GS4, N_F4, N_A4, N_C4, N_E4,

    N_A4, N_A4, N_A4, N_A4,
    N_GS4, N_G4, N_FS4, N_F4,
    N_FS4,
    N_AS4, N_DS4, N_D4, N_CS4,
    N_C4, N_B4, N_C4,

    N_F4, N_GS4, N_F4, N_C4, N_A4, N_F4, N_C4, N_A4, N_C4, N_A4, N_A4
};

uint32_t starwars2_melody[] = {
    500, 500, 500, 350,
    150, 500, 350, 150,
    650,
    500, 500, 500,
    350, 150, 500, 350,
    150, 1000

    500, 300, 150, 500,
    325, 175, 125, 125,
    500,
    250, 500, 325, 175,
    125, 125, 500,

    250, 500, 350, 125, 500, 375,
```

```
    500, 300, 150, 500,
    325, 175, 125, 125,
    500,
    250, 500, 325, 175
    125, 125, 500,

    250, 500, 375, 125, 500, 375, 125, 1000, 125, 1000, 1000
};
*/

uint32_t antoshka_melody[] = {
    N_E6, N_D6, N_D6, 0, N_E6, N_C6, N_C6,
    N_E6, N_D6, N_D6, N_B5, N_G5, N_E6, N_C6,

    N_E6, N_D6, N_D6, 0, N_E6, N_C6, N_C6,
    N_E6, N_D6, N_E6, N_D6, N_C6, N_B5, N_A5, N_G5,

    N_E6, N_F6, N_E6, N_D6, 0, N_D6, N_E6, N_D6, N_C6,

    N_E6, N_F6, N_E6, N_D6, N_C6, N_B5, N_D6, N_C6,
    N_E6, N_F6, N_E6, N_D6, N_C6, N_B5, N_D6, N_C6,

    N_E6, N_D6, N_B5, N_C6, 0, N_E6, N_D6, N_B5, N_C6
};

uint32_t antoshka_delays[] = {
    240, 240, 240, 60, 240, 240, 240,
    120, 120, 120, 120, 120, 120, 120,

    240, 240, 240, 60, 240, 240, 240,
    120, 120, 120, 120, 120, 120, 120,

    240, 240, 240, 240, 90, 240, 240, 240, 240,

    120, 120, 120, 120, 120, 120, 120, 120,
    120, 120, 120, 120, 120, 120, 120, 120,

    240, 240, 240, 240, 90, 240, 240, 240, 240,
};

// Взято из примеров проектов на SDK-1.1M
uint32_t megalovania_melody[] = {
        N_D3, N_D3, N_D4, N_A3, 0, N_GS3, N_G3, N_F3, N_D3, N_F3, N_G3, N_C3, N_C3, N_D4, N_A3,
0, N_GS3, N_G3, N_F3, N_D3, N_F3, N_G3, N_B2, N_B2, N_D4, N_A3, 0, N_GS3, N_G3, N_F3, N_D3,
N_F3, N_G3, N_AS2, N_AS2, N_D4, N_A3, 0, N_GS3, N_G3, N_F3, N_D3, N_F3, N_G3, N_D3, N_D3, N_D4,
N_A3, 0, N_GS3, N_G3, N_F3, N_D3, N_F3, N_G3, N_C3, N_C3, N_D4, N_A3, 0, N_GS3, N_G3, N_F3,
N_D3, N_F3, N_G3, N_B2, N_B2, N_D4, N_A3, 0, N_GS3, N_G3, N_F3, N_D3, N_F3, N_G3, N_AS2, N_AS2,
N_D4, N_A3, 0, N_GS3, N_G3, N_F3, N_D3, N_F3, N_G3, N_D4, N_D4, N_D5, N_A4, 0, N_GS4, N_G4,
N_F4, N_D4, N_F4, N_G4, N_C4, N_C4, N_D5, N_A4, 0, N_GS4, N_G4, N_F4, N_D4, N_F4, N_G4, N_B3,
N_B3, N_D5, N_A4, 0, N_GS4, N_G4, N_F4, N_D4, N_F4, N_G4, N_AS3, N_AS3, N_D5, N_A4, 0, N_GS4,
N_G4, N_F4, N_D4, N_F4, N_G4, N_D4, N_D4, N_D5, N_A4, 0, N_GS4, N_G4, N_F4, N_D4, N_F4, N_G4,
N_C4, N_C4, N_D5, N_A4, 0, N_GS4, N_G4, N_F4, N_D4, N_F4, N_G4, N_B3, N_B3, N_D5, N_A4, 0,
N_GS4, N_G4, N_F4, N_D4, N_F4, N_G4, N_AS3, N_AS3, N_D5, N_A4, 0, N_GS4, N_G4, N_F4, N_D4,
N_F4, N_G4, N_F4, N_F4, N_F4, N_F4, N_F4, N_D4, N_D4, N_D4, N_F4, N_F4, N_F4, N_G4, N_GS4,
N_G4, N_F4, N_D4, N_F4, N_G4, 0, N_F4, N_F4, N_F4, N_G4, N_GS4, N_A4, N_C5, N_A4, N_D5, N_D5,
```

```
N_D5, N_A4, N_D5, N_C5, N_F4, N_F4, N_F4, N_F4, N_F4, N_D4, N_D4, N_D4, N_F4, N_F4, N_F4, N_F4,
N_D4, N_F4, N_E4, N_D4, N_C4, 0, N_G4, N_E4, N_D4, N_D4, N_D4, N_D4, N_F3, N_G3, N_AS3, N_C4,
N_D4, N_F4, N_C5, 0, N_F4, N_D4, N_F4, N_G4, N_GS4, N_G4, N_F4, N_D4, N_GS4, N_G4, N_F4, N_D4,
N_F4, N_F4, N_F4, N_GS4, N_A4, N_C5, N_A4, N_GS4, N_G4, N_F4, N_D4, N_E4, N_F4, N_G4, N_A4,
N_C5, N_CS5, N_GS4, N_GS4, N_G4, N_F4, N_G4, N_F3, N_G3, N_A3, N_F4, N_E4, N_D4, N_E4, N_F4,
N_G4, N_E4, N_A4, N_A4, N_G4, N_F4, N_DS4, N_CS4, N_DS4, 0, N_F4, N_D4, N_F4, N_G4, N_GS4,
N_G4, N_F4, N_D4, N_GS4, N_G4, N_F4, N_D4, N_F4, N_F4, N_F4, N_GS4, N_A4, N_C5, N_A4, N_GS4,
N_G4, N_F4, N_D4, N_E4, N_F4, N_G4, N_A4, N_C5, N_CS5, N_GS4, N_GS4, N_G4, N_F4, N_G4, N_F3,
N_G3, N_A3, N_F4, N_E4, N_D4, N_E4, N_F4, N_G4, N_E4, N_A4, N_A4, N_G4, N_F4, N_DS4, N_CS4,
N_DS4,
};
uint32_t megalovania_delays[] = {

120,120,240,320,60,240,240,240,120,120,120,120,120,240,320,60,240,240,240,120,120,120,120,120,
240,320,60,240,240,240,120,120,120,120,120,240,320,60,240,240,240,120,120,120,120,120,240,320,
60,240,240,240,120,120,120,120,120,240,320,60,240,240,240,120,120,120,120,120,240,320,60,240,2
40,240,120,120,120,120,120,240,320,60,240,240,240,120,120,120,120,120,240,320,60,240,240,240,1
20,120,120,120,120,240,320,60,240,240,240,120,120,120,120,120,240,320,60,240,240,240,120,120,1
20,120,120,240,320,60,240,240,240,120,120,120,120,120,240,320,60,240,240,240,120,120,120,120,1
20,240,320,60,240,240,240,120,120,120,120,120,240,320,60,240,240,240,120,120,120,120,120,240,3
20,60,240,240,240,120,120,120,240,120,240,240,240,240,480,120,240,120,240,240,240,120,120,120,
120,120,240,240,120,240,240,240,240,240,240,240,240,120,120,120,960,240,120,240,240,240,240,48
0,120,240,120,240,240,240,240,240,120,240,120,240,240,240,240,240,240,240,240,120,240,128,240,240,
960,640,120,120,120,120,120,120,120,120,120,120,120,120,240,960,120,240,120,240,120,120,120,12
0,120,120,240,240,240,240,240,240,120,120,120,960,240,240,240,240,480,480,480,480,480,480,960,
240,240,240,240,960,960,640,120,120,120,120,120,120,120,120,120,120,120,120,120,240,960,120,240,12
0,240,120,120,120,120,120,120,240,240,240,240,240,240,120,120,120,960,240,240,240,240,480,480,
480,480,480,480,960,240,240,240,240,960,1920
};

uint32_t zelda_melody[] = {
        N_AS4, 0, 0, N_AS4, N_AS4, N_AS4, N_AS4, N_AS4, 0, N_GS4, N_AS4, 0, 0, N_AS4, N_AS4,
N_AS4, N_AS4, N_AS4, 0, N_GS4, N_AS4, 0, 0, N_AS4, N_AS4, N_AS4, N_AS4, N_AS4, N_F3, N_F3,
N_F3, N_F3, N_F3, N_F3, N_F3, N_F3, N_F3, N_F3, N_AS4, N_F3, N_F3, 0, N_AS4, N_AS4, N_C5, N_D5,
N_DS5, N_F5, 0, N_F5, N_F5, N_FS5, N_GS5, N_AS5, 0, N_AS5, N_AS5, N_AS5, N_GS5, N_FS5, N_GS5,
0, N_FS5, N_F5, N_F5, N_DS5, N_DS5, N_F5, N_FS5, N_F5, N_DS5, N_CS5, N_CS5, N_DS5, N_F5, N_DS5,
N_CS5, N_C5, N_C5, N_D5, N_E5, N_G5, N_F5, N_F3, N_F3, N_F3, N_F3, N_F3, N_F3, N_F3, N_F3,
N_F3, N_F3, N_AS4, N_F3, N_F3, 0, N_AS4, N_AS4, N_C5, N_D5, N_DS5, N_F5, 0, N_F5, N_F5, N_FS5,
N_GS5, N_AS5, 0, N_CS6, N_C6, N_A5, 0, N_F5, N_FS5, 0, N_AS5, N_A5, N_F5, 0, N_F5, N_FS5, 0,
N_AS5, N_A5, N_F5, 0, N_D5, N_DS5, 0, N_FS5, N_F5, N_CS5, 0, N_AS4, N_C5, N_C5, N_D5, N_E5, 0,
N_G5, N_F5, N_F3, N_F3, N_F3, N_F3, N_F3, N_F3, N_F3, N_F3, N_F3, N_F3, N_AS4, N_F3, N_F3, 0,
N_AS4, N_AS4, N_C5, N_D5, N_DS5, N_F5, 0, N_F5, N_F5, N_FS5, N_GS5, N_AS5, 0, N_AS5, N_AS5,
N_AS5, N_GS5, N_FS5, N_GS5, 0, N_FS5, N_F5, N_F5, N_DS5, N_DS5, N_F5, N_FS5, N_F5, N_DS5,
N_CS5, N_CS5, N_DS5, N_F5, N_DS5, N_CS5, N_C5, N_C5, N_D5, N_E5, N_G5, N_F5, N_F3, N_F3, N_F3,
N_F3, N_F3, N_F3, N_F3, N_F3, N_F3, N_F3, N_AS4, N_F3, N_F3, 0, N_AS4, N_AS4, N_C5, N_D5,
N_DS5, N_F5, 0, N_F5, N_F5, N_FS5, N_GS5, N_AS5, 0, N_CS6, N_C6, N_A5, 0, N_F5, N_FS5, 0,
N_AS5, N_A5, N_F5, 0, N_F5, N_FS5, 0, N_AS5, N_A5, N_F5, 0, N_D5, N_DS5, 0, N_FS5, N_F5, N_CS5,
0, N_AS4, N_C5, N_C5, N_D5, N_E5, 0, N_G5, N_F5, N_F3, N_F3, N_F3, N_F3, N_F3, N_F3, N_F3,
N_F3, N_F3, N_F3
};
uint32_t zelda_delays[] = {

960,240,240,240,240,240,240,320,120,120,480,240,240,240,240,240,240,320,120,120,480,240,240,24
0,240,240,240,240,120,120,240,120,120,240,120,120,240,240,480,480,320,120,120,120,120,120,120,
960,240,240,240,240,240,960,240,240,240,240,240,240,320,120,120,960,480,240,120,120,960,240,24
0,240,120,120,960,240,240,240,120,120,960,480,240,120,120,240,120,120,240,120,120,240,240,480,
```

```
480,320,120,120,120,120,120,120,960,240,240,240,240,240,960,480,480,480,480,480,480,960,480,48
0,480,480,480,480,960,480,480,480,480,480,480,960,480,480,480,480,480,480,240,120,120,480,480,
480,240,120,120,240,120,120,240,120,120,240,240,480,480,320,120,120,120,120,120,120,960,240,24
0,240,240,240,960,240,240,240,240,240,240,320,120,120,960,480,240,120,120,960,240,240,240,120,
120,960,240,240,240,120,120,960,480,240,120,120,240,120,120,240,120,120,240,240,480,480,320,12
0,120,120,120,120,120,960,240,240,240,240,240,960,480,480,480,480,480,480,960,480,480,480,480,
480,480,960,480,480,480,480,480,480,960,480,480,480,480,480,480,240,120,120,480,480,480,240,12
0,120,240,120,120,240,120,120,240,240
};

// Random.org
uint32_t start_melody[] = {1861, 0, 3009, 0, 1950, 0, 2486, 0, 2826};
uint32_t start_delays[] = {274, 60, 176, 60, 215, 60, 99, 60, 219};

uint32_t simple_melody[] = {
          N_D4, N_G4, N_FS4, N_A4,
          N_G4, N_C5, N_AS4, N_A4,
          N_FS4, N_G4, N_A4, N_FS4, N_DS4, N_D4,
          N_C4, N_D4,0,

          N_D4, N_G4, N_FS4, N_A4,
          N_G4, N_C5, N_D5, N_C5, N_AS4, N_C5, N_AS4, N_A4,      //29                //8
          N_FS4, N_G4, N_A4, N_FS4, N_DS4, N_D4,
          N_C4, N_D4,0,

          N_D4, N_FS4, N_G4, N_A4, N_DS5, N_D5,
          N_C5, N_AS4, N_A4, N_C5,
          N_C4, N_D4, N_DS4, N_FS4, N_D5, N_C5,
          N_AS4, N_A4, N_C5, N_AS4,             //58

          N_D4, N_FS4, N_G4, N_A4, N_DS5, N_D5,
          N_C5, N_D5, N_C5, N_AS4, N_C5, N_AS4, N_A4, N_C5, N_G4,
          N_A4, 0, N_AS4, N_A4, 0, N_G4,
          N_G4, N_A4, N_G4, N_FS4, 0,

          N_C4, N_D4, N_G4, N_FS4, N_DS4,
          N_C4, N_D4, 0,
          N_C4, N_D4, N_G4, N_FS4, N_DS4,
          N_C4, N_D4
};

uint32_t simple_delays[] = {
  240,480,240,480,
  480,480,480,180,
  480,480,480,480,480,480,
  480,120,480,

  240,480,240,480,
  480,960,1920,1920,960,1920,1920,180,
  480,480,480,480,480,480,
  480,120,480,

  480,480,480,480,480,480,
  480,480,480,180,
  480,480,480,480,480,480,
  480,480,480,180,
```

```
  480,480,480,480,480,480,
  960,1920,1920,960,1920,1920,480,240,480,
  960,360,480,920,360,480,
  960,1920,1920,120,480,

  480,240,480,480,480,
  480,120,480,
  480,240,480,480,480,
  480,96
};

#endif
```

main.h

```
#include "notes.h"

uint32_t user_melody[256];
uint32_t user_delays[256];
uint32_t user_melody_size = 0;

uint32_t duration = 1;
bool melody_playing = true;
uint32_t i = 0;

// Начальная мелодия старта инициализация
uint32_t * melody = start_melody;
uint32_t * melody_delays = start_delays;
uint32_t melody_size = sizeof(start_melody) / sizeof (uint32_t);


// Установка проигрывающей мелодии
void play_melody(uint32_t * m, uint32_t * d, uint32_t size) {
    i = 0;
    melody = m;
    melody_delays = d;
    melody_size = size;
    melody_playing = true;
}

// Проигрывание мелодии
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef * htim) {
    if (htim->Instance == TIM6) {
        if (!melody_playing) {
            return;
        }
        if (duration > 0) duration--;
        if (duration == 0) {
            if (melody[i] == 0) {
```

```
                // Выключаем звук
                htim1.Instance->CCR1 = 0; // 0%
                duration = melody_delays[i];
            } else {
                // Проигрываем ноты
                duration = melody_delays[i];
                htim1.Instance->ARR = 90000000 / (melody[i] * htim1.Instance->PSC) - 1;
                htim1.Instance->CCR1 = htim1.Instance->ARR >> 1; // 50% громкость (100%) -
скважность
            }
            i++;
            if (i == melody_size){
                i = 0;
                duration = 1;
                melody_playing = false;

                // Выключаем звук
                htim1.Instance->CCR1 = 0; // 0%
            }
        }
    }
}
```

```
int main(void)
{
  HAL_Init();

  SystemClock_Config();

  MX_GPIO_Init();
  MX_TIM1_Init();
  MX_TIM6_Init();
  MX_USART6_UART_Init();

  HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
  HAL_TIM_Base_Start_IT(&htim6);

  bool prog_start = true;

  char empty_msg[] = {"\r"};
  char hello_msg[] = {""
          "Привествую в нашем простой музыкально шкатулке!\r\n"
  };
  char list_melody_msg[] = {
          "\r\nДоступные мелодии:\r\n"
          "\t1. - StarWars: Imperial March.\r\n"
          "\t2. - Undertale: Megalovania.\r\n"
          "\t3. - Zelda.\r\n"
          "\t4. - Simple Melody.\r\n"
          "\t5. - Ваша мелодия.\r\n"
          "\tEnter - Настройки вашей мелодии."
  };
```

```
  char stop_melody_msg[] = {
          " - Мелодия остановлена!"
  };

  // Сообщения выбора мелодии
  char invalid_option_msg[] = {" - Неправильный ввод!"};
  char starwars_msg[] = {" - Играет: \"StarWars: Imperial March\""};
  char megalovania_msg[] = {" - Играет: \"Undertale: Megalovania\""};
  char zelda_msg[] = {" - Играет: \"Zelda\""};
  char simple_msg[] = {" - Играет: \"Antoshka Melody\""};
  char user_msg[] = {" - Играет: \"Ваша мелодия\""};
  char user_bad_msg[] = {" - Вы не создали свою мелодию! :("};

  // Сообщения Настройки пользовательской мелодии
  char prompt_msg[] = {
          "Вы перешли в режим настройки вашей мелодии!\r\n"
          "Введите ваши ноты в данном формате:\r\n"
          "\tЧАСТОТА,ДЛИТЕЛЬНОСТЬ\r\n"
          "Нажмите 'q', чтобы выйти.\r\n"
          "Нажмите 's', чтобы сохранить мелодию и выйти."
  };
  char note_save_msg[] = {" - Нота сохранена!"};
  char invalid_input_msg[] = {" - Неправильный формат! Попробуйте снова."};
  char input_limit_reached_msg[] = { "\r\nВаша мелодия слишком длинная :(" };
  char save_msg[] = {" - Вы успешно сохранили вашу мелодию и вышли!"};
  char exit_msg[] = {" - Вы вышли из решима настройки вашей мелодии!"};

  enable_interrupt(&status);
  buf_init(&ringBufferRx);
  buf_init(&ringBufferTx);

  while (1)
  {
      if (prog_start){
              transmit_uart_nl(&status, hello_msg, sizeof(hello_msg));
              prog_start = false;
          }

      receive_uart(&status);
      char c[2];
      if (!buf_pop(&ringBufferRx, c)) {
          continue;
      }

      switch (c[0]) {
          case '1': {
              transmit_uart_nl(&status, starwars_msg, sizeof(starwars_msg));
              play_melody(starwars2_melody, starwars2_delays, sizeof(starwars2_melody) / sizeof
(uint32_t));
              break;
          }
          case '2': {
              transmit_uart_nl(&status, megalovania_msg, sizeof(megalovania_msg));
              play_melody(megalovania_melody, megalovania_delays, sizeof(megalovania_melody) /
sizeof (uint32_t));
              break;
```

```
            }
            case '3': {
                transmit_uart_nl(&status, zelda_msg, sizeof(zelda_msg));
                play_melody(zelda_melody, zelda_delays, sizeof(zelda_melody) / sizeof
(uint32_t));
                break;
            }
            case '4': {
                transmit_uart_nl(&status, simple_msg, sizeof(simple_msg));
                play_melody(antoshka_melody, antoshka_delays, sizeof(antoshka_melody) / sizeof
(uint32_t));
                break;
            }
            case '5': {
                if (user_melody_size != 0){
                    transmit_uart_nl(&status, user_msg, sizeof(user_msg));
                    play_melody(user_melody, user_delays, sizeof(user_melody) / sizeof
(uint32_t));
                    break;
                }
                transmit_uart_nl(&status, user_bad_msg, sizeof(user_bad_msg));
                break;
            }
            case '?': {
                transmit_uart_nl(&status, list_melody_msg, sizeof(list_melody_msg));
                break;
            }
            case 'x': {
                play_melody(stop_melody, stop_delays, sizeof(stop_melody) / sizeof (uint32_t));
                transmit_uart_nl(&status, stop_melody_msg, sizeof(stop_melody_msg));
                break;
            }
            case '\r': {
                transmit_uart_nl(&status, prompt_msg, sizeof(prompt_msg));

                uint32_t cur_melody[256];
                uint32_t cur_delays[256];
                uint32_t cur_size = 0;

                uint32_t note = 0;
                uint32_t delay = 0;
                bool comma_encountered = 0;

                while (1) {
                    receive_uart(&status);
                    char c[2];
                    while (!buf_pop(&ringBufferRx, c));

                    if (c[0] == 'q') {
                        transmit_uart_nl(&status, exit_msg, sizeof(exit_msg));
                        break;
                    }

                    if (c[0] == 's') {
                        transmit_uart_nl(&status, save_msg, sizeof(save_msg));
                        memcpy(user_melody, cur_melody, sizeof(user_melody) / sizeof (uint32_t));
```

```
                    memcpy(user_delays, cur_delays, sizeof(user_melody) / sizeof (uint32_t));
                    user_melody_size = sizeof(user_melody) / sizeof (uint32_t);
                    break;
                }

                if (c[0] == ',') {
                    if (comma_encountered) {
                        transmit_uart_nl(&status, invalid_input_msg,
sizeof(invalid_input_msg));
                        note = 0;
                        delay = 0;
                        comma_encountered = false;
                        continue;
                    }
                    comma_encountered = true;
                    continue;
                }

                if (c[0] == '\r') {
                    cur_melody[cur_size] = note;
                    cur_delays[cur_size] = delay;
                    cur_size++;

                    note = 0;
                    delay = 0;
                    comma_encountered = false;
                    continue;
                }

                if (!isdigit(c[0])) {
                    transmit_uart_nl(&status, invalid_input_msg, sizeof(invalid_input_msg));
                    note = 0;
                    delay = 0;
                    comma_encountered = false;
                    continue;
                }

                if (cur_size == 256) {
                    transmit_uart_nl(&status, input_limit_reached_msg,
sizeof(input_limit_reached_msg));
                }

                if (comma_encountered) {
                    delay = delay * 10 + (c[0] - 48);
                } else {
                    note = note * 10 + (c[0] - 48);
                }
            }
            break;
        }
        default: {
            transmit_uart_nl(&status, invalid_option_msg, sizeof(invalid_option_msg));
            break;
        }
    }
}
```

Вывод

В ходе выполнения лабораторной работы мы реализовали простую музыкальную шкатулку, которая умеет проигрывать 4 исходные мелодии, а также пользовательскую. Добавили режим настройки пользовательской мелодии. Также нам пришлось использовать код из прошлой лабораторной работы, чтобы была возможность взаимодействия с UART.