Федеральное государственное автономное образовательное учреждение высшего образования

Университет ИТМО

Дисциплина: Проектирование вычислительных систем

Лабораторная работа 4

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 интерфейс I2C для того, чтобы корректно считывать нажатие кнопок клавиатуры стенда SDK-1.1. Подсистема опроса клавиатуры должна удовлетворять следующим требованиям:

- реализуется защита от дребезга;
- нажатие кнопки фиксируется сразу после того, как было обнаружено, что кнопка нажата (с учетом защиты от дребезга), а не в момент отпускания кнопки; если необходимо, долгое нажатие может фиксироваться отдельно;
- кнопка, которая удерживается дольше, чем один цикл опроса, не считается повторно нажатой до тех пор, пока не будет отпущена (нет переповторов);
- распознается и корректно обрабатывается множественное нажатие (при нажатии более чем одной кнопки считается, что ни одна кнопка не нажата, если это не противоречит требованиям к программе);
- всем кнопкам назначаются коды от 1 до 12 (порядок на усмотрение исполнителей).

Программа должна иметь два режима работы, переключение между которыми производится по нажатию кнопки на боковой панели стенда:

- режим тестирования клавиатуры;
- прикладной режим.

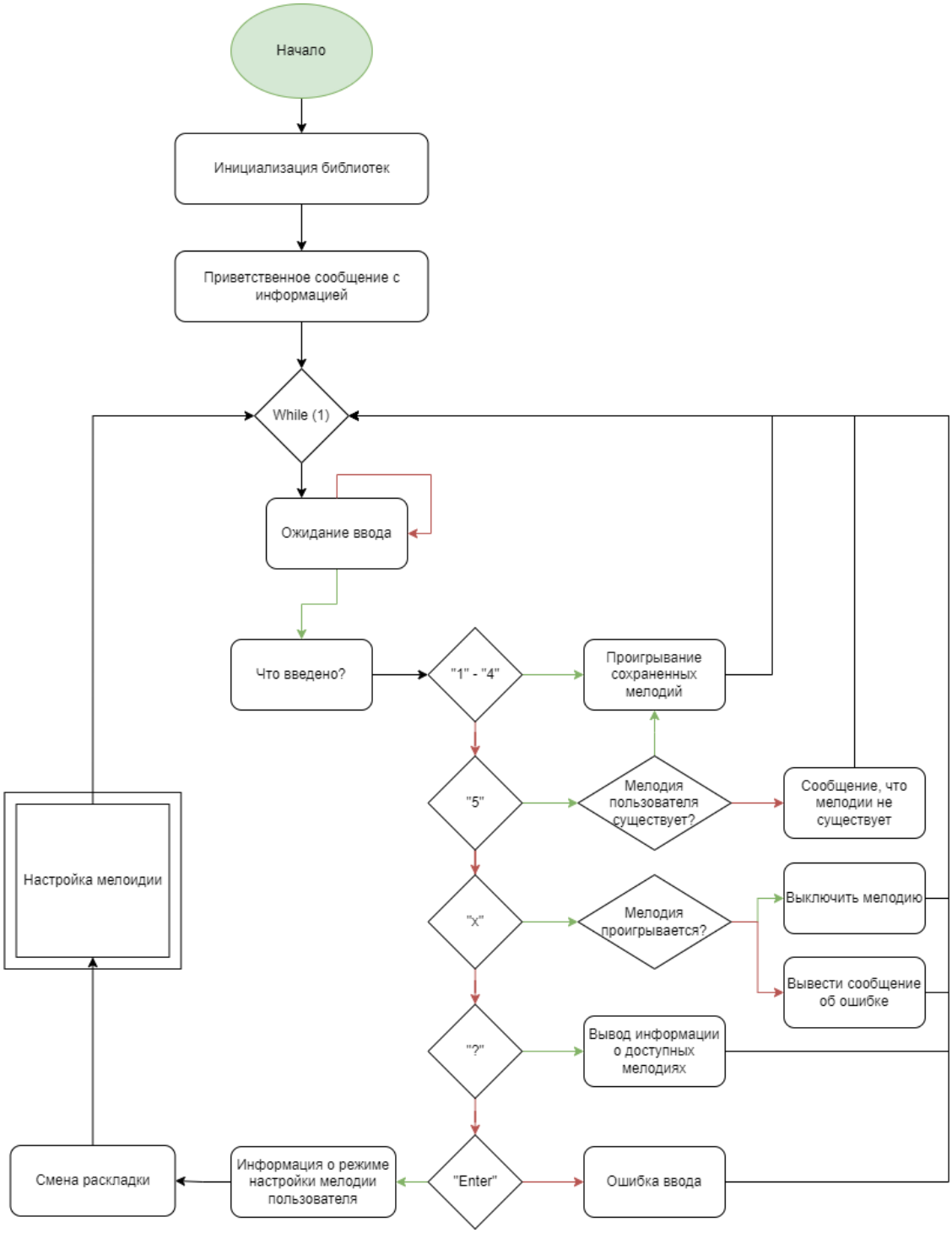
Уведомление о смене режима выводится в UART.
1) В режиме тестирования клавиатуры программа выводит в UART коды нажатых кнопок.
2) В прикладном режиме программа обрабатывает нажатия кнопок и выполняет действия в соответствии с вариантом задания.

Вариант 4

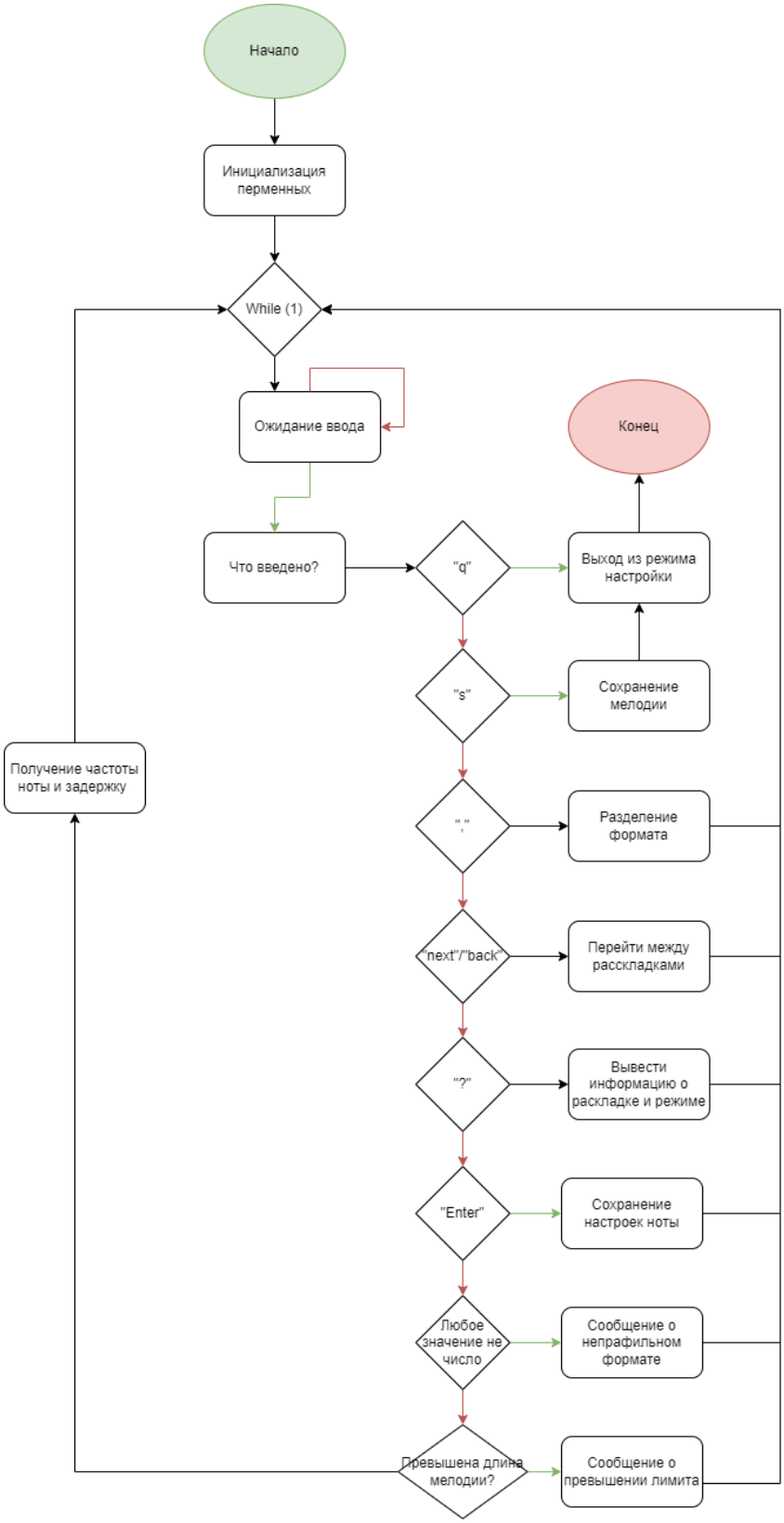
Задания аналогичны вариантам лабораторной работы №3, за исключением того, что ввод символов должен выполняться не с клавиатуры через UART, а с помощью клавиатуры стенда. Выбор кнопок клавиатуры стенда, играющих роль кнопок клавиатуры компьютера должен выполняться по усмотрению исполнителей. В отчете необходимо привести описание функций кнопок в реализованной программе.

Блок-схемы

Основная программа



Режим настройки мелодии



Описание функций кнопок

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.
«x»	Остановить мелодию
«?»	Вывести информацию о режиме и раскладке.
«q»	Выйти без сохранения из настройки пользовательской мелодии.
«s»	Выйти с сохранением из настройки пользовательской мелодии.
«next» / «back»	Смена раскладки.

Музыкальная раскладка

1	**2**	**3**
4	**5**	**x**
?		**enter**

Пользовательская настройка раскладка - 1 и 2

1	**2**	**3**
4	**5**	**6**
7	**8**	**9**
,	**0**	**next**

enter	**q**	**s**
?		**back**

Исходный код

main.c

```
// -------------------- KEYBOARD --------------------
bool is_music_mode = true;
bool is_settings_2 = false;
uint32_t last_pressing_time = 0;
int last_pressed_btn_index = -1;

bool is_co = false;
bool is_test_keyboard_mode = false;
bool last_btn_state = false;

char music[] = {'1', '2', '3', '4', '5', 'x', '!', '!', '!', '?', '!', '\r'};
char settings[] = {'1', '2', '3', '4', '5', '6', '7', '8', '9', ',', '0', 'e'};
char save_exit[] = {'\r', 'q', 's', '!', '!', '!', '!', '!', '!', '?', '!', 'e'};

bool is_btn_press() {
     return HAL_GPIO_ReadPin(GPIOC,GPIO_PIN_15) == 0;
 }

int get_pressed_btn_index(){
    const uint32_t t = HAL_GetTick();
    if (t - last_pressing_time < KB_KEY_DEBOUNCE_TIME) return -1;
    int index = -1;
    uint8_t reg_buffer = ~0;
    uint8_t tmp = 0;
    HAL_I2C_Mem_Write(&hi2c1, KB_I2C_WRITE_ADDRESS, KB_OUTPUT_REG, 1, &tmp, 1,
KB_KEY_DEBOUNCE_TIME);
    for (int row = 0; row < 4; row++) {
        uint8_t buf = ~((uint8_t) (1 << row));
        HAL_I2C_Mem_Write(&hi2c1, KB_I2C_WRITE_ADDRESS, KB_CONFIG_REG, 1, &buf, 1,
KB_KEY_DEBOUNCE_TIME);
        HAL_Delay(100);
        HAL_I2C_Mem_Read(&hi2c1, KB_I2C_READ_ADDRESS, KB_INPUT_REG, 1, &reg_buffer, 1,
KB_KEY_DEBOUNCE_TIME);
        switch(reg_buffer >> 4){
            case 6: index = row * 3 + 1; break;
            case 5: index = row * 3 + 2; break;
            case 3: index = row * 3 + 3; break;
        }
    }
    if (index != -1) last_pressing_time = t;
    if (index == last_pressed_btn_index){
        return -1;
    }
    last_pressed_btn_index = index;

    return index;
}
```

```
char key2char(const int key){
    return is_music_mode ? music[key - 1] : (is_settings_2 ? save_exit[key - 1] : settings[key
- 1]);
}

static void set_green_led(bool on) { HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, on ? GPIO_PIN_SET :
GPIO_PIN_RESET); }

static void set_yellow_led(bool on) { HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, on ? GPIO_PIN_SET :
GPIO_PIN_RESET); }

static void set_red_led(bool on) { HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, on ? GPIO_PIN_SET :
GPIO_PIN_RESET); }


int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_TIM1_Init();
  MX_TIM6_Init();
  MX_USART6_UART_Init();
  MX_I2C1_Init();
  /* USER CODE BEGIN 2 */
  HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
  HAL_TIM_Base_Start_IT(&htim6);
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  bool prog_start = true;

  char empty_msg[] = {"\r"};
  char hello_msg[] = {""
          "Привествую в нашем простой музыкально шкатулке!\r\n"
```

```
};
char layout_music[] = {
        "\r\nМузыкальная раскладка:\r\n"
        "\t1 | 2 | 3\r\n"
        "\t----------\r\n"
        "\t4 | 5 | x\r\n"
        "\t----------\r\n"
        "\t- | - | -\r\n"
        "\t----------\r\n"
        "\t? | - | Enter\r\n"
};
char layout_settings[] = {
        "\r\nПользовательская раскладка 1:\r\n"
        "\t1 | 2 | 3\r\n"
        "\t----------\r\n"
        "\t4 | 5 | 6\r\n"
        "\t----------\r\n"
        "\t7 | 8 | 9\r\n"
        "\t----------\r\n"
        "\t, | 0 | next\r\n"
        "\r\n"
        "\r\nПользовательская раскладка 2:\r\n"
        "\te | q | s\r\n"
        "\t----------\r\n"
        "\t- | - | -\r\n"
        "\t----------\r\n"
        "\t- | - | -\r\n"
        "\t----------\r\n"
        "\t? | - | back\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"
        "\tx. - Завершить мелодию.\r\n"
        "\t? - Получить информацию.\r\n"
        "\tEnter - Настройки вашей мелодии."
};
char list_settings_msg[] = {
        "\r\nДоступные мелодии:\r\n"
        "\t0-9. - Цифры.\r\n"
        "\t',' - Разделитель\r\n"
        "\tnext. - Следующая раскладка.\r\n"
        "\tback. - Предыдущая раскладка\r\n"
        "\t?. - Получить информацию.\r\n"
        "\ts. - Сохранить мелодию и выйти.\r\n"
        "\tq - Выйти без сохранения.\r\n"
};

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! :("};
  char gg_msg[] = {" - Мелодия не играет!"};

  // 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,ДЛ?ТЕЛЬНОСТЬ\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!"};

  char test_keyboard_on[] = {"Тестовый режим запущен!"};
  char test_keyboard_off[] = {"Прикладной режим запущен!"};

  set_green_led(false);


  disable_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;
          }

      bool btn_state = is_btn_press();
      if (last_btn_state && !btn_state){
          is_test_keyboard_mode = !is_test_keyboard_mode;
          if (is_test_keyboard_mode) transmit_uart_nl(&status, test_keyboard_on,
sizeof(test_keyboard_on));
          else transmit_uart_nl(&status, test_keyboard_off, sizeof(test_keyboard_off));
      }
      last_btn_state = btn_state;

      int btn_index = get_pressed_btn_index();
      if (btn_index != -1) {
          char received_char[] = {key2char(btn_index)};

          if (is_test_keyboard_mode){
```

```
            transmit_uart_nl(&status, received_char, sizeof(received_char));
            continue;
        } else {
            transmit_uart(&status, received_char, sizeof(received_char));
        }

        switch (received_char[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, user_melody_size + 1);
                    break;
                }
                transmit_uart_nl(&status, user_bad_msg, sizeof(user_bad_msg));
                break;
            }
            case '?': {
                transmit_uart_nl(&status, layout_music, sizeof(layout_music));
                transmit_uart_nl(&status, list_melody_msg, sizeof(list_melody_msg));
                break;
            }
            case 'x': {
                if (!melody_playing) {
                    transmit_uart_nl(&status, gg_msg, sizeof(gg_msg));
                    break;
                }
                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': {
```

```
                    bool settings_start = true;

                    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;

                    set_green_led(true);

                    is_music_mode = false;

                    while (1) {
                        if (settings_start){
                            transmit_uart_nl(&status, prompt_msg, sizeof(prompt_msg));
                            transmit_uart_nl(&status, layout_settings, sizeof(layout_settings));
                            settings_start = false;
                        }

                        int btn_index = get_pressed_btn_index();
                        if (btn_index != -1) {
                            char received_char[] = {key2char(btn_index)};
                            if (received_char[0] != 'e'){
                                transmit_uart(&status, received_char, sizeof(received_char));
                            } else {
                                is_settings_2 = !is_settings_2;
                                set_red_led(is_settings_2);
                                set_green_led(!is_settings_2);
                                continue;
                            }

                            if (received_char[0] == '?'){
                                transmit_uart_nl(&status, layout_settings,
sizeof(layout_settings));
                                transmit_uart_nl(&status, list_settings_msg,
sizeof(list_settings_msg));
                                continue;
                            }

                            if (received_char[0] == 'q') {
                                transmit_uart_nl(&status, exit_msg, sizeof(exit_msg));
                                is_music_mode = true;
                                is_settings_2 = false;
                                set_green_led(0);
                                set_red_led(0);
                                break;
                            }

                            if (received_char[0] == 's') {
                                is_music_mode = true;
                                is_settings_2 = false;
                                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 = cur_size;
                                set_green_led(0);
                                set_red_led(0);
                                break;
                            }

                            if (received_char[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 (received_char[0] == '\r') {
                                transmit_uart_nl(&status, empty_msg, sizeof(empty_msg));
                                cur_melody[cur_size] = note;
                                cur_delays[cur_size] = delay;
                                cur_size++;

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

                            if (!isdigit(received_char[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 + (received_char[0] - 48);
                            } else {
                                note = note * 10 + (received_char[0] - 48);
                            }
                        }
                    }
                    break;
                default: {
```

```
                    transmit_uart_nl(&status, invalid_option_msg, sizeof(invalid_option_msg));
                    break;
                }
            }
            }
        }
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */

  }
  /* USER CODE END 3 */
}
```

Вывод

В ходе выполнения лабораторной работы мы перенесли нашу музыкальную шкатулку на клавиатуру, благодаря чему UART стал нужен только для отображения необходимой информации. Мы добавили различные виды раскладок используя I2C для разных режимов работы со шкатулкой, а также для ввода своей собственной мелодии.